یک الگوریتم سریع و کارآمد برای استخراج مجموعه عناصر با سودمندی بالا با استفاده از مدل نرخ فراموشی

فاطمه جهانی[1*]، عادل قاضی خانی[2]

۱- دانشجوی کارشناسی ارشد، دانشگاه امام رضا (ع) ، ایران،
fatemehjahani1367@gmail.com

۲- استادیار ، عضو هیئت علمی دانشگاه امام رضا (ع) ، ایران،
adel.ghazi@gmail.com

چکیده

استخراج مجموعه عناصر با سودمندی بالا بعنوان یک کار مهم داده کاوی معرفی شده و محاسبات آن از لحاظ زمان اجرا و مصرف حافظه همچنان پرهزینه است. علاوه بر این، از آنجایی که آنها فرآیندهای استخراج را بدون در نظر گرفتن زمان ورود تراکنشها انجام می‌دهند، تحقق نیازهای کاربران با استفاده از این روش زمانی که می‌خواهند فقط اطلاعات به روز مربوط به داده‌های جریانی را نگه دارند، مشکل به نظر می‌رسد. در این مقاله، الگوریتمی مبتنی بر درخت برای استخراج مجموعه عناصر اخیر با سودمندی بالا پیشنهاد شده است. در این الگوریتم از مدل نرخ فراموشی برای تاکید بر اهمیت عناصر داده‌های اخیر استفاده می‌کند. همچنین براساس این مدل، سودمندی تراکنش‌ها را براساس زمان ورودشان به منظور تعیین وزن‌های بیشتر به داده‌های اخیر در مقایسه با داده‌های قدیمی کاهش می‌دهد. نتایج تجربی بر روی مجموعه داده‌های واقعی نشان داد که روش پیشنهادی، نه تنها اطلاعات مهم مجموعه عناصر اخیر با سودمندی بالا را فراهم می‌سازد بلکه نیاز به منابع محاسباتی کمتری مانند زمان اجرا و حافظه مصرفی را فراهم می‌سازد.

کلمات کلیدی: استخراج الگو، استخراج مجموعه عناصر، استخراج با سودمندی بالا، مدل نرخ فراموشی

۱–مقدمه

استخراج مجموعه عناصر پرتکرار[1] یک کار داده کاوی محبوب است [۱]. با توجه به پایگاه داده تراکنشها، استخراج مجموعه عناصر پرتکرار شامل کشف مجموعه عناصر پرتکرار می‌باشد. بعنوان مثال، گروهی از عناصر که اغلب مکرراً در پایگاه داده تراکنشهای مشتری ظاهر می‌شوند [۱-۴]. از محدودیتهای اساسی استخراج مجموعه عناصر پرتکرار این است که فرض می‌کند هر عنصر نمی‌تواند بیش از یکبار در هر تراکنش ظاهر شود و همه ی عناصر دارای اهمیت یکسان هستند (وزن، سود واحد یا ارزش). این دو فرضیه اغلب در کاربردهای واقعی صدق نمی‌کنند. بعنوان مثال، یک پایگاه داده از تراکنش‌های مشتری را در نظر بگیرید. متداول است که یک مشتری چندین واحد از یک محصول مشابه را خریداری کند (بعنوان مثال، یک مشتری ممکن است چند بطری آب بخرد) و تمام عناصر دارای سود واحد یکسانی نیستند (بعنوان مثال، فروش الماس سود بیشتری از فروش یک بطری آب دارد). الگوریتم‌های استخراج مجموعه عناصر پرتکرار سنتی، اطلاعات مربوط به مقادیر خرید اقلام و سود واحد اقلام را در نظر نمی‌گیرند. بنابراین، الگوریتم‌ها این اطلاعات را دور انداخته و تنها مجموعه عناصر پرتکرار بجای یافتن آنهایی که سود بالایی دارند، پیدا می‌کنند. در نتیجه، بسیاری از مجموعه عناصر پرتکرار ناخوشایند که سود کم تولید می‌کنند ممکن است کشف شوند و بسیاری از مجموعه عناصر نادر که سود بالایی تولید می‌کنند ممکن است از دست رفته باشند[۵].

1 . Frequent Itemset Mining

برای مقابله با این مشکل، مسئله استخراج مجموعه عناصر با سودمندی بالا [1] تعریف شده است[۶-۱۵]. در مقابل با استخراج مجموعه عناصر پرتکرار [۱-۳]، استخراج مجموعه عناصر با سودمندی بالا موردی را در نظر می‌گیرد که در آن عناصر می‌توانند بیش از یکبار در هر تراکنش ظاهر شوند و در جایی که هر عنصر یک وزن دارد (بعنوان مثال، سود واحد). هدف از استخراج مجموعه عناصر با سودمندی بالا ، کشف مجموعه عناصر دارای سودمندی بالا است [۵].

اخیرا، پردازش داده‌های جریان حساس به زمان مساله مهمی بوده است چرا که داده‌های اخیر می‌توانند برای کشف اطلاعات الگوهای مهم در مقایسه با داده‌های گذشته مفید باشند. بعنوان مثال، در یک پایگاه داده پزشکی، سوابق اخیر بیماران برای بررسی روند فعلی پزشکی در مقایسه با سوابق قدیمی مهم‌تر است[۱۶]. در این مقاله، استفاده از مدل نرخ فراموشی در الگوریتم EFIM[2] [۵] برای استخراج مجموعه عناصر اخیر با سودمندی بالا پیشنهاد شده است. در این الگوریتم از مدل نرخ فراموشی[3] به منظور یافتن مجموعه عناصر اخیر و مفید با سودمندی بالا بکار می‌بریم. براساس مدل نرخ فراموشی، اهمیت داده‌های قدیمی‌تر به طور مداوم با گذر زمان کاهش می‌یابد. در نتیجه اهمیت داده‌ها می‌تواند براساس زمان ورود آن‌ها متفاوت باشد.

بقیه این مقاله به شرح زیر سازماندهی شده است: بخش ۲، ۳، ۴، ۵ و ۶ به ترتیب پیشینه تحقیق، روش پیشنهادی، ارزیابی تجربی، نتیجه گیری و پیشنهادها ارائه شده است.

۲-پیشینه ی تحقیق

در روشهای استخراج مجموعه عناصر با سودمندی بالا، مقدار و اهمیت اطلاعات هر عنصر در پایگاه داده را برای یافتن مجموعه عناصر معنادار در مقایسه با روشهای استخراج مجموعه عناصر پرتکرار مورد استفاده قرار می‌دهند. همچنین از مدل نرخ فراموشی در روشهای استخراج مجموعه عناصر با سودمندی بالا برای بدست آوردن اطلاعات به روز و تشخیص اطلاعات اخیر و قدیمی استفاده می‌شود.

در این بخش، ما به بررسی مقالات پیشین در مورد روش های استخراج مجموعه عناصر با سودمندی بالا و استفاده از مدل نرخ فراموشی در روشهای استخراج مجموعه عناصر با سودمندی بالا می‌پردازیم.

۲-۱-استخراج مجموعه عناصر با سودمندی بالا

- چان[4] و همکاران، الگوریتم‌های استخراج قوانین انجمنی سنتی تنها تعداد زیادی از قوانین بسیار مکرر را تولید می‌کنند، اما این قوانین پاسخ های مفیدی را برای آنچه که قوانین با سودمندی بالا ارائه می‌دهند، ندارند. ابتدا مسئله استخراج مجموعه عناصر با سودمندی بالا و کشف k بالا الگوهای سودمند بسته را معرفی کرده‌اند[۱۷].

- یائو[5] و همکاران، سودمندی مجموعه عناصر ممکن است متفاوت باشد. مفهوم سودمندی داخلی و سودمندی خارجی را معرفی کردند که در آن سودمندی داخلی نشان دهنده مقدار عنصر و سودمندی خارجی نشان دهنده واحد سود از عناصر برای تعیین مجموعه عناصر با سودمندی بالا است. الگوریتم های HUIM پیشین از مشکل ترکیبی برای کشف مجموعه عناصر با سودمندی بالا رنج می‌برند[۷].

- لان[6] و همکاران، الگوریتم مبتنی بر پیش بینی[7] که از مکانیزم شاخص[8] برای سرعت بخشیدن به اجرا و کاهش حافظه مورد نیاز در فرآیند استخراج HUIM سریع استفاده می‌کنند، معرفی کرده‌اند. علاوه بر این، از استراتژی هرس برای کاهش

1 . High-Utility Itemset Mining
2 . EFficient High-Utility Itemset Mining
3 . Damped Window Model
4 . Chan
5 . Yao
6 . Lan
7 . projection-based algorithm
8 . Index

تعداد مجموعه عناصر نامطلوب در استخراج استفاده می‌شود. نتایج نشان می‌دهد که الگوریتم PB از نظر محاسباتی بسیار قدرتمندتر از الگوریتم های دو مرحله‌ای است[۸].

– در [۱۸] چارچوب جدیدی را برای استخراج k بالا مجموعه عناصر با سودمندی بالا که k تعداد مورد نظر از مجموعه عناصر با سودمندی بالا برای استخراج است، ارائه می‌دهند. دو نوع الگوریتم کارآمد بنام TKU[۱] و TKO[۲] برای استخراج مجموعه عناصر بدون نیاز به تنظیم آستانه حداقل سودمندی پیشنهاد می‌دهند.

۲–۲–استخراج مجموعه عناصر با سودمندی بالا براساس مدل نرخ فراموشی

– در [۱۹] الگوریتم های مبتنی بر درخت با استفاده از مدل نرخ فراموشی برای استخراج مجموعه عناصر پرتکرار از جریان داده‌های نامعلوم پیشنهاد می‌کنیم. به طوری که وزن سنگین‌تر بر روی داده‌های اخیر به نسبت داده‌های قدیمی‌تر قرار داده می‌شود. اثر داده‌های قدیمی بر روی نتایج استخراج (بعنوان مثال، وزن آنها بر روی داده‌های قدیمی) با گذشت زمان کاهش می‌یابد.

– در [۲۰] الگوریتمی برای تخمین مقادیر چگالی در داده‌های جریانی با استفاده از تابع هش[۳] بنام λ-HCount ارائه شده است و برای تاکید بر اهمیت داده‌های اخیر از مدل نرخ فراموشی برای اندازه گیری فرکانس در داده‌های جریانی استفاده می‌شود.

– در [۲۱] الگوریتمی برای استخراج مجموعه عناصر مکرر بسته از داده‌های جریانی براساس مدل نرخ فراموشی پیشنهاد می‌دهد. با ایجاد درخت پویا، الگوریتم چگالی مجموعه عناصر در درخت الگو را با استفاده از فاکتور فراموشی محاسبه می‌کند و مجموعه عناصر غیرمکرر واقعی را از درخت الگو حذف می‌کند تا هزینه حافظه کاهش یابد. همچنین الگوریتم با تغییر درخت الگو و تشخیص مجموعه عناصر مکرر بسته در یک بازه زمانی ثابت، زمان محاسباتی را کاهش می دهد.

– در [۱۶] استخراج الگوی سودمند[۴] برای ارزیابی سودمندی الگوها مناسب‌تر است. در این مقاله از رویکرد استخراج الگوها با سودمندی متوسط بالا استفاده می‌شود که یکی از رویکردهای استخراج الگوی سودمند می باشد و الگوهای جالبی را کشف می‌کند که در آن عناصر با استفاده از یک معیار سودمندی جدید روابط معنادار بیشتری را بین یکدیگر دارند. این رویکرد با استفاده از مفهوم مدل نرخ فراموشی الگوهای مفید بیشتری را در محیط های جریان بدست می‌آورد.

۳–راهکار پیشنهادی

روش پیشنهادی، استفاده از مدل نرخ فراموشی در الگوریتم EFIM است. این مدل با استفاده از یک فاکتور فراموشی که توسط کاربر مشخص شده و با عنوان f که عدد واقعی آن بین صفر تا یک است به منظور کاهش اهمیت تراکنشها با گذر زمان استفاده می‌شود. برای محاسبه فاکتور فراموشی از فرمول $f^{t`-t}$ استفاده می‌شود که $t`$ زمان جاری و t زمان ورود یک تراکنش است. از این مدل، برای بدست آوردن اطلاعات به روز، اطلاعات مربوط از داده‌های جریانی با تفکیک تراکنشهای اخیر و قدیمی از لحاظ اهمیت آنها طراحی شده است.

الگوریتم EFIM (الگوریتم ۱) بعنوان ورودی یک پایگاه داده تراکنش‌ها، حداقل آستانه سودمندی تعیین شده توسط کاربر[۵]، نرخ فراموشی و تعداد تراکنشها در هر مرحله را دریافت می‌کنند و بعنوان خروجی، مجموعه عناصر با سودمندی بالا را بر می‌گردانند. الگوریتم در ابتدا در نظر می گیرد که مجموعه عناصر فعلی α یک مجموعه تهی است و متغیرهای دیگر را مقداردهی می‌کنیم (خط ۱ و ۲). سپس این الگوریتم پایگاه‌داده را با استفاده از آرایه utility-bin به منظور محاسبه سودمندی محلی هر عنصر، اسکن می‌کند (خط ۵). توجه داشته باشید که در جایی که $\alpha = 0$ است، سودمندی محلی از هر عنصر

1 . Mining Top-K Utility Itemsets
2 . Mining Top-K Utility Itemsets In One Phase
3 . Hash
4 . Utility Pattern Mining
5 . user-specified minimum utility threshold

TWU[1] است. سپس سودمندی محلی هر عنصر با آستانه minutil مقایسه می‌شود تا عناصر ثانویه α را بدست آورند، این مواردی است که باید در گسترش α در نظر گرفته شود (خط ۶). سپس این عناصر به ترتیب صعودی از TWU مرتب شده اند و از آن بعنوان نظم استفاده می‌شود (خط ۷). آرایه utility-bin را برای محاسبه سودمندی زیر درخت مقداردهی می‌کنیم (خط ۸). سپس پایگاه داده یکبار برای حذف تمام عناصری که عناصر ثانویه نیستند، اسکن می‌شود، از آنجایی که آنها نمی توانند بخشی از مجموعه عناصر با سودمندی بالا باشند (خط ۹). در همان زمان، عناصر در هر تراکنش مرتب می‌شوند و اگر یک تراکنش خالی شود، از پایگاه داده حذف می شود. سپس پایگاه داده دوباره برای مرتب سازی تراکنش‌ها اسکن می شود (خط ۱۰). سپس الگوریتم مجدداً پایگاه داده را برای محاسبه سودمندی زیر درخت از هر عنصر ثانویه با استفاده از آرایه utility-bin اسکن می‌کند (خط ۱۳ و ۱۴). پس از آن، الگوریتم روش بازگشتی جستجو را برای انجام جستجوی اول عمق با شروع از α فراخوانی می‌کند (خط ۱۵).

Algorithm 1: The EFIM algorithm

Input: **D**: a transaction database
minutil: a user- specified threshold
f: a given decaying factor
numberOfTransactionStep: The number of transactions at each stage
output: the set of high-utility itemsets
1. Initialize the current itemset $\alpha = 0$
2. t` = 0;
3. for each transaction in database
4. for each item in a transaction
5. calculate the local utility using a utility-bin array
6. Secondary(α) = $\{i | i \in I \wedge lu(\alpha, i) \geq minutil\}$
7. Sort promising items according to the increasing order of TWU using insertionSort
8. initialize the utility-bin array for counting the subtree utility
9. Scan D to remove $i \notin$ Secondary(α) from the transactions, and sort items in each transaction by increasing TWU order and delete empty transaction
10. Sort transactions in D the lexicographical order
11. for each transaction in database
12. for each item in transaction
13. Calculate the sub-tree utility $su(\alpha, i)$ of each item using utility-bin array;
14. Primary(α) = $\{i | i \in$ Secondary(α) $\wedge\ su(\alpha, i) \geq minutil\}$;
15. Call function **Search(α,$Primary(\alpha)$,$Secondary(\alpha)$, minutil, f, numberOfTransactionStep);**

روش جستجو (الگوریتم ۲) پارامترهایی را بعنوان مجموعه عناصر فعلی با توسعه α می‌گیرد، پیش بینی پایگاه داده، عناصر اولیه و ثانویه α، حداقل آستانه سودمندی تعیین شده توسط کاربر، نرخ فراموشی و تعداد تراکنشها در هر مرحله را دریافت می‌کنند. این روش در هر مرحله یک حلقه را برای در نظر گرفتن هر پسوند یک عنصر واحد به فرم $\beta = \alpha \cup \{i\}$ انجام می‌دهد، در جایی که i بعنوان عنصر اولیه از α است (خط ۱ تا ۱۳). تعداد تراکنشها را به چندین مرحله تقسیم بندی می‌کنیم (خط ۵). براساس مدل نرخ فراموشی برای هر یک از این توسعه β ، یک اسکن پایگاه داده برای محاسبه سودمندی β در همان زمان ساخت پیش بینی پایگاه داده β انجام می‌شود (خط ۶). توجه داشته باشید که ادغام تراکنشها در حالی انجام می‌شود که پیش بینی پایگاه داده β ساخته شده است. اگر سودمندی β بیشتر از حداقل آستانه سودمندی تعیین شده توسط کاربر باشد، β خروجی است که بعنوان مجموعه عناصر با سودمندی بالا در نظر گرفته می‌شود (خط ۷). سپس پایگاه داده دوباره برای محاسبه سودمندی زیر درخت و سودمندی محلی β از هر عنصر z که می‌تواند β توسعه یابد (عناصر ثانویه با α) با استفاده از دو آرایه utility-bin اسکن می شود (خط ۸). این امر اجازه تعیین عناصر اولیه و ثانویه از β را می‌دهد (خط ۹و ۱۰). سپس، روند جستجو بصورت بازگشتی با β با ادامه جستجوی اول عمق توسط توسعه β فراخوانی می‌شود (خط ۱۱).

1 . Transaction Weighted Utilization

براسـاس ویژگیهای ارائه شده، می توان مشاهده کرد که وقتی الگوریتم EFIM براساس مدل نرخ فراموشی به پایان می‌رسد، همه و تنها مجموعه عناصر اخیر با سودمندی بالا خروجی بوده‌اند.

Algorithm 2: The Search procedure

Input: $(\boldsymbol{\alpha})$: an itemset,
$\boldsymbol{\alpha - D}$: the α projected database,
Primary$(\boldsymbol{\alpha})$: the primary items of α ,
Secondary$(\boldsymbol{\alpha})$**:** the secondary items of α,
minutil: The minutil threshold,
f: a given decaying factor
numberOfTransactionStep: The number of transactions at each stage
Output: the set of high-utility itemsets that are extensions of α

1. **foreach** $i \in$ numberOfTransactionStep
2. **foreach** item $i \epsilon$ primary(α) **do**
3. $\beta = \alpha \cup \{i\}$
4. t` = t`+1;
5. D.SIZE / numberOfTransactionStep //Determine the number of transactions per step
6. Scan $\alpha - D$ to calculate $u(\beta) * f^{t`-t}$ and create $\beta - D$ //uses transaction merging
7. if $u(\beta) \geq minutil$ then output β
8. calculate $su(\beta, z)$and $lu(\beta, z)$ for all item $z \epsilon$ Secondary(α) by scanning $\beta - D$ once, using two utility-bin arrays
9. Primary(β) = $\{z \epsilon$ Secondary$(\alpha) | su(\beta, z) \geq minutil\}$
10. Secondary(β) = $\{z \epsilon$ Secondary$(\alpha) | lu(\beta, z) \geq minutil\}$
11. Call **Search**$(\beta, \beta - D, Primary(\beta), Secondary(\beta)$, minutil, f, numberOfTransactionStep)
12. **end**
13. **end**

در الگوریتم پیشنهادی، هدف از مدل نرخ فراموشی استخراج اطلاعات مهم الگوهای اخیر بطور موثر می‌باشد. با توجه به اصـل مدل نرخ فراموشی، این روش به طور مداوم سودمندی تراکنشهای تولید شده در گذشته را کاهش می‌دهد و بر روی داده‌های اخیر به جای داده‌های قدیمی تمرکز می‌کند زیرا اسـتفاده از این رویکرد سـبب می‌شود که تاثیر داده‌های قدیمی به تدریج کاهش یابد.

۴–نتایج و بحث

ما چندین آزمایش را برای ارزیابی عملکرد الگوریتم EFIM براسـاس مدل نرخ فراموشـی انجام می‌دهیم. آزمایش‌ها بر روی یک کامپیوتر با یک پردازنده ۶۴ بیتی نسـل چهارم با هسـته i7 در ویندوز ۸,۱ و ۱۶ گیگابایت رم اجرا می‌شـود. عملکرد الگوریتم پیشنهادی را با چهار الگوریتم اخیر به نامهای EFIM، HUI-Miner، FHM و HUP-Miner مقایسه کردیم. الگوریتم پیشنهادی با زبان جاوا پیاده سازی شده است.

۴–۱–مجموعه داده ها

آزمایش ها با اسـتفاده از مجموعه داده‌های اسـتاندارد مورد اسـتفاده در ادبیات HUIM برای ارزیابی الگوریتم‌های HUIM بنامهای (Accident, BMS) انجام شد. این مجموعه داده‌ها به این دلیل انتخاب می‌شوند که ویژگی‌های متفاوتی دارند. جدول ۱و ۲ خلاصـه ای از این ویژگیهاسـت. تصـادف[1] یک مجموعه داده بزرگ با تراکنشـهای نسـبتاً طولانی اسـت. BMS یک مجموعه داده پراکنده با تراکنشهای کوتاه است.

1 . Accident

جدول ۱- ویژگیهای داده ها

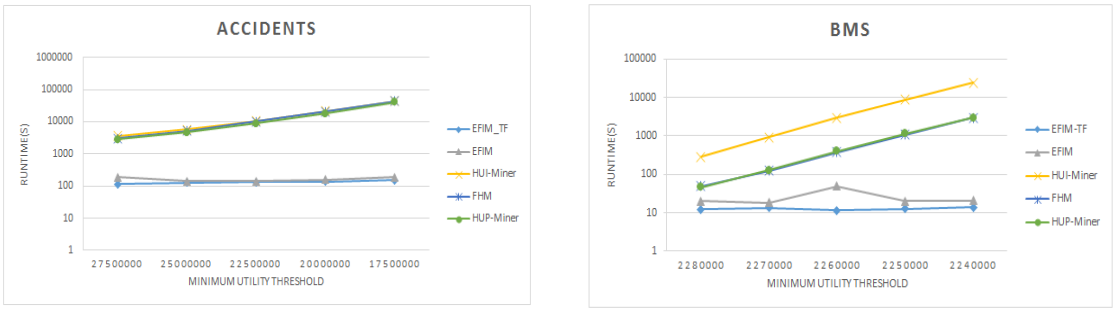
Dataset	# Transaction	# Distinct Items	Avg. Trans.Length
Accident	340,183	468	33.8
BMS	59,601	497	4.8

جدول ۲- انواع داده ها

Dataset	Type
Accident	moderately dense, moderately long transactions
BMS	sparse, short transactions

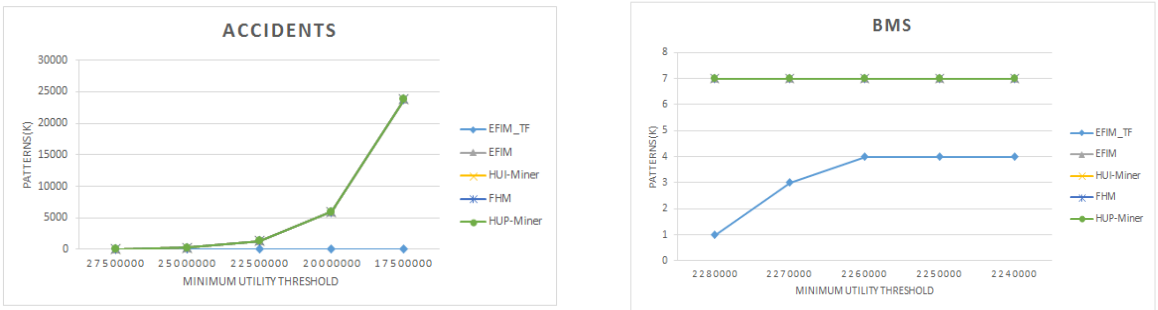
۴-۲-مقایسه عملکرد زمان اجرا

ابتدا عملکرد زمان اجرای الگوریتم‌های مختلف را مقایسه می‌کنیم. شکل ۱، عملکرد زمان اجرای الگوریتم‌های تحت مقادیر متفاوت minutil را با نرخ فراموشی $f = 0.99999$ نشان می‌دهد. الگوریتم پیشنهادی از نظر زمان اجرا بر روی مجموعه داده‌های مختلف نسبت به سایر الگوریتم ها بهتر عمل می‌کند. در الگوریتم پیشنهادی به دلیل استفاده از مدل نرخ فراموشی، زمان اجرا کوتاه‌تر است چرا که تعداد الگوهای کاندید تولید شده با حذف الگوهای غیرضروری که اهمیت کمتری در داده‌های اخیر دارند، کاهش می‌یابد.



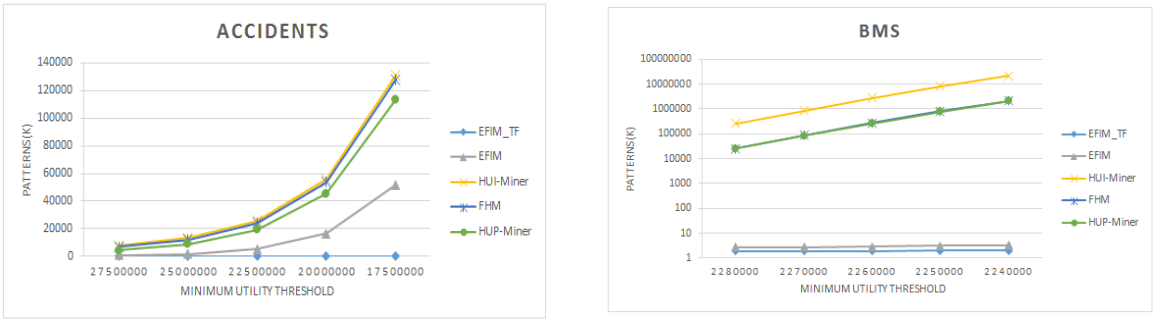
شکل ۱-زمان اجرا در مجموعه داده های مختلف

شکل ۲، تعداد الگوهای تولید شده توسط الگوریتم‌های مقایسه شده با نرخ فراموشی $f = 0.99999$ نشان می‌دهد. در مقایسه الگوریتم پیشنهادی با دیگر الگوریتم‌ها، به دلیل استفاده از مدل نرخ فراموشی تعداد کمتری از الگوها تولید می‌شود، زیرا الگوریتم پیشنهادی به احتمال زیاد به تولید الگوهایی با سودمندی کم در داده‌های اخیر نمی پردازد. با این حال الگوریتم‌های دیگر تعداد مشابهی از الگوها را تولید می‌کنند.



شکل ۲-تعداد الگوهای قابل توجه

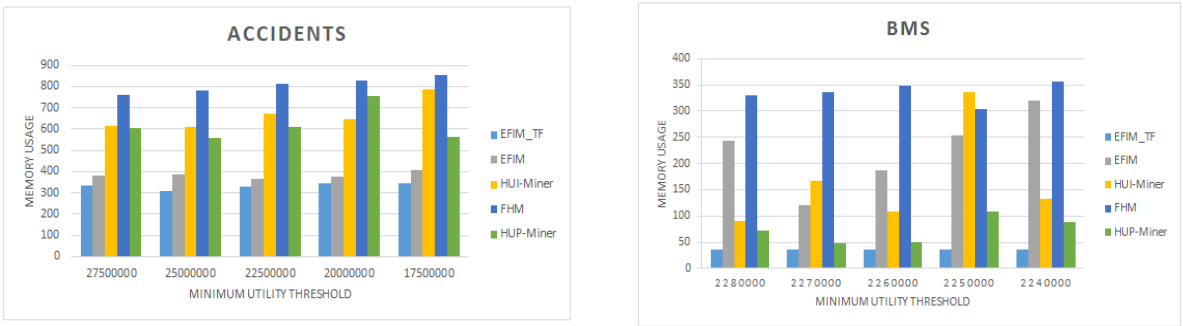
شکل ۳، الگوهای کاندید تولید شده توسط هر الگوریتم با توجه به مجموعه داده‌ها و با نرخ فراموشی $f = 0.99999$ نشـان می‌دهد. با توجه به نتایج، الگوریتم‌های دیگر در مقایسـه با الگوریتم پیشـنهادی تعداد بیشتری الگوهای کاندید تولید می‌کنند. الگوریتم پیشـنهادی به دلیل اسـتفاده از مدل نرخ فراموشی، کمترین تعداد کاندیداها را در هر مورد تضمین می‌کند. چرا که تعداد الگوهای کاندید تولید شـــده با حذف الگوهای غیرضـــروری که اهمیت کمتری در داده‌های اخیر دارند، کاهش می‌یابد.



شکل ۳-تعداد الگوهای کاندید

۴-۳- مقایسه عملکرد حافظه

سپس، عملکرد حافظه الگوریتم‌ها را مقایسه می کنیم. شکل ۴، عملکرد حافظه الگوریتم‌ها را نشان می‌دهد. بنابراین، از طریق مشـــاهدات در نتایج تجربی آزمون حافظه، الگوریتم پیشـــنهادی دارای بهترین حافظه برای پردازش داده‌ها اســـت. چون در الگوریتم پیشـنهادی با اسـتفاده از مدل نرخ فراموشی، حافظه مورد استفاده برای داده‌های زمان سپری شده با حذف داده‌های قدیمی ذخیره می‌شود.



شکل ٤-عملکرد حافظه

۵- نتیجه‌گیری

در این مقاله، ما یک الگوریتم جدید داده کاوی به نام EFIM براسـاس مدل نرخ فراموشـی به منظور استخراج موثر مجموعه عناصـر اخیر با سودمندی بالا را پیشنهاد کردیم. در روش پیشنهادی، از مدل نرخ فراموشی در الگوریتم EFIM برای تاکید بر اهمیت عناصـر داده اخیر، اسـتفاده نموده‌ایم. در این مقاله، از مجموعه داده‌های واقعی مختلف برای مقایسه عملکرد الگوریتم پیشنهادی با سایر الگوریتم‌های استخراج الگوی مبتنی بر سودمندی استفاده و نتایج تجربی نشان داد که روش پیشنهادی نیاز به حافظه کمتر و زمان اجرای کمتری نسبت به سایر الگوریتم‌ها دارند.

۶–پیشنهادها

در تحقیقات آینده، مثلا با طبقه بندی یا خوشه بندی داده‌ها در فرآیند استخراج الگو، ممکن است بتوانیم اطلاعات الگوی قابل اعتمادتری را کسب کنیم.

۷–مراجع

1. Agrawal, R. and R. Srikant, Fast Algorithms for Mining Association Rules.
2. Han, J., et al., Mining frequent patterns without candidate generation: A frequent-pattern tree approach. 2004. **8**(1): p. 53-87.
3. Uno, T., M. Kiyomi, and H. Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. in Fimi. 2004.
4. Zaki, M.J.J.I.t.o.k. and d. engineering, Scalable algorithms for association mining. 2000. **12**(3): p. 372-390.
5. Zida, S., et al., EFIM: a fast and memory efficient algorithm for high-utility itemset mining. 2017. **51**(2): p. 595-625.
6. Ahmed, C.F., et al., Efficient tree structures for high utility pattern mining in incremental databases. 2009. **21**(12): p. 1708-1721.
7. Yao, H., H.J. Hamilton, and C.J. Butz. A foundational approach to mining itemset utilities from databases. in Proceedings of the 2004 SIAM International Conference on Data Mining. 2004. SIAM.
8. Lan, G.-C., et al., An efficient projection-based indexing approach for mining high utility itemsets. 2014. **38**(1): p. 85-107.
9. Liu, Y., W.-k. Liao, and A. Choudhary. A two-phase algorithm for fast discovery of high utility itemsets. in Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2005. Springer.
10. Song, W., et al., BAHUI: fast and memory efficient mining of high utility itemsets based on bitmap. 2014. **10**(1): p. 1-15.
11. Tseng, V.S., et al., Efficient algorithms for mining high utility itemsets from transactional databases. 2013. **25**(8): p. 1772-1786.
12. Yun, U., H. Ryang, and K.H.J.E.S.w.A. Ryu, High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates. 2014. **41**(8): p. 3861-3878.
13. Liu, M. and J. Qu. Mining high utility itemsets without candidate generation. in Proceedings of the 21st ACM international conference on Information and knowledge management. 2012. ACM.
14. Fournier-Viger, P., et al., SPMF: a Java open-source pattern mining library. 2014. **15**(1): p. 3389-3393.
15. Krishnamoorthy, S.J.E.S.w.A., Pruning strategies for mining high utility itemsets. 2015. **42**(5): p. 2371-2381.
16. Yun, U., et al., Damped window based high average utility pattern mining over data streams. 2018. **144**: p. 188-205.
17. Chan, R., Q. Yang, and Y.-D. Shen. Mining high utility itemsets. in Third IEEE international conference on data mining. 2003. IEEE.
18. Tseng, V.S., et al., Efficient algorithms for mining top-k high utility itemsets. 2015. **28**(1): p. 54-67.
19. Leung, C.K.-S. and F. Jiang. Frequent itemset mining of uncertain data streams using the damped window model. in Proceedings of the 2011 ACM Symposium on Applied Computing. 2011. ACM.
20. Chen, L. and Q.J.I.S. Mei, Mining frequent items in data stream using time fading model. 2014. **257**: p. 54-69.
21. Caiyan, D., C.J.I.J.o.C.S. Ling, and Engineering, An algorithm for mining frequent closed itemsets with density from data streams. 2016. **12**(2-3): p. 146-154.