Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: www.jscit.nit.ac.ir
Volume 6, Number 2, Fall & Winter 2018, pp. 1-12
Received: 2015/09/10; Revised: 2017/09/25; Accepted: 2018/02/05

Providing a new model for the distance between query words based on the minimal displacement

Javad Paksima[1*], Ali Mohammad Zare Bidoki[2] and Vali Derhami[3]
1*- Department of Computer, Payame Noor University, Iran.
2- Department of Computer, Yazd University, Yazd, Iran.
3- Department of Computer, Yazd University, Yazd, Iran.
[1*]Paksima@pnu.ac.ir, [2]Alizareh@yazd.ac.ir, and [3]Vderhami@yazd.ac.ir

Corresponding author address: Javad Paksima, Faculty of Computer Engineering, Payame Noor University , Tehran, Iran, Post Code : 19395-3697.

Abstract- Based on the researches performed on search engines, most user queries contain more than one word. For queries with more than one word, two models can be presented. In the first model, query words are assumed to be independent of each other, and in the second model, the place and the order of words are assumed to be dependent. Experiments show that there are dependencies among most query words. One of the parameters that can determine the dependency between query words is the distance between the query words in the document. In this paper, a new distance definition based on the minimum displacement of the document words in order to match the query is presented. Also, given that most ranking algorithms use the word frequency in the documents (Term Frequency) to score the documents and since there is no clear definition for this parameter for queries with more than one word; in this paper, the frequency of the occurrence of a phrase (Phrase Frequency) and Inverted Document Frequency are defined according to the new concept of distance and the proper algorithms are presented to calculate them. Also, the results of the proposed algorithm are compared with the algorithm implemented by the open source Lucene indexer, which shows a good increase in the mean accuracy.

Keywords- Search engine, Ranking, Distance, Proximity.

مجله علمی پژوهشی رایانش نرم و فن‌آوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: www.jscit.nit.ac.ir
جلد ۶، شماره ۲، پاییز و زمستان ۱۳۹۶ ، صفحه ۱-۱۲
دریافت: ۱۳۹۳/۰۹/۱۲، بازنگری: ۱۳۹۶/۰۷/۰۳، پذیرش:۱۳۹۶/۱۱/۱۶

ارائه یک مدل جدید از فاصله بین کلمات پرس و جو براساس حداقل جابجایی

جواد پاک سیما*[۱]، علی‌محمد زارع بیدکی[۲]، ولی درهمی[۳]
*۱- گروه کامپیوتر و فناوری اطلاعات، دانشگاه پیام نور، ایران.
*۲- دانشکده مهندسی برق و کامپیوتر، دانشگاه یزد، یزد، ایران.
۳- دانشکده مهندسی برق و کامپیوتر، دانشگاه یزد، یزد، ایران.
[1*]paksima@pnu.ac.ir, [2]Alizareh@yazd.ac.ir, and [3]Vderhami@yazd.ac.ir

* نشانی نویسنده مسئول: جواد پاک‌سیما، تهران، دانشگاه پیام نور، گروه کامپیوتر و فناوری اطلاعات، کد پستی: ۳۶۹۷-۱۹۳۹۵

چکیده- براساس تحقیقات انجام شده روی موتورهای جستجو، اکثر پرس‌وجوهای کاربران بیش از یک کلمه می‌باشد. برای پرس‌وجوها با بیش از یک کلمه دو مدل می‌توان ارائه کرد. در مدل اول فرض می‌شود کلمات پرس‌وجو مستقل از یکدیگر هستند و در مدل دوم محل و ترتیب کلمات وابسته فرض می شود. آزمایش‌ها نشان می‌دهد که در اکثر پرس‌وجوها بین کلمات وابستگی وجود دارد. یکی از پارامترهایی که می‌تواند وابستگی بین کلمات پرس‌وجو را مشخص کند فاصله‌ی بین کلمات پرس‌وجو در سند است. در این مقاله تعریف جدیدی از فاصله براساس حداقل جابجایی کلمات سند به‌منظور تطبیق بر پرس‌وجو ارائه می‌گردد. همچنین با توجه به این‌که اکثر الگوریتم‌های رتبه‌بندی از فرکانس رخداد یک کلمه در سند (Term Frequency) برای امتیازدهی به اسناد استفاده می‌کنند و برای پرس‌وجو با بیش از یک کلمه تعریف روشنی از این پارامتر وجود ندارد؛ در این مقاله پارامترهای فرکانس رخداد یک عبارت (Phrase Frequency) و معکوس فرکانس سند (Inverted Document Frequency) با توجه به مفهوم جدید فاصله تعریف شده است و الگوریتم‌هایی برای محاسبه آن‌ها ارائه گردیده است. همچنین نتایج الگوریتم پیشنهادی با الگوریتم پیاده سازی شده توسط نمایه ساز متن‌باز لوسین مقایسه شده است که افزایش خوبی را در میانگین دقت نشان می‌دهد.

واژه‌های کلیدی: موتور جستجو، رتبه‌بندی، فاصله، وابستگی کلمات.

۱- مقدمه

پیدا کردن صفحات وب دارای کیفیت بالا یکی از مهم‌ترین وظایف موتورهای جستجو می‌باشد. مفهوم میزان ارتباط اسناد پیداشده با پرس‌وجو وابسته به نظر کاربر می‌باشد و این موضوع باعث افزایش پیچیدگی الگوریتم‌های رتبه‌بندی می‌شود. نکته‌ی دیگر این است که غالباً کاربران ۱۰ تا ۲۰ نتیجه اول را بررسی می‌کنند[۱] درحالی‌که برای یک پرس‌وجو ممکن است میلیون‌ها صفحه‌ی مرتبط وجود داشته باشد. بنابراین موتورهای جستجو باید برای یافتن مرتبط‌ترین صفحات، الگوریتم مناسب با کارایی بالا ارائه نمایند.

بخش رتبه‌بندی یکی از مهم‌ترین قسمت‌های موتورهای جستجو می‌باشد. رتبه‌بندی فرآیندی است که در آن کیفیت صفحه توسط موتور جستجو تخمین زده می شود. در حال حاضر دو روش عمده برای رتبه‌بندی صفحات وب وجود دارد. در روش اول رتبه‌بندی براساس محتوای اسناد انجام می‌شود (رتبه‌بندی سنتی). مدل‌هایی مانند مدل بولی، مدل احتمالی و مدل فضای برداری جهت رتبه‌بندی اسناد مبتنی بر محتوا ارائه شده‌اند[۲]. در روش دوم براساس گراف و اتصالات وب و میزان اهمیت صفحات رتبه‌بندی صورت می‌گیرد.

در رتبه‌بندی سنتی، موتور جستجو سعی می‌کند با پیدا کردن میزان ارتباط سند و پرس‌وجو رتبه‌بندی را انجام دهد. در این

الگوریتم‌ها برای هر پرس‌وجو، اسناد با محتوای شبیه‌تر به کلمات موجود در پرس‌وجو امتیاز بالاتری را دارا می‌باشند. به‌عنوان‌مثال الگوریتم‌های TF-IDF[۳]، BM25[۴] دو نمونه از رایج‌ترین الگوریتم‌های این نوع رتبه‌بندی هستند.

وب شامل تعداد زیادی اسناد غیر ساخت‌یافته می‌باشد که به هم متصل هستند و یک گراف خیلی بزرگ را ایجاد می‌کند. معمولاً تعداد کلمات پرس‌وجوها کم بوده (2.4 کلمه در هر پرس‌وجو[۵]) و مجموعه‌ی کل لغات خیلی زیاد می‌باشد. این در حالی است که در اکثر الگوریتم‌های بازیابی اطلاعات معمولاً برای تعداد اسناد کم و تعداد کلمات پرس‌وجوی زیاد کارایی خوبی دارند[۱].

یک عبارت[۱]، لیستی از کلمات است که دارای ترتیب مشخصی هستند. به‌عنوان‌مثال "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].

در یک کار تحلیلی بر روی یک و نیم میلیون پرس‌وجو در موتور جستجوی Excite مشخص شد که ۸,۴ در صد از پرس‌وجوها شامل علامت نقل‌قول (' یا ") می‌باشند[۷]. تحقیقی دیگر نشان می‌دهد که مقصود کاربران در ۴۰ درصد از پرس‌وجوهایی که دو کلمه یا بیشتر هستند و علامت نقل‌قول هم ندارند عبارت بوده است نه کلمات مجزا[۸].

روش اصلی برای جستجوی یک عبارت استفاده از نمایه معکوس[۲] می‌باشد[۹]. هر ردیف در نمایه معکوس شامل یک کلمه می‌باشد که لیستی از سه‌تایی‌های مرتب شامل شماره سند، تعداد رخداد کلمه در سند و موقعیت‌هایی که در آن کلمه موردنظر در سند ظاهر شده است به‌صورت زیر می‌باشد:

$<d, tf_{d,t}, [p_1, p_2, \ldots.]>$

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 می‌باشد.

<101, 4, [5, 11, 25, 77]>

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

برای رتبه‌بندی اسناد، زمانی که پرس‌وجو فقط یک کلمه باشد پارامتر TF[۳] و IDF[۴] مهم‌ترین پارامترها برای رتبه‌بندی هستند. این پارامترها در روش‌های TF-IDF و BM25 استفاده شده‌اند. اگر بخواهیم از این روش‌ها برای جستجوی عبارت استفاده شود باید پارامترهای مشابهی برای عبارت نیز تعریف گردد.

در جستجوی عبارت اگر از علامت نقل‌قول استفاده شده باشد که ابهامی وجود ندارد و مشابه جستجوی تک کلمه می‌توان عمل کرد و الگوریتم‌های متعددی برای آن ارائه گردیده است[۱۰][۱۱]. اما اگر علامت نقل‌قول نباشد بهتر است بیشترین امتیاز برای اسنادی باشد که بیشترین تکرار عبارت با همان ترتیب و با کمترین فاصله در آن صورت گرفته است. به همین منظور در این مقاله پارامتری مشابه TF به نام PF[۵] تعریف شده است که فرکانس یک عبارت را نشان می‌دهد و الگوریتمی برای محاسبه IDF براساس PF ارائه می‌گردد.

در این مقاله در ابتدا کارهای مرتبط انجام شده توضیح داده می‌شود. سپس اصطلاحات استفاده شده، تعریف می‌گردد. سپس الگوریتم استفاده شده در نمایه‌ساز لوسین برای محاسبه PF و IDF تحلیل می‌شود. پس‌ازآن در بخش چهار الگوریتم پیشنهادی برای محاسبه PF و IDF معرفی شده است. در پایان مقایسه‌ای بین الگوریتم مبتنی بر فاصله‌ی استفاده شده در نمایه‌ساز لوسین و الگوریتم پیشنهادی انجام پذیرفته است.

۲- کارهای مرتبط

از قدیمی‌ترین کارها در مورد وابستگی کلمات، کار ریجس برگن[۶](۱۹۷۷) می‌باشد که به‌صورت نظری رتبه‌بندی اسناد را براساس مجاورت کلمات پرس‌وجو بررسی کرده است[۱۲]. در روش پیشنهادی وی میزان ارتباط کلمات پرس‌وجو براساس اطلاعات زوج کلمه مشخص می‌گردید و برای بازیابی از درخت پوشای ماکزیمم (MST[۷]) استفاده می‌شد. سال‌ها بعد در سال ۲۰۰۲ نالاپاتی[۸] و آلان[۹] ایده خود را در مورد نحوه مشخص کردن میزان ارتباط کلمات پرس‌وجو ارائه دادند[۱۳]. برای کاهش زمان، آن‌ها پیشنهاد کردند که بجای استفاده از آمار اسناد، از آمار جملات برای ساخت درخت پوشای ماکزیمم استفاده شود.

در سال ۱۹۹۱ کین[۱۰] نتایج تجربی خودش را منتشر کرد. او فرض کرد که موقعیت کلمات پرس‌وجو در سند با غربال اسناد غیر مرتبط می‌تواند نتایج را دقیق‌تر کند. کین 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. او هفت دیدگاه مختلف برای استفاده از مجاورت کلمات پیشنهاد داد. در مطالعه‌ی وی بهترین نتایج زمانی بدست می‌آمد که الگوریتم به کم بودن فاصله بین کلمات پرس‌وجو پاداش می‌داد[۱۴].

همچنین در سال ۱۹۹۱ کرافت[11] و همکارانش [۱۵] برای اولین بار روشی برای استفاده از شبکه استنتاج برای کلمات مجاور با عنوان InQuery ارائه دادند. آن‌ها همچنین کلمات با DF[12] بالا را از پرس‌وجو حذف کردند مثلاً کلمه "شهر" را از پرس‌وجوی "اماکن تفریحی شهر شیراز" حذف کردند. آن‌ها دقت خوبی را در صفحات ابتدایی جستجو بدست آورند ولی در کل دقت کاهش یافت. این موضوع بعداً در سال ۲۰۰۵ توسط متزلر[13] و کرافت تائید شد[۱۶]. دنگ[14] و همکارانش کار کرافت را دنبال کردند. آن‌ها براساس کلمات ظاهر شونده در پنجره‌ای که محل تمرکز کلمات پرس‌وجو است و همچنین بازخورد کاربر امتیاز اسناد را محاسبه می‌کردند.[17]

مدل‌های زبانی متنوعی برای وابستگی کلمات پرس‌وجو ارائه شده است. اولین بار در سال ۱۹۹۹ سانگ[15] و کرافت[۱۸] مدل تک‌کلمه‌ای[16] استاندارد را به‌وسیله درون‌یابی مدل زوج کلمه‌ای[17] توسعه دادند. در مقیاس آزمایشی کوچک، استفاده از زوج کلمات نتایج را بهبود بخشید. گائو[18] و همکارانش در سال ۲۰۰۴ مدل زبانی وابسته (DLM[19]) را پیشنهاد دادند[۱۹]. در این مدل آن‌ها اتصال بین توزیع مدل زبانی تک‌کلمه‌ای و اسناد شامل کلمات مجاور را در پنجره‌ای به طول سه تعریف کردند. آن‌ها با استفاده از مجموعه‌ای از اسناد، بیشترین شباهت پیوندها را برای کلمات متوالی پرس‌وجوها تخمین زدند. در مدل فقط زوج کلمات موردتوجه قرار گرفتند. این الگوریتم در عمل قابل‌استفاده نبود زیرا استخراج تمام ساختارهای پیوند برای یک پرس‌وجو در زمان جستجو فوق‌العاده زمان‌بر بود[۲۰].

گروه دیگری از محققان روش‌هایی برای افزودن وابستگی کلمات در مدل‌های احتمالی پیشنهاد دادند. مثلاً رسولف[20] و ساوی در سال ۲۰۰۳ روش BM25 را بر اساس وابستگی کلمات توسعه دادند[۲۱]. آن‌ها تابعی از فاصله را جایگزین پارامتر TF کردند. در روش آن‌ها فاصله زوج کلمات در یک پنجره پنج کلمه‌ای محاسبه می‌شد. هی[21] و همکارانش در سال ۲۰۱۱ از یک پنجره برای شمارش فرکانس n کلمه‌ای[22] در یک سند استفاده کردند و BM25 را تغییر دادند که از فرکانس برای محاسبه امتیاز استفاده کند. معیار فاصله در مدل پیشنهادی آن‌ها، حداقل تعداد کلمه‌ای بود که یک دنباله از کلمات شامل تمام کلمات پرس‌وجو را از هم جدا می‌کرد[۲۰]. در الگوریتم پیشنهادی از این ایده استفاده شده است و پارامتر فرکانس عبارت(PF) تعریف می‌شود تا به‌جای TF در الگوریتم‌های رتبه‌بندی مورداستفاده قرار گیرد.

Eickhoff و همکارانش با استفاده از ابزاری آماری به نام Copulas مدل آماری خود را توسعه دادند[22]. آن‌ها با استفاده Copulas احتمال هم‌رخدادی کلمات پرس‌وجو را بدست آوردند. مزیت اصلی روش آن‌ها سرعت بالای آن بود.

باتچر[23] و همکارانش در سال ۲۰۰۶ مدل تجمعی را پیشنهاد دادند که به‌طور مجزا امتیاز مجاورت پرس‌وجو را برای هر کلمه محاسبه می‌کرد[۲۳]. الگوریتم آن‌ها به‌طور خاص بر اساس فایل‌های ایندکس معکوس پیاده‌سازی شد. در موقع پردازش لیست موقعیت‌های[24] کلمات پرس‌وجو، اگر کلمه‌ای تغییر می‌کرد فاصله به‌صورت تجمعی افزوده می‌شد. کار مشابهی توسط تائو[25] و همکارش در سال ۲۰۰۷ انجام شد[۲۴]. آن‌ها پنج ویژگی مربوط به فاصله کلمات را به‌صورت تجمعی محاسبه کردند و به‌عنوان وزن کلمات در روش‌های مبتنی بر محتوا استفاده کردند. بهترین نتیجه زمانی بدست می‌آمد که معیار وزن براساس حداقل کردن فاصله بین تمام کلمات پرس‌وجو در نظر گرفته می‌شد.

بر اساس نتایج تائو و زای، زائو[26] و یان[27] در سال ۲۰۰۹ مدل زبانی کلمات مجاور (PLM[28]) را پیشنهاد دادند[۲۵]. در روش آن‌ها کمترین فاصله‌ها بین تمام کلمات پرس‌وجو در متن سند بدست می‌آمد و مجموع آن‌ها برای محاسبه امتیاز به‌کاربرده می‌شد.

کارهای متعدد دیگری در این زمینه انجام شده است[۲۶]. مطالب مرتبط دیگری هم مطرح است مثلاً جستجوی عبارت به‌طور دقیق یعنی زمانی که کاربر با استفاده از علائم نقل‌قول عبارت را مشخص می‌کند. موضوع دیگر توسعه‌ی پرس‌وجو[29] می‌باشد. به‌عنوان‌مثال میائو و همکارانش در سال ۲۰۱۲ از مفهوم کلمات مجاور با استفاده از بازخورد کاربر، توسعه پرس‌وجو را بهبود بخشیدند[۲۷].

۳- اصطلاحات استفاده‌شده

فاصله : حداقل تعداد جابجایی کلمات یک متن برای تطبیق بر عبارت پرس‌وجو را فاصله می‌نامیم. مثلاً اگر در سند ’a c b‘ عبارت ’a b‘ جستجو شود فاصله یک خواهد بود زیرا b موجود در سند باید یک جابجایی داشته باشد تا بر ’a b‘ منطبق شود (برای سادگی هر یک از حروف لاتین را یک کلمه فرض کرده‌ایم. یعنی سند فوق دارای سه کلمه a و c و b می‌باشد.). جستجوی عبارت ’b a‘ در سند فوق به دو جابجایی برای تطبیق نیاز دارد و بنابراین فاصله برابر با دو می‌شود. در حالتی که فاصله صفر باشد مفهوم آن تطابق کامل بدون نیاز به جابجایی است. مثلاً جستجوی ’c b‘ در سند فوق فاصله صفر را نتیجه می‌دهد.

البته این اصطلاح در روش‌های مختلفی که بخش قبلی آمد تعاریف متفاوتی دارد. برخی از محققین حداقل فاصله بین زوج کلمات را در نظر گرفته‌اند و برخی جمع فاصله و روش‌های دیگر برای محاسبه فاصله بکار برده‌اند.

تابع s(d) : تابع s(d) تابعی اســـت که فاصـــله را به امتیاز تبدیل می‌کند. ورودی این تابع فاصـــله می‌باشـــد. این تابع باید با افزایش فاصله کاهش یابد و همچنین در صفر برابر یک باشد. یعنی:

s(0)=1

If d1>d2 then s(d1)<s(d2)

به‌عنوان‌مثال s(d)=1/(d+1) ویژگی های فوق را دارد و در ارز یابی عبارت قابل‌ا ستفاده ا ست[۲۸]. یا در [۲۱] امتیاز به صورت عکس مربع فاصله محاسبه شده است یا در [۲۴] از یک رابطه‌ی لگاریتمی استفاده شده است.

PF : برای مطابقت دادن تعداد عبارت در یک سند با روش‌هایی که از TF برای محاســـبه امتیاز اســـتفاده می‌کنند پارامتری به نام PF تعریف می شود که مخفف Phrase Frequency می‌با شد. TF همواره یک عدد صحیح می‌باشد ولی PF می‌تواند یک عدد اعشاری باشد. PF به‌صورت زیر قابل‌محاسبه می‌باشد:

$$PF = \sum_{i=1}^{n} s(d_i) \qquad (۱)$$

که در آن n تعداد عبارت پرس‌وجویی ا ست که در سند پیدا شده اســت و d_i مقدار فاصـله برای عبارت i ام می‌باشـد. در حالت بهینه ترکیب عبارات باید به‌گونه‌ای انتخاب شوند که PF ماکزیمم باشد.

QMT[۳۰] : اکثر موتورهای جســـتجو برای پرس‌وجوها حداکثر تعداد کلمه را مشـــخص می‌کنند. مثلاً در موتور گوگل حداکثر تعداد کلمات برابر ۳۲ می‌با شد[۲۹]. این متغیر حداکثر تعداد کلمه مجاز در پرس‌وجو را مشخص می‌کند.

۴- الگوریتم استفاده‌شده در لوسین

نمایه‌ســـاز لوســـین[۳۱] یکی از ابزار متن‌باز می‌باشـــد که به زبان‌های مختلف مثل C# و جاوا پیاده‌ســـازی شـــده اســـت. این ابزار با کتابخانه هایی که در اختیار کاربر قرار می‌د هد امکان نما یه و ج ستجو را فراهم می‌کند. روش‌های مختلفی برای ج ستجو در این موتور پیش بینی شده ا ست مثل جستجوی یک کلمه، یک عبارت به‌طور دقیق، یک ع بارت به‌طور تقریبی[۳۲]، پرس‌وجوی بولین و موارد دیگر. روشـــی که با موضـــوع این مقاله مرتبط می‌باشـــد جستجوی عبارت به‌طور تقریبی می‌باشد.

لوســـین برای جســـتجو عبارت به‌طور تقریبی بدین‌صـــورت عمل می‌کند که از ابتدای متن ســـند کلمات را بررســـی می‌کند و یک پنجره روی کلمات ایجاد می‌کند. با پیدا کردن اولین کلمه در پرس‌وجو شروع پنجره را مشخص می‌کند. سپس متن را برای پیدا کردن تمام کلمات پرس‌وجو پیمایش می‌کند و زمانی که تمام کلمات پرس‌وجو پیدا شـد پایان پنجره را مشـخص می‌کند. در این حالت اولین انطباق رخ داده است و فاصله محاسبه می‌شود. در این حالت برای محاسبه فاصله، تا زمانی‌که تمام کلمات پرس‌وجو داخل پنجره ه ست، ابتدای پنجره را ا ضافه می‌کند. به فا صله پیدا شده یک واحد اضـــافه می‌کند و وارون حاصـــل را به‌عنوان امتیاز در نظر می‌گیرد. برای تطبیق ب عدی ا بتدای پنجره را افزایش می‌د هد و عمل فوق را برای جستجو انطباق‌های بعدی تکرار می‌کند. درنهایت PF به‌صورت زیر محاسبه می‌شود:

$$PF = \sum_{i=1}^{n} \frac{1}{1+d_i} \qquad (۲)$$

الگوریتم استفاده‌شده در لوسین یک الگوریتم حریصانه می‌باشد و همواره نتایج درستی را به دنبال ندارد. این موضوع توسط طراحان لوسین نیز پذیرفته شده است[۳۳]. عیب اساسی این روش همان‌طور که در توضیحات موجود در متن برنامه آمده است این است که ممکن است در شرایط خاصی تعداد عبارت، درست شمارش نشود. مثلاً اگر در سند ’a a a b b b‘ عبارت ’a b‘ جستجو شود یک عبارت بیشتر پیدا نمی‌شود (PF=1) درحالی‌که سه بار ’a b‘ با فاصله‌های مختلف در سند ظاهر شده است و باید طبق تعریف لوسین PF=1.53 باشد. یا مثلاً اگر در متن ’a b a‘ عبارت’a b‘ جستجو شود PF برابر با ۱،۳۳ خواهد شد درحالی‌که ’a b‘ فاصله صفر دارد. این در حالی است که اگر در متن ’b a b‘ عبارت ’a b‘ جستجو شود مقدار PF برابر با ۱ خواهد شد.

حســـن این الگوریتم ســـرعت بالای آن می‌باشـــد و در بدترین حالت جســـتجوی یک عبارت در ســـندی با n کلمه را در O(n) انجام می‌دهد.

رابطه زیر رابطه‌ای است که لوسین از آن استفاده می‌کند:

$$IDF(q) = \sum_{i=1}^{m} IDF(t_i) \qquad (۳)$$

در رابطه ۳، q عبارت مورد جســـتجو، t_i کلمه i ام موجود در عبارت و m تعداد کلمات عبارت است.

روش محاسبه IDF در لوسین دارای اشکال است زیرا از جمع IDF کلمات موجود در نمایه مقدار IDF عبارت را محاسبه می‌کند. طبق تعریف، DF مشخص کننده تعداد ا سنادی ا ست که کلمه موردنظر در آن وجود دارد و IDF با عکس DF رابطه مستقیمی دارد و نشان دهنده میزان اهمیت کلمه اســـت. این موضـــوع برای پرس‌وجوهای بیش از یک کلمه هم باید صـــادق باشـــد اما در فرمول پیشـــنهادی لوسین این موضوع صادق نیست. مثلاً اگر پرس‌وجو "شبکه یک" با شد DF کلمه "شبکه" مقدار بالایی ا ست و کلمه "یک" هم DF بالایی دارد و درنتیجه جمع IDF این دو کلمه مقدار کمی می شود اما عبارت "شبکه یک" دارای DF کمی ا ست و درنتیجه باید IDF بالایی داشـــته باشـــد. این مثال نشـــان‌دهنده‌ی ضـــعف رابطه ۳ در محاسبه‌ی IDF می‌باشد.

برای الگوریتم‌هایی که جستجو در فیلدهای مختلف با وزن‌های مختلف انجام می‌شود IDF نقش بیشتری دارد. مثلاً در الگوریتم BM25f براساس هر پرس‌وجو، جستجوها در فیلدهای بدنه، عنوان، متن ارجاع[۳۴] و... انجام می‌شود و با ترکیب امتیازهای حاصل شده از بخش‌های مختلف امتیاز کلی محاسبه می‌شود[۳۰].

۵- الگوریتم پیشنهادی

در الگوریتم پیشنهادی دو موضوع را دنبال می‌کنیم، یکی محاسبه فاصله و دیگر محاسبه PF. همچنین IDF هم براساس تعریف PF با رابطه‌ای واقعی‌تر ارائه می‌شود. در موقع محاسبه فاصله فرض می‌گیریم که در سند فقط یک‌بار کلمات عبارت ظاهر شده است و سپس مکانی را برای جمع شدن کلمات می‌یابیم که جابجایی‌ها حداقل باشد. در بخش دوم الگوریتمی با استفاده از روش جستجوی کامل ارائه می‌دهیم که تمام ترکیب‌های عبارات را ا ستخراج کند و ترکیبی که بیشترین امتیاز را شامل می‌شود استخراج کند تا از آن برای محاسبه PF استفاده نماییم.

۵-۱- الگوریتم محاسبه فاصله

در موقع انطباق یک عبارت در یک سند برا ساس موقعیت‌هایی که کلمات عبارت در سند دارند فاصله محاسبه می‌شود. در اینجا فاصله برابر کمترین جابجایی کلمات پرس‌وجو در سند برای ساختن عبارت پرس‌وجو می‌باشد. برای مثال اگر در متن ‘a d f c d b e’ عبارت ‘a b c’ را جستجو کنیم؛ در یک روش می‌توان a را ثابت در نظر گرفت و b و c را جابجا کرد که در این حالت تعداد جابجایی برای b چهار می‌باشد و برای c یک و درنتیجه مجموع جابجایی(فاصله) پنج می‌شود و اگر c را ثابت بگیریم مجموع جابجایی چهار می شود زیرا b به سه جابجایی نیاز دارد تا قبل از c قرار گیرد و a تنها به یک جابجایی نیاز دارد تا قبل از b قرار گیرد.
به‌عنوان‌مثال دیگر اگر پرس‌وجو a b c باشد و در سند مفروض a در مکان ۱۰۰ و b در مکان ۳۰۰ و c در مکان ۲۰۰ قرار گرفته باشد یعنی سند به‌صورت …a….c…b…. باشد منظور از میانه c می‌باشد. با استفاده از لم ۱ و ۲ و قضیه یک ثابت می‌شود اگر میانه ثابت با شد و کلمات دیگر جابجا شوند کمترین جابجایی صورت گرفته ا ست یعنی در مثال فوق باید c ثابت با شد و a و b جابجا شوند تا پرس‌وجو ساخته شود.
در ادامه نشان می‌دهیم که میانه موقعیت کلمات عبارت در متن بهترین مکان برای انتخاب محور جابجایی‌ها هست و باید تمام کلمات را در آنجا جمع نمود و فاصله را محاسبه کرد. الگوریتم ۱ روش محاسبه فاصله را نشان می‌دهد

Algorithm Distance
01: **Input** Query *q*, Text *T*
02: **Output** *Distance*
03: **Assumption**
04: *q* include $<t_1, t_2, \ldots, t_m>$
05: *T* include $<(t_1,p_1), (t_2,p_2), \ldots, (t_m, p_m)>$
06: **Begin Algorithm**
07: sort $<p_1, p_2, \ldots, p_m>$
08: Mid← midpoint of $<p_1, p_2, \ldots, p_m>$
09: **For** *i*=1 to m
10: //move (t_i, p_i) to *Mid+i-1*
11: *Distance* ← *Distance*+abs(*Mid+i-1-*p_i)
12: **Return** Distance

الگوریتم ۱ : الگوریتم محاسبه فاصله

برای اثبات درستی الگوریتم شکل ۱ فرض می‌کنیم که عبارت دارای m کلمه باشد و کلمات عبارت به‌صورت $t_1,t_2,\ldots,t_m$ می‌باشند. همچنین موقعیت کلمه t_i در سند p_i در نظر می‌گیریم(p_iها لزوماً مرتب شده نیستند). می‌توان ثابت کرد که این الگوریتم برای محاسبه فاصله بهینه است و کمترین مقدار جابجایی با فرض محور قرار دادن میانه p_iها حاصل خواهد شد. زیرا اگر نقطه‌ای غیر از میانه به‌عنوان محور فرض شود جابجایی بی شتری ن سبت به میانه نیاز است زیرا لااقل نیمی از کلمات باید بیشتر از فاصله‌شان تا میانه جابجا شوند و نیم دوم که به‌ظاهر جابجایی کمتری دارند درمجموع جابجایی بیشتری را باید داشته باشند.

لم ۱ : اگر تمام کلمات عبارت که در سند موجود ه ستند را جابجا کنیم تا در میانه‌ی p_iها متمرکز شوند موقعیت میانه در تعداد جابجایی‌ها تأثیرگذار نیست.

اثبات : ابتدا فرض می‌کنیم که m عددی زوج باشد. برای یافتن میانه باید کلمات موجود در عبارت را براساس موقعیت مرتب کرد. فرض می‌کنیم که موقعیت کلمات عبارت پس از مرتب‌سازی به‌صورت زیر درآید:

$$p_{i_1}, p_{i_2}, \ldots, p_{i_{\frac{m}{2}}}, p_{i_{\frac{m+1}{2}}}, \ldots, p_{i_m}$$

اگر موقعیت میانه x فرض شود جمع تعداد جابجایی کلمات قبل از x برابراست با :

$$d_{left} = \sum_{j=i_1}^{i_{\frac{m}{2}}} (x - p_j + j - 1) \qquad (۴)$$

که با توجه به این‌که j محل کلمه در عبارت را مشخص می‌کند $j-1$ جابجایی برای تطبیق کامل نیاز است. به‌طور مشابه جمع تعداد جابجایی کلمات در سمت راست x به صورت رابطه ۵ خواهد بود.

$$d_{right} = \sum_{j=i_{\frac{m+1}{2}}}^{i_m} (p_j - x - j + 1) \qquad (۵)$$

درنتیجه در محاسبه جمع رابطه ۴ و ۵ متغیر x حذف خواهد شد و رابطه ۶ نتیجه می‌شود که مستقل از x می‌باشد(همچنین مستقل از j).

$$d_m = \sum_{j=i_1}^{\frac{im}{2}}(-p_j) + \sum_{j=i_{\frac{m+1}{2}}}^{i_m}(p_j) \qquad (۶)$$

برای حالتی که m فرد باشد موقعیت میانه بر موقعیت کلمه وسط منطبق هست و درصورتی‌که کلمه واقع در میانه ثابت فرض شود و کلمات دیگر جابجا شوند تعداد جابجایی‌ها رابطه‌ای مشابه روابط ۴ و ۵ و ۶ پیدا خواهد کرد و مستقل از x خواهد بود.

لم ۲ : در تطبیق کلمات یک عبارت با کلمات موجود در متن کمترین جابجایی زمانی اتفاق می‌افتد که تمام کلمات عبارت که در سند موجود هستند را جابجا کنیم تا در میانه‌ی p_iها متمرکز شوند یا دقیقاً روی یکی از p_i ها.

اثبات : برهان خلف. اگر فرض کنیم در حالت بهینه نقطه‌ای غیر از میانه مثل نقطه y محلی مناسب برای تجمیع کلمات باشد و k تعداد کلماتی از عبارت است که قبل از موقعیت y در متن ظاهر شده است. بنابراین مجموع جابجایی‌ها برای تطبیق کامل به‌صورت رابطه ۷ خواهد بود.

$$d_y = \sum_{j=i_1}^{i_k}(y - p_j + j - 1) + \sum_{j=i_{k+1}}^{i_m}(p_j - y - j + 1)$$
$$\approx (2k-m)y + \sum_{j=i_1}^{i_k}(-p_j) + \sum_{j=i_{k+1}}^{i_m}(p_j) \qquad (۷)$$

دلیلی که رابطه ۷ تقریب زده شده است به خاطر حذف تعدادی از زها می‌باشد که با توجه به تعداد کم کلمات در عبارت قابل‌اغماض می‌باشد. تعداد کلمات پرس‌وجو در مقابل اغماض است.

اگر k>m/2 باشد ضریب 2k-m مثبت خواهد بود و کاهش y باعث کاهش d_y خواهد شد و بنابراین d_y بهینه نخواهد بود. البته کاهش y تا رسیدن به آخرین کلمه ماقبلش، d_y را کاهش می‌دهد و پس‌ازآن سیگماهای اول و دوم تغییر می‌کند. بنابراین در این حالت موقعیت y و p_{i_k} یکی خواهد شد.

به‌طور مشابه اگر k<m/2 باشد ضریب 2k-m منفی خواهد شد و افزایش y باعث کاهش d_y خواهد شد و بنابراین d_y بهینه نخواهد بود. البته افزایش y تا رسیدن به اولین کلمه مابعدش، d_y را کاهش می‌دهد و پس‌ازآن سیگماهای اول و دوم تغییر می‌کند. بنابراین در این حالت موقعیت y و $p_{i_{k+1}}$ یکی خواهد شد.

بنابراین در دو حالت y منطبق بر یکی از کلمات خواهد شد.

قضیه ۱ : برای تطبیق کلمات یک عبارت با کلمات موجود کمترین جابجایی زمانی اتفاق می‌افتد که تمام کلمات عبارت که در سند موجود هستند را جابجا کنیم تا در میانه‌ی p_iها متمرکز شوند.

اثبات : برهان خلف. فرض کنیم غیر از میانه نقطه‌ای مثل y وجود داشته باشد که در صورت تجمیع کلمات در آن نقطه جابجایی کمتری نسبت به میانه داشته باشیم. طبق لم ۲ محل y باید یا میانه باشد یعنی d_m در لم ۱ یا d_y در لم ۲ باید باشد که y بر یکی از مقادیر p_iها منطبق هست. در ادامه نشان می‌دهیم که d_y از d_m همواره بزرگ‌تر است.

ابتدا فرض می‌گیریم که k<m/2 باشد و بنابراین ضریب y در رابطه ۷ منفی خواهد بود. در این حالت رابطه ۷را می‌توان به‌صورت رابطه زیر تغییر داد:

$$d_y = (2k-m)y + \sum_{j=i_1}^{i_k}(-p_j) + \sum_{j=i_{k+1}}^{i_m}(p_j)$$
$$= (2k-m)y + \sum_{j=i_1}^{\frac{im}{2}}(-p_j) - \sum_{j=i_{k+1}}^{\frac{im}{2}}(-p_j) + \sum_{j=i_{k+1}}^{\frac{im}{2}}(p_j) + \sum_{j=i_{\frac{m+1}{2}}}^{i_m}(p_j)$$
$$= d_m + (2k-m)y + 2\sum_{j=i_{k+1}}^{\frac{im}{2}}(p_j) \qquad (۸)$$

در رابطه ۸ سیگما اول به دو سیگما تبدیل شده است. یک‌بار تا میانه حساب شده است و یک‌بار بخش اضافه کم گردیده است و به‌طور مشابه سیگما دوم نیز به دو بخش تقسیم شده است بخش قبل از میانه و بخش بعد از میانه. همچنین با توجه به این‌که در سیگمای نهایی تمام p_jها بزرگ‌تر از y هستند پس حد پایینی d_y به‌صورت زیر نتیجه خواهد شد:

$$d_y > d_m + (2k-m)y + 2\left(\frac{m}{2} - (k+1) + 1\right)y$$
$$\Rightarrow d_y > d_m \qquad (۹)$$

و این خلاف فرض می‌باشد. برای حالتی که k<m/2 باشد هم به همین تناقض می‌رسیم. پس حالت بهینه زمانی است که رابطه ۷ مستقل از y باشد و یا به عبارتی y باید میانه باشد. که در لم ۱ به این موضوع رسیدیم.

۲-۵- الگوریتم محاسبه PF

یکی از الگوریتم‌هایی که می‌تواند PF را محاسبه کند الگوریتم جستجوی کامل می‌باشد. در این روش تمام ترکیب‌هایی که می‌توان با استفاده از آن عبارت را ایجاد کرد استخراج می‌شود و شکلی که مجموع امتیاز آن بالاتر است انتخاب می‌شود.

درروش پیشنهادی ابتدا با استفاده از "and query" اسنادی که تمام کلمات عبارت را دارا هستند را استخراج می‌شوند. سپس با استفاده از نمایه معکوس به ازای تمام کلمات عبارت تمام جایگشت‌های ممکن از انطباق‌های عبارت بررسی می‌گردد و سپس برای هر جایگشت جمع فاصله محاسبه می شود. پیدا کردن جایگشتی که جمع بیشینه‌ی فاصله را تولید کند هدف اصلی است.

الگوریتم ۲ روش محاسبه PF را نشان می‌دهد. در این الگوریتم متغیر Max برای نگهداری بیشترین مقدار مجموع فاصله مورداستفاده قرارگرفته است. کار اصلی در این الگوریتم تولید تمام ترکیبات عبارت مورد جستجو در متن می‌باشد.

Algorithm PF

01: **Input** Query q,Document d
02: **Output** PF
03: **Assumption**
04: q is $<t_1, t_2,..., t_m>$
05: d includes $<t_1,(P_{11},P_{12},...)>$, $<t_2,(P_{21},P_{22},...)>$,..., $<t_m,(P_{m1},P_{m2},...)>$
06: **Begin Algorithm**
07: $Max \leftarrow 0$
08: **Repeat**
09: Make a new permutation
10: $S \leftarrow$ Calulate Distance of all phrases
11: If $S > Max$ Then
12: $Max \leftarrow S$
13: **Until** no new permutation
14: **Return** Max

الگوریتم ۲: الگوریتم محاسبه PF برای یک عبارت

در الگوریتم ۲ از متغیر P_{ij} برای مشخص کردن j-امین موقعیت کلمه i-ام عبارت استفاده شده است. خط ۹ الگوریتم وظیفه‌ی تولید جایگشت‌های مختلف تطابق کلمات عبارت با کلمات معادل در سند را بر عهده دارد. مثلاً اولین جایگشت می‌تواند به‌صورت زیر انتخاب شود:

Phrase1: $<t_1,P_{11}>,<t_2,P_{21}>,\ldots,<t_n,P_{n1}>$
Phrase2: $<t_1,P_{12}>,<t_2,P_{22}>,\ldots,<t_n,P_{n2}>$
.
.
.

برای جایگشت بعدی می‌توان موقعیت t_1 در Phrase1 را تغییر داد و P_{12} در نظر گرفت و به‌طور مشابه موقعیت t_1 در Phrase2 را P_{11} قرار داد و یک جایگشت جدید به دست آورد.

الگوریتم در خط ۱۰ برای تک‌تک عبارات استخراج‌شده در هر جایگشت مقدار حداقل جابجایی را محاسبه و جمع امتیازهای حاصل را در S ذخیره می‌کند.

این الگوریتم با این که در بدترین حالت به دلیل محدود بودن QMT خطی است ولی به خاطر اهمیت ضریب خطی در جستجو انبوه باید بهینه شود. برای بهینه کردن این الگوریتم می‌توان از حرص کردن براساس فاصله کلمات استفاده کرد. یکی از روش‌ها برای حرص کردن استفاده از همین مفهوم فاصله است. مشخص است که از یک فاصله‌ای بیشتر ارتباط بین کلمات تقریباً قطع می‌شود و با این فرض می‌توان فاصله کلمات را محدود کرد و بخشی از درخت را حرص نمود. مثلاً اگر دو کلمه در فاصله ۱۰۰۰ از یکدیگر قرار گرفته باشند امتیاز قابل‌توجهی را تولید نمی‌کنند و بنابراین می‌توان بررسی آن حالت را ادامه نداد.

روش دیگر برای بهینه کردن الگوریتم فوق پردازش آماری پرس‌وجوها و استخراج مرز تصمیم‌گیری براساس tf و idf هست. یعنی برای idf های پایین tf های کم پردازش نشود.

البته در لوسین هم مفهومی به نام Slop تعریف شده است که مشابه همین وظیفه را بر عهده دارد. در لوسین طول پنجره حداکثر به‌اندازه Slop می‌باشد.

۳-۵- محاسبه IDF

در موقع جستجوی یک عبارت پارامتر IDF نیز باید به شکل مناسبی تعریف شود تا به‌درستی میزان اهمیت عبارت را مشخص کند. IDF پارامتری است که با لگاریتم وارون DF رابطه مستقیمی دارد و DF نشان‌گر تعداد اسنادی است که شامل عبارت مورد نظر می‌باشد. هر چه تعداد اسناد کمتر باشد اهمیت عبارت بیشتر است. یکی از مشهورترین تعاریف برای IDF به‌صورت رابطه زیر می‌باشد.

$$idf_t = log\frac{N}{1+df_t} \qquad (۱۰)$$

که در آن N تعداد کل اسناد می‌باشد df_t تعداد سندی را نشان می‌دهد که شامل کلمه t هستند.

اگر بخواهیم به شکل مشابه IDF را برای یک عبارت تعریف کنیم باید DF را برای یک عبارت محاسبه نماییم. برای محاسبه DF یک عبارت می‌توان اسناد را به دو دسته تقسیم کرد:

- اسنادی که PF بزرگ‌تر یا مساوی یک دارند.
- اسنادی که PF کوچک‌تر از یک دارند.

واضح است که در شمارش اسناد شامل عبارت، اسنادی که PF بزرگ‌تر یا مساوی یک دارند باید شمارش شوند.

اما اسنادی که PF کمتر از یک دارند باید شمارش شوند ولی با یک ضریب کمتر از یک. رابطه ۱۱ را برای این منظور به‌صورت زیر تعریف می‌کنیم و در رابطه ۱۲ مقدار df یک عبارت محاسبه می‌شود.

$$F(d,q) = \begin{cases} 1 & PF \geq 1 \\ PF & PF < 1 \end{cases} \qquad (۱۱)$$

$$df_q = \sum_d F(d,q) \qquad (۱۲)$$

حال با استفاده از df به‌سادگی می‌توان از رابطه ۱۰ مقدار idf را محاسبه است. الگوریتم ۳ مراحل محاسبه‌ی IDF را نشان می‌دهد.

Algorithm IDF

01: **Input** Query q
02: **Output** *IDF*
03: **Assumption**
04: q include $<t_1, t_2, \ldots, t_m>$
05: N is total document in corpus
06: **Begin Algorithm**
07: $DF \leftarrow 0$
08: **For** every document d_i
09: $PF \leftarrow$ calculate PF(q,d_i)
10: **If** PF>=1 **Then**
11: $DF \leftarrow DF+1$
12: Else
13: $DF \leftarrow DF+PF$
14: $IDF \leftarrow \text{Log}(N/(1+DF))$
15: **Return** *IDF*

الگوریتم ۳: الگوریتم محاسبه IDF برای یک عبارت

با یک تغییر ساده می‌توان الگوریتم ۳ را در الگوریتم ۲ یعنی الگوریتم محاسبه PF ادغام کرد. یعنی می‌توان در موقع محاسبه PF برای کلیه اسناد مقدار IDF عبارت را نیز محاسبه کرد. البته مسئله کارایی این الگوریتم‌ها نسبت به روش‌های سنتی برای استخراج TF و IDF بسیار متفاوت خواهد بود.

در روش‌های سنتی TF هر کلمه برای هر سند به‌سادگی در دسترس می‌باشد و همچنین IDF پایگاه برای هر کلمه در زمان ایندکس می‌تواند محاسبه و ذخیره شود و در موقع جستجو بدون هیچ پردازشی در اختیار بخش رتبه‌بندی قرار داده شود. اما در این روش این دو پارامتر باید در موقع جستجو با یک پردازش محاسبه گردد که نسبت به روش سنتی ازنظر زمانی کندتر خواهد بود.

۴-۵- پیچیدگی زمانی الگوریتم

درصورتی‌که بخواهیم یک عبارت را تطبیق دهیم طبق قضیه۱ باید میانه را بیابیم. اگر برای یافتن میانه از مرتب سازی سریع استفاده شود پیچیدگی زمانی O(nlogn) است که n نشان‌دهنده تعداد مقادیری می‌باشد که می‌خواهیم میانه آن‌ها را حساب کنیم. البته الگوریتم های سریع‌تری برای یافتن میانه نیز وجود دارد که پیچیدگی زمانی آن نسبت به n خطی است[۳۱] ولی خواهیم دید که تأثیری روی پیچیدگی محاسبه PF ندارند.

همان‌طور که در بخش ۲ دیدیم، موتورهای جستجو یک حداکثر تعداد کلمه در پرس‌وجو را مشخص می‌کنند که ما آن را با مقدار ثابت QMT تعریف کردیم. پس پیچیدگی زمانی پیدا کردن فاصله به‌صورت O(QMT*log(QMT)) خواهد بود و با توجه به ثابت بودن QMT این پیچیدگی به‌صورت O(1) می‌باشد.

پیچیدگی زمانی محاسبه PF طبق الگوریتم ارائه‌شده، رابطه مستقیمی با تعداد جایگشت‌های عبارت در متن دارد. تعداد جایگشت یک پرس‌وجو به‌صورت $t_1t_2\ldots t_m$ به‌صورت رابطه ۱۳ خواهد بود.

$$T(m) = \prod_{i=1}^{m} tf(t_i) \qquad (۱۳)$$

اما برای پیچیدگی زمانی محاسبه‌ی PF باید سه حالت بهترین، بدترین و حالت متوسط را بدست آوریم. واضح است که بهترین حالت زمانی است که tf تمام کلمات عبارت یک باشد و درنتیجه پیچیدگی زمانی در بهترین حالت O(1) می‌باشد.

همچنین بدترین حالت زمانی اتفاق می‌افتد که متن به‌طور مطلق شامل فقط کلمات عبارت باشد. لم ۳ و قضایای ۲ و ۳ برای محاسبه‌ی پیچیدگی زمانی محاسبه‌ی PF در بدترین حالت و حالت متوسط ارائه می‌شود.

لم ۳ : اگر $n = \sum_{i=1}^{m} a_i$ ثابت باشد آنگاه $\prod_{i=1}^{m} a_i$ زمانی بیشینه هست که a_iها مساوی باشند.

اثبات این لم با استفاده از ضرایب لاگرانژ به‌سادگی امکان‌پذیر است[۳۲].

قضیه ۲ : پیچیدگی زمانی PF در بدترین حالت $O(n^{QMT})$ می‌باشد که n تعداد کلمات سند می‌باشد.

اثبات : مشخص است بدترین حالت زمانی اتفاق می‌افتد که تک‌تک کلمات سند در کلمات عبارت باشد پس $n = \sum_{i=1}^{m} tf(t_i)$ می‌باشد. طبق لم ۳ برای این‌که حاصل ضرب tf ها بیشینه باشد $tf(t_i)$ها مساوی باشند. پس در بدترین حالت داریم:

$$tf(t_i) = \frac{n}{m} \qquad (۱۴)$$

با توجه به این‌که حداکثر مقدار m برابر با QMT می‌باشد بنابراین پیچیدگی زمانی محاسبه PF در بدترین حالت $O(n^{QMT})$ می‌باشد.

قضیه ۳ : پیچیدگی زمانی PF در حالت متوسط در O(1) می‌باشد.

اثبات: برای حالت متوسط باید امید ریاضی tf ها را پیدا کرد. می‌توان اثبات کرد که توزیع هر کلمه در هر سند دارای توزیع پواسون می‌باشد[۳۳] و بدیهی است که تعداد تکرار هر کلمه در هر سند یک توزیع دوجمله‌ای است. این موضوع با استفاده از دیتاست‌های TREC2003 و TREC2004[۳۰] نیز قابل‌بررسی می‌باشد. بنابراین داریم $E(tf) = \mu$ که μ میانگین توزیع پوآسون می‌باشد.

اگر فرض کنیم tf ها از هم مستقل هستند امید ریاضی حاصل‌ضرب را می‌توان به حاصل‌ضرب امید ریاضی‌ها تبدیل کرد[۳۴]. یعنی حد بالایی تعداد جایگشت‌ها در حالت متوسط را می‌توان با استفاده از رابطه ۱۴ به‌صورت زیر بدست آورد:

$$E\left[\prod_{i=1}^{m} tf(t_i)\right] = \prod_{i=1}^{m} E(tf(t_i)) = \mu^m \leq \mu^{MTQ} \qquad (۱۵)$$

و با توجه به ثابت بودن μ و QMT پیچیدگی زمانی محا سبه PF در حالت میانگین O(1) خواهد بود.

۶- مقایسه الگوریتم پیشنهادی با لوسین

در این بخش مقایسه‌ای بین کارایی الگوریتم استفاده‌شده در لوسین و الگوریتم پیشنهادی در جدول ۱ انجام گردیده است.

جدول ۱: مقایسه پیچیدگی زمانی لوسین و الگوریتم پیشنهادی

پارامتر	لوسین	الگوریتم پیشنهادی
فاصله	O(1)	O(1)
محاسبه PF در بدترین حالت	O(n)	$O(n^{QMT})$
محاسبه PF در حالت	O(1)	O(1)
محاسبه IDF در بدترین	O(1)	$O(n^{QMT})$
محاسبه IDF در حالت	O(1)	O(1)

همان‌طور که جدول ۱ نشان می‌دهد ازنظر سرعت اجرا، لوسین کارایی بهتری را دارد ولی در حالت متو سط دو الگوریتم پیچیدگی یکسانی دارند.

اما ازنظر صحت عملکرد رفتار لوسین غیرقطعی است و این موضوع در کد لوسین نیز ذکر شده است. در خط ۹۰ ماژول SloppyPhraseScorer.java[۳۵] در نسخه‌ی ۴,۹ لوسین (آخرین نسخه‌ی لوسین در زمان نگارش این مقاله) این مسئله به‌صورت زیر عنوان شده است:

"به خاطر کارایی امکان دارد در شرایطی تمام ترکیب‌های ممکن بررسی نشود و ما همیشه تعداد محدودی از ترکیب‌ها را بررسی کنیم. این روش باعث می‌شود که بازگشت به عقب نداشته باشیم و سریع‌تر امتیاز را محاسبه کنیم ولی نتایج همیشه صحیح نیست. برای مثال اگر در متن 'a b c b a' یک‌بار عبارت 'a b c' و یک‌بار عبارت 'c b a' را جستجو کنیم نتایج متفاوت خواهد بود درحالی‌که باید یکسان باشد. این مشکل شاید در نسخه‌های بعدی حل شود (ولی فعلاً به خاطر مسائل کارایی این کار انجام نمی‌شود)."

جدول ۲ مقایسه‌ای بین چند پرس‌وجو در چند سند با استفاده از لوسین و الگوریتم پیشنهادی را نشان می‌دهد و به‌وضوح اشکالات مشخص است.

جدول ۲: مقایسه بین PF ها در لوسین و الگوریتم پیشنهادی

عبارت	متن	PF لوسین	PF الگوریتم پیشنهادی
'a b'	'b a b'	1	1
'a b'	'a b a'	1.33	1
'a b'	'a a b b'	1	1.33
'a b c'	'b c a'	0.25	0.25
'a b c'	'a b c a'	1.25	1
'a b c'	'a b c b a'	1.2	1

۷- نتایج ارزیابی

به‌منظور ارزیابی الگوریتم پیشنهادی از داده تست موتور جستجوی پارسی جو[۳۶] که بخشی از داده واقعی موتور مذکور می‌باشد، استفاده شده است. این داده تست مشتمل بر حدود چهار صد هزار سند صفحه وب و حدود سی پرس‌وجوی ارزیابی‌شده می‌باشد. همچنین از معیار دقت[۳۷] در مکان n-ام (p@n) و میانگین متوسط دقت (MAP[۳۸]) برای مقایسه بین الگوریتم پیشنهادی با لوسین و با حالتی که پرس‌وجو به صورت منطقی جستجو شود، استفاده شده است.

معیار دقت در مکان n-ام یا P@n نشان‌دهنده‌ی نسبت تعداد اسناد مرتبط در n سند اول رتبه‌بندی نهایی برای هر پرس‌وجو به n می‌باشد. فرمول P@n در معادله‌ی زیر نشان داده شده است:

$$P@n = \frac{1}{n}\sum_{j=1}^{n} r_j \qquad (۱۶)$$

که r_j نشان‌دهنده‌ی مرتبط بودن سند j-ام در رتبه‌بندی نهایی است.

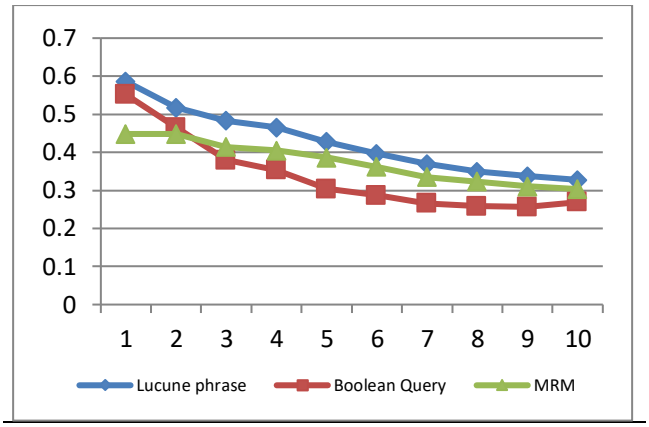
همچنین میانگین متوسط دقت یا MAP برای هر پرس‌وجو به‌عنوان میانگین مقادیر P@n-ها برای همه‌ی اسناد مرتبط تعریف می‌شود.

$$AP = \frac{1}{|D_+|}\sum_{j=1}^{N} r_j \times P@j \qquad (۱۷)$$

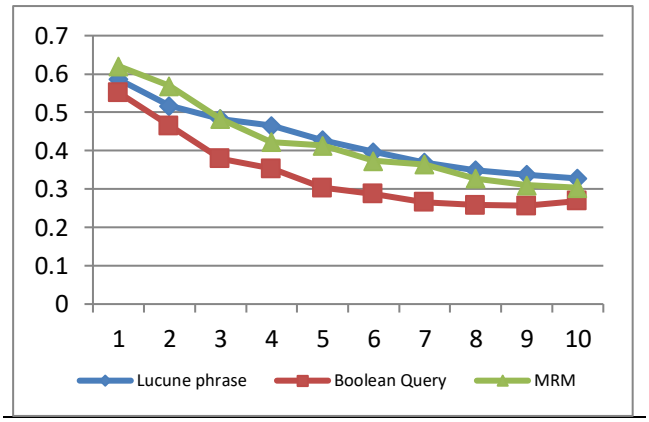
N نشان‌دهنده‌ی تعداد کل اسناد، D_+ نشان‌دهنده‌ی تعداد اسناد مرتبط و r_j نشان‌دهنده‌ی مرتبط بودن سند j-ام است؛ متوسط میانگین دقت (MAP) به‌عنوان میانگین مقادیر AP همه‌ی پرس‌وجوهای ارائه شده است.

شکل‌های ۴ و ۵ و ۶ مقایسه‌ی بین سه الگوریتم جستجو را نشان می‌دهد. در این شکل‌ها منظور از جستجوی منطقی ترکیب نتایج هر کلمه به‌طور مجزا می‌باشد که با عملگرهای منطقی یعنی and و or باهم ترکیب شده باشند. برای الگوریتم حداقل جابجایی نیز به‌اختصار MRM[۳۹] انتخاب شده است. در این سه شکل از سه تابع متفاوت برای s(d) استفاده گردید است. در شکل‌های ۵ و ۶،

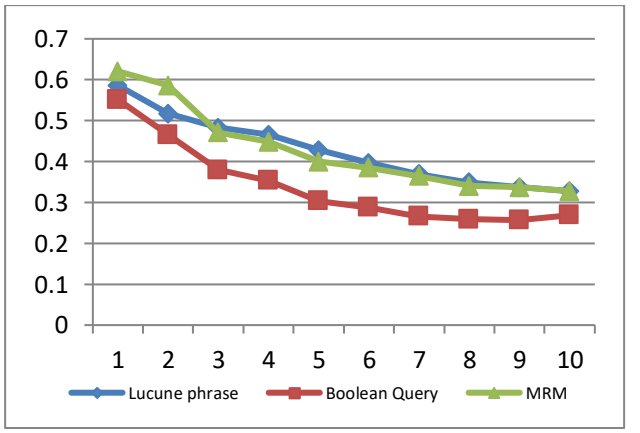
الگوریتم پیشنهادی p@1 و p@2 بیشتری دارد و مابقی p@n ها نزدیک به هم می‌باشد. اما در شکل ۴ که از تابع $s(d)=\frac{1}{\sqrt{d}}$ استفاده شده است، نتایج الگوریتم پیشنهادی ضعیف‌تر می‌باشد و دلیل آن تأثیر فاصله روی نتایج می‌باشد. بنابراین انتخاب تابع s(d) اهمیت دارد. یعنی با جذر گرفتن از فاصله و سپس معکوس کردن عملاً تأثیر فاصله کمتر می‌شود.



شکل ۴: مقایسه بین الگوریتم پیشنهادی(MRM) با الگوریتم لوسین و جستجوی منطقی با استفاده از معیار P@n در حالت $s(d)=\frac{1}{\sqrt{d}}$

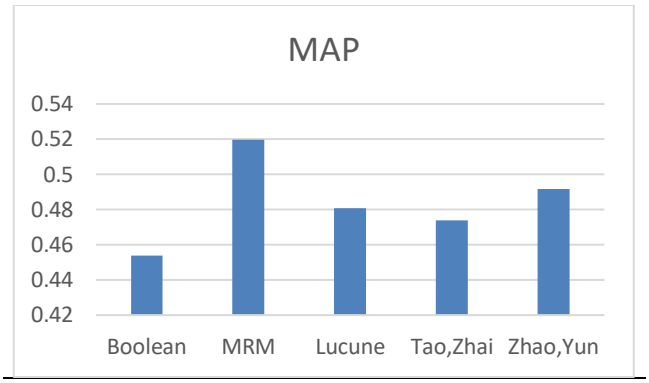


شکل ۵: مقایسه بین الگوریتم پیشنهادی با الگوریتم لوسین و جستجوی منطقی با استفاده از معیار P@n در حالت $s(d)=1/d^2$



شکل ۶: مقایسه بین الگوریتم پیشنهادی با الگوریتم لوسین و جستجوی منطقی با استفاده از معیار P@n در حالت s(d)=1/d

با توجه به نزدیک بودن نتایج p@n از معیار MAP نیز استفاده شده است. شکل ۷ نشان می‌دهد که در حالت متوسط با استفاده از معیار MAP نیز دقت الگوریتم پیشنهادی بیشتر از دو الگوریتم دیگر است.



شکل ۷: مقایسه بین الگوریتم پیشنهادی و چند الگوریتم دیگر با استفاده از معیار MAP در حالت s(d)=1/d [26]

۸- نتیجه‌گیری و کارهای آینده

در این مقاله یکی از موضوعات موتورهای جستجو یعنی جستجو عبارت بدون علامت نقل‌قول بررسی گردید. در جستجوی یک عبارت در یک متن ممکن است کلمات عبارت در جاهای مختلف متن موجود باشند و درنتیجه مفهوم فاصله تعریف شد که نشان‌دهنده‌ی حداقل جابجایی برای تطبیق عبارت بود و با استفاده از فاصله برای محاسبه PF و IDF الگوریتم‌هایی ارائه شد و بین الگوریتم پیشنهادی و الگوریتم استفاده‌شده در لوسین مقایسه‌ای ازنظر سرعت و صحت انجام شد.

به‌عنوان چشم‌انداز آینده می‌توان روی افزایش سرعت محاسبه PF و IDF الگوریتم‌های سریع‌تری ارائه داد. همچنین موازی سازی نیز می‌تواند یک روش برای افزایش سرعت محاسبه‌ی PF و IDF قابل‌بررسی است. برای بخشی از یک عبارت در سند هم باید امتیازی لحاظ شود که شاید بتوان با روش‌های آموزش این امتیاز محاسبه گردد.

مراجع

[1] A. Z. Bidoki, “Effective Web Ranking and Crawling(in persian),” University of Tehran, 2009.

[2] R. Baeza-Yates, B. Ribeiro-Neto, and others, *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[3] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Inf. Process. Manag.*, vol. 24, no. 5, pp. 513–523, 1988.

[4] S. E. Robertson, *Overview of the Okapi projects*, vol. 53, no. 1. MCB UP Ltd, 1997, pp. 3–7.

[5] Y. Zhang and A. Moffat, “Some Observations on User Search Behaviour.,” *Austr. J. Intell. Inf. Process. Syst.*, vol. 9, no. 2, pp. 1–

8, 2006.

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

[7] H. E. Williams, J. Zobel, and D. Bahle, "Fast phrase querying with combined indexes," *ACM Trans. Inf. Syst.*, vol. 22, no. 4, pp. 573–594, 2004.

[8] A. Doucet and H. Ahonen-Myka, "An efficient any language approach for the integration of phrases in document retrieval," *Lang. Resour. Eval.*, vol. 44, no. 1–2, pp. 159–180, 2010.

[9] I. H. Witten, A. Moffat, and T. C. Bell, *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[10] D. Bahle, "Efficient Phrase Querying," *Sch. Comput. Sci. Inf. Technol. R. Melb. Inst. Technol.*, 2003.

[11] A. Fellinghaug, "Phrase searching in text indexes," no. June, p. 137, 2008.

[12] C. J. van Rijsbergen, "A theoretical basis for the use of co-occurrence data in information retrieval," *J. Doc.*, vol. 33, no. 2, pp. 106–119, 1977.

[13] R. Nallapati and J. Allan, "Capturing term dependencies using a language model based on sentence trees," in *Proceedings of the eleventh international conference on Information and knowledge management*, 2002, pp. 383–390.

[14] E. M. Keen, "The use of term position devices in ranked output experiments," *J. Doc.*, vol. 47, no. 1, pp. 1–22, 1991.

[15] W. B. Croft, H. R. Turtle, and D. D. Lewis, "The use of phrases and structured queries in information retrieval," in *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, 1991, pp. 32–45.

[16] D. Metzler and W. B. Croft, "A Markov random field model for term dependencies," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 472–479.

[17] E. K. F. Dang, R. W. P. Luk, and J. Allan, "A context-dependent relevance model," *J. Assoc. Inf. Sci. Technol.*, 2015.

[18] F. Song and W. B. Croft, "A general language model for information retrieval," in *Proceedings of the eighth international conference on Information and knowledge management*, 1999, pp. 316–321.

[19] J. Gao, J.-Y. Nie, G. Wu, and G. Cao, "Dependence language model for information retrieval," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 170–177.

[20] B. He, J. X. Huang, and X. Zhou, "Modeling term proximity for probabilistic information retrieval models," *Inf. Sci. (Ny).*, vol. 181, no. 14, pp. 3017–3031, 2011.

[21] Y. Rasolofo and J. Savoy, *Term proximity scoring for keyword-based retrieval systems*. Springer, 2003.

[22] C. Eickhoff, A. P. de Vries, and T. Hofmann, "Modelling Term Dependence with Copulas," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 783–786.

[23] S. Büttcher, C. L. A. Clarke, and B. Lushman, "Term proximity scoring for ad-hoc retrieval on very large text collections," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 621–622.

[24] T. Tao and C. Zhai, "An exploration of proximity measures in information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 295–302.

[25] J. Zhao and Y. Yun, "A proximity language model for information retrieval," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 291–298.

[26] J. B. P. Vuurens and A. P. de Vries, "Distance matters! Cumulative proximity expansions for ranking documents," *Inf. Retr. Boston.*, vol. 17, no. 4, pp. 380–406, 2014.

[27] J. Miao, J. X. Huang, and Z. Ye, "Proximity-based rocchio's model for pseudo relevance," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 2012, pp. 535–544.

[28] C. L. A. Clarke, G. V Cormack, and E. A. Tudhope, "Relevance ranking for one to three term queries," *Inf. Process. Manag.*, vol. 36, no. 2, pp. 291–311, 2000.

[29] J. Klekota, F. P. Roth, and S. L. Schreiber, "Query Chem: a Google-powered web search combining text and chemical structures," *Bioinformatics*, vol. 22, no. 13, pp. 1670–1673, 2006.

[30] H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson, "Microsoft Cambridge at TREC 13: Web and Hard Tracks.," in *TREC*, 2004, vol. 4, p. 1.

[31] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," *J. Comput. Syst. Sci.*, vol. 7, no. 4, pp. 448–461, 1973.

[32] R. Courant, *Differential and integral calculus*, vol. 2. John Wiley & Sons, 2011.

[33] S. E. Robertson and S. Walker, "Some for Simple Effective Approximations to the 2 – Poisson Model Probabilistic Weighted Retrieval The," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994, vol. 1994, no. 1, pp. 232–241.

[34] R. Duda O., P. Hart E., and D. Stork G., *Pattern Classification*. 2000.

زیرنویس‌ها:

1 Phrase
2 Inverted Index
3 Term Frequency
4 Inverse Document Frequency
5 Phrase Frequency
6 Rijsbergen
7 Maximum Spanning Tree
8 Nallapati
9 Allan
10 Keen
11 Croft
12 Document Frequency
13 Metzler
14 Dang
15 Song
16 Unigram
17 Bigram
18 Gao

[19] Dependence Language Model
[20] Rasolofo
[21] He
[22] n-grams
[23] Büttcher
[24] Posting List
[25] Tao
[26] Zhao
[27] Yun
[28] Proximity Language Model
[29] Query Expansion
[30] Query Terms Maximum
[31] http://lucene.apache.org/
[32] Sloppy
[33] https://lucene.apache.org/core/5_3_1/core/org/apache/lucene/search/PhraseQuery.html
[34] Anchor Text
[35] https://lucene.apache.org/core/5_3_1/core/org/apache/lucene/search/PhraseQuery.html
[36] http://www.parsijoo.ir
[37] Percision
[38] Mean Average Percision
[39] Minimum Relocation Model