علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۶، شماره ۲، ۱۳۹۷
صفحات ۴۹- ۶۶
مقاله عادی

روشی سریع‌تر برای تشخیص گزارش خطای تکثیر با حفظ صحت

زهرا امین‌الرعایائی[۱] بهزاد سلیمانی نیسیانی[۲] محمد حسین ندیمی شهرکی[۳]

[۱]موسسه آموزش عالی علامه نائینی، نائین، اصفهان، ایران
[۲]دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، اصفهان، ایران
[۳]دانشکده مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد نجف‌آباد، نجف‌آباد، اصفهان، ایران

چکیده

امروزه یکی از مهم‌ترین چالش‌های سیستم‌های ردیابی گزارش‌های خطای کاربران، تشخیص گزارش‌های خطای تکراری است. بسیاری از محققان از روش‌ها و ابزارهای بازیابی اطلاعات برای حل این مشکل استفاده کرده‌اند که در این پژوهش نیز از آن‌ها با معرفی چندین ویژگی استخراج شده جدید مبتنی بر کمینه و بیشینه و میانگین تعداد تکرار کلمات مشابه در دو گزارش بهره گرفته شده است. ابتدا با در نظر گرفتن مجموعه داده‌هایی از ۴ مخزن بزرگ گزارش خطای Android Mozilla, OpenOffice, و Eclipse تعداد ۱۶۲ ویژگی جدید با ترکیب ویژگی‌های موجود در کارهای گذشته به‌دست آمده است. سپس بسیاری از این ویژگی‌ها، به دلیل اهمیت ناچیز و طولانی کردن زمان اجرای الگوریتم‌های طبقه‌بندی، با اعمال روش‌های کاهش بعد حذف شده‌اند. نتایج پیاده‌سازی نشان می‌دهد که زمان اجرای الگوریتم‌های طبقه‌بندی با ویژگی‌های کاهش یافته نسبت به زمان اجرای تمام ویژگی‌ها، از میزان چندین دقیقه به چندین ثانیه کاهش یافته است و در عین حال نیز باعث بهبود تشخیص گزارش خطای تکراری بین ٪۱ الی ٪۶ شده است. همچنین نتایج به دلیل وجود ویژگی‌های جدید، بیانگر دقت بالای ٪۹۶ و نرخ فراخوانی بالای ۰/۹۰ نسبت به دیگر تحقیقات پیشین شده است.

کلمات کلیدی: استخراج ویژگی، تشخیص گزارش خطای تکثیر، الگوریتم‌های طبقه‌بندی، دقت، زمان اجرا.

۱- مقدمه

با ظهور اینترنت و گسترش کاربران کامپیوتر، نرم‌افزارهای کاربردی بسیاری توسعه یافتند که توسط میلیون‌ها کاربر برای امور روزمره استفاده می‌شوند مانند نرم‌افزارهای اداری یا مرورگرهای وب[۱]. فعالیت‌های تعمیر و نگهداری بیش از دوسوم هزینه چرخه حیات یک سیستم نرم‌افزاری را دربر دارد، به‌طوری‌که میلیون‌ها دلار هزینه بابت خطاهای نرم‌افزار از بین می‌رود. در نتیجه بسیاری از پروژه‌های نرم‌افزاری بر روی سیستم‌های ردیابی خطا به منظور مدیریت فعالیت‌های نگهداری اصلاحی تکیه کرده‌اند. هر نرم‌افزاری تحت شرایط مختلف ممکن است دچار خطا شود. خطاهای نرم‌افزار به دلایل مختلفی اتفاق می‌افتد. گزارش خطاها[۲] یا به صورت خودکار تولید می‌شوند و یا توسط کاربر نرم‌افزار پیش‌نویس شده‌اند. گزارش خطاها گاهی بر اثر اختلال عملکرد در نرم‌افزار دیگر یا نسخه‌ی بتای نرم‌افزار به وجود می‌آید. اکثر گزارش خطاها با کمک تجارب دیگر کاربران که در واقع از مشکل رخ داده در سیستم اطلاع دارند بهبود می‌یابد. پرداختن به این خطاها یعنی تلاش و صرف زمان در فاز نگهداری از یک چرخه حیات پروژه‌ی نرم‌افزاری. در اکثر موارد گزارش خطاها توسط چندین کاربر مختلف ارسال می‌شوند. با این وجود هر گزارش خطا به صورت مجزا و با دقت تجزیه و تحلیل می‌شود. به سیستمی که مسئول پردازش خطاهای تازه منتشر شده، چک کردن گزارش‌های تکراری و ارسال آن‌ها به توسعه‌دهنده‌ی مناسب است، سیستم مراقبت نرم‌افزار[۳] گفته می‌شود. کل این فرآیند را، سیستم ردیابی خطا نامیده‌اند. سیستم ردیابی خطا مانع از ورود تعداد زیادی از گزارش خطاهای تکراری به سیستم می‌شود. در بسیاری از پروژه‌های نرم‌افزاری منبع باز، بیش از یک سوم از تمام گزارش‌ها، تکراری هستند [۵]. شناسایی این تکرار در گزارش خطاها زمان و

هزینه زیادی را به همراه دارد. از این رو تاکنون راهکارهایی برای تشخیص بهتر و سریع‌تر تکرار در گزارش خطاها انجام شده است که تا حد زیادی باعث سرعت‌العمل در تشخیص خطا و صرف زمان کمتر را به دنبال دارد.

به دلیل وجود پیچیدگی در سیستم‌های نرم‌افزاری بزرگ، خطای نرم‌افزار یک پدیده‌ی اجتناب‌ناپذیر است. توسعه‌ی نرم‌افزار یک روند تکاملی است که در آن پس از انتشار اولین نسخه، خطاها توسط کاربران و تست‌کننده‌ها گزارش می‌شوند. سیستم ردیابی خطا یا سیستم ردیابی نقص یک برنامه کاربردی نرم‌افزار است که گزارش خطاهای نرم‌افزار را در مرحله پروژه‌های توسعه نرم‌افزار در خود نگه می‌دارد. اغلب سیستم‌های ردیابی خطا که از پروژه‌های نرم‌افزاری منبع باز استفاده می‌کنند به کاربران نهایی اجازه می‌دهند تا خود به طور مستقیم خطاها را وارد کنند [۱]. سیستم‌های دیگر، گزارش خطاها را در خود شرکت مربوطه و یا سازمان انجام توسعه نرم‌افزار ثبت می‌کنند. سیستم‌های ردیابی خطای عمومی با دیگر برنامه‌های کاربردی مدیریت پروژه نرم‌افزار تلفیق شده است. پایگاه داده گزارش‌ها به دلیل توصیف دو خطایی که نیاز به تعمیر و یا اضافه کردن ویژگی را دارند، برای بسیاری از پروژه‌های نرم‌افزاری مهم است. به طور معمول پروژه‌های نرم‌افزاری منبع باز درون خود یک مخزن خطا دارند که اجازه می‌دهد هم کاربران و هم توسعه‌دهندگان مشکلات مواجه شده با نرم‌افزار را ارسال کنند که نشان‌دهنده‌ی پیشرفت ممکن و اظهار نظر در مورد گزارش‌های موجود است. پروژه‌های منبع باز محبوب تعداد زیادی از گزارش خطاها را در خود دارند [۳]. در بیشتر نرم‌افزارهای پیچیده خطاهای بیشتری گزارش می‌شوند که به راحتی می‌توان آن‌ها را به کار گرفت. گزارش خطاها به دلایل مختلفی اتفاق می‌افتند از جمله: محدودیت مشخصات تعریف شده، بی‌دقتی، مشکل سوءتفاهم برنامه‌نویسان، مسائل فنی، کیفیت غیرکاربردی و غیره. همچنین خطاهای نرم‌افزار به طور قابل توجهی هزینه‌ی سنگینی دربر دارند. تحقیقات نشان می‌دهد که هزینه‌ی خطاهای نرم‌افزاری در ایالات متحده امریکا سالانه میلیون‌ها دلار است [۵]. بسیاری از پروژه‌های نرم‌افزاری، روش‌هایی برای گزارش خطا و یا ذخیره‌ی آن‌ها در یک سیستم ردیابی خطا را در اختیار کاربران قرار می‌دهند. گزارش خطا (به عبارتی نقص[۴] یا اشکال[۵]) یک بخش جدایی‌ناپذیر از فرآیند توسعه[۶]، تست[۷] و نگهداری[۸] نرم‌افزار است.

به‌طور کلی خطاها توسط سیستم ردیابی خطا که قبلاً توسط سیستم مراقبت نرم‌افزار تحلیل شده، گزارش می‌شوند. سیستم مراقبت نرم‌افزار نسبت به سیستم، پروژه و توسعه‌دهندگان آگاهی دارد و باید تعیین کند که آیا خطا بامعنی است؟ و آن را به سمت توسعه‌دهنده مناسب هدایت کند [۱۲]. هر گزارش تحلیل شده توسط سیستم مراقبت نرم‌افزار، اگر معتبر بود باید تعیین کند که این گزارش در چه طبقه‌ای از فرآیند توسعه قرار دارد. علاوه بر این یکی از مهم‌ترین وظیفه سیستم مراقبت نرم‌افزار تشخیص گزارش خطاهای تکراری است که قبلاً ارسال شده‌اند ولی هنوز کشف نشده‌اند. در پروژه‌های نرم‌افزاری بزرگ که شامل هزاران گزارش خطا هستند تشخیص خطاهای تکراری وظیفه‌ی سنگینی هست.

استفاده از مخازن باز ردیابی خطا شبیه Bugzilla و Google's issue-tracker در بسیاری از نرم‌افزارهای کاربردی رایج است [۱۳]. این مخازن به توسعه‌دهندگان، تست‌کننده‌ها و کاربران اجازه می‌دهند تا خطاهای موجود در سیستم را گزارش و وضعیت حل آن‌ها را دنبال کنند. در واقع یک مخزن خطا موقعیتی را برای کاربران، تیم تضمین کیفیت، توسعه‌دهندگان و مدیران شرکت به منظور فرآیند توسعه یکپارچه فراهم می‌کند. با این حال مخزن خطا دربردارنده‌ی هزینه است. ردیابی خطا به منظور انجام فعالیت‌هایی ذیل انجام می‌گیرد: بررسی کیفیت گزارش از جهت دربر داشتن اطلاعات مفید و موردنیاز، تشخیص تکثیر خطا، مسیریابی خطا به سمت متخصص مناسب به منظور اصلاح و ویرایش فرادادهای[۹] مختلف پروژه و خصوصیات همراه با گزارش خطا (به عنوان مثال: وضعیت فعلی، اختصاص آن به توسعه‌دهنده، تأثیر خطا و غیره). استفاده از سیستم ردیابی خطا می‌تواند روند توسعه را با روش‌های زیر بهبود دهد:

۱) ردیابی پروژه با فهم اینکه چگونه بسیاری از گزارش‌ها مهم تلقی می‌شوند؛ تکامل می‌یابد.

۲) به توسعه‌دهندگان اجازه می‌دهد برای برقراری ارتباط پیرامون توسعه پروژه به لحاظ جغرافیایی تفکیک شوند.

۳) این روش قادر به تعیین توسعه‌دهندگان متخصص در زمینه‌های مختلف محصول است.

۴) به بهبود کیفیت نرم‌افزار تولید شده کمک می‌کند.

۵) به کاربران دیدی نسبت به وضعیت گزارش خطا بدهد.

اگر چه مزایای سیستم‌های ردیابی خطا در پروژه‌های توسعه نرم‌افزار آشکار است، اما استفاده از آن‌ها مشکلاتی را نیز دربر دارد. برخی از مشکلات ممکن است نتیجه‌ی استفاده از سیستم ردیابی خطا که به صورت پویا به گزارش خطاها اختصاص داده شده‌اند، کیفیت توصیفات گزارش خطاها، تکامل نرم‌افزار و قابلیت ردیابی و تشخیص گزارش خطاهای تکراری باشد.

هنگام بررسی گزارش‌ها توسط توسعه‌دهندگان دو تأثیر وجود دارد. اولین تأثیر این است که توسعه‌دهنده از مرحله‌ی بهبود محصول به سمت مدیریت پروژه هدایت می‌شوند. اگر یک پروژه ۳۰ گزارش در یک روز بگیرد و پنج دقیقه طول کشد تا تنها یک گزارش تریاگر شود، بیش از دو نفر-ساعت در هر روز صرف تریاگر گزارش می‌شود. اگر همه این گزارش‌ها منجر به بهبود کد شوند، این هزینه ممکن است برای پروژه رضایت‌بخش باشد. با این حال، برای برخی از پروژه‌ها، کمتر از نیمی از گزارش‌های ارائه شده به بهبود کد منجر شود. به عنوان مثال در پروژه Eclipse، ۵۵۱۵ گزارش غیر مولد در سال ۲۰۰۴ بود [۷]. تأثیر دوم این‌که ممکن است گزارش‌ها به موقع خوانده نشوند. اگر تعدادی از گزارش‌هایی که وارد مخزن می‌شوند بیش از مقدار زمان مناسب یک پروژه باشد، برخی از گزارش‌ها در مخزن باقی مانده و نسبت به گزارش‌های دیگر از اولویت بالاتری برخوردارند. برای یک پروژه منبع باز که در آن پاسخ‌ها از تیم توسعه به سمت اجتماع است اغلب بررسی می‌شود که چگونه گزارش‌ها به سرعت آدرس‌دهی شوند و تعدادی از گزارش‌های برجسته، که در آن نرخ گزارش تریاگر شده می‌تواند یک عامل مهم در تعیین اینکه چگونه به خوبی پروژه در حال وقوع پالایش می‌شود [۵].

در این مطالعه نشان داد که میزان موفقیت یک پروژه منبع باز نرخ ارسال گزارش خطاها توسط کاربران و در لیست‌های پستی پروژه شرکت است. شخصی که این گزارش‌ها را تریاگر می‌کند باید دو هدف داشته باشد. اول این‌که مخزن حاوی کوچک‌ترین مجموعه‌ای از بهترین گزارش این پروژه است. کوچک‌ترین مجموعه از بهترین گزارش مطلوب است زیرا گزارش‌ها به طور معمول از منابع مختلف، مانند اعضای یک بخش فنی پشتیبانی، توسعه‌دهندگان دیگر، و جامعه کاربران وارد مخزن شده است. متأسفانه در بسیاری از منابع گوناگون گزارش، برخی از گزارش‌ها معنی‌دار نیستند. به عنوان مثال، در یک پروژه بزرگ با تعداد اعضای تیم زیاد ممکن است چند توسعه‌دهنده یک گزارش خطای یکسان را ارسال کنند. این گزارش تکراری باید با هم یکی شود تا تلاش‌های تیم توسعه با داشتن حل دو خطای مشابه هدر نرود. یک سیستم مراقبت نرم‌افزار نیز نیاز به فیلتر گزارش‌های تکراری دارد. گاهی اوقات یک سیستم مراقبت نرم‌افزار نیز نیاز به فیلتر کردن گزارش‌های هرزنامه[۱۰] دارد. در نهایت، یک سیستم مراقبت نرم‌افزار ممکن است نشان دهد که خطا تعمیر خواهد شد و یا این‌که ویژگی‌هایی به محصول اضافه نشده است. در گزارش‌ها هر یک از این معیارها باید مشخص شود به‌طوری‌که تلاش‌های تیم توسعه بتواند منجر به بهبود محصول شود. به عنوان مثال، نزدیک به یک‌سوم از گزارش‌های ارائه شده پروژه فایرفاکس بین سال‌های ۲۰۰۳ تا ۲۰۰۵ تکراری بوده است [۸]. در واقع می‌توان نتیجه تصمیمات سیستم مراقبت نرم‌افزار را در یک گزارش به عنوان یک تصمیم‌گیری مخزن‌گرا و نه معنی‌دار در نظر گرفت.

تشخیص خودکار گزارش خطاهای تکراری و ارتباط گزارش خطاهای مشابه با توجه به دلایل زیر یک مسئله چالش برانگیز است:

- گزارش خطاها به صورت زبان طبیعی متنی بیان شده‌اند. به‌طوری‌که زبان طبیعی وسیع و مبهم است.
- مقدار گزارش‌های خطا در تنظیمات نرم‌افزارهای بزرگ و پیچیده بیشتر هستند.
- گزارش خطاها اغلب ضعیف بیان شده‌اند (اطلاعات از دست رفته یا متن مخدوش).

۱-۱- اصطلاحات گزارش خطا

در این قسمت مفاهیم پایه در این زمینه به شرح زیر است:

۱) خطا: خطای نرم‌افزار یک اشکال[۱۱]، نقص[۱۲] و یا عیب[۱۳] در یک برنامه یا سیستم کامپیوتری است که نتایج و رفتار غیر منتظره تولید می‌کنند. بین خطا و اشکال تمایز وجود دارد. یک اشکال می‌تواند خطا باشد اما همیشه یک خطا نیست. اشکال می‌تواند یک سند گم شده یا یک وظیفه و غیره باشد. فرآیند یافتن و حذف خطاها "خطازدایی[۱۴]" نامیده می‌شود. خطاها ممکن است به علت اشتباهات ناچیز برنامه‌نویسی باشند، اما نتایج حاصل از آن جدی باشد و یافتن و اصلاح آن‌ها یک وظیفه نسبتاً چالش برانگیز باشد.

۲) گزارش خطا: گزارش خطا یک سند نرم‌افزار است که خطاهای نرم‌افزاری که توسط توسعه‌دهنده، تست‌کننده و یا کاربر نهایی ارائه شده است را توصیف می‌کند [۹]. معمولاً یک گزارش خطا ترکیبی است از: شماره شناسایی آن خطا، عنوان[۱۵]، شدت[۱۶] یا اهمیت، برنامه‌نویس اختصاص یافته برای رفع اشکال، وضعیت حل خطا (جدید، تأیید نشده یا حل شده)، توصیف گزارش (مانند: مراحل بازسازی خطا، ردیابی پشته و رفتار مورد انتظار)، نظرات اضافی (بحث در مورد راه‌حل‌های ممکن)، فایل پیوست (به عنوان مثال: تکه‌های پیشنهادی، موارد آزمون) و یک لیست از گزارش‌هایی که باید قبل از حل این گزارش رسیدگی شوند [۱۰].

۳) مخزن خطا: مخازن خطا اغلب در پروژه‌های نرم‌افزار منبع باز استفاده می‌شود تا هر دو توسعه‌دهندگان و کاربران مشکلاتی را که با نرم‌افزار مواجه شدند، ارسال کنند، و این بیانگر این است که پیشرفت امکان‌پذیر است و روی گزارش خطاهای موجود نظر می‌دهد [۱۱]. یک مخزن خطای باز برای عموم قابل مشاهده است. خطاهای گزارش شده در مخازن خطا ممکن است توسط عموم شناسایی و حل شوند و از این طریق کیفیت پروژه‌های منبع باز بهبود می‌یابد [۱۴].

۴) تریاگ گزارش خطا[۱۷]: تریاگ گزارش خطا شامل مراحل زیر است: دانستن این‌که آیا یک خطا، خطای واقعی است؟ یا بررسی اینکه آیا خطای گزارش شده یک مورد تکراری از خطای موجود است؟ اولویت گزارش خطاها و تصمیم‌گیری این‌که کدام توسعه‌دهنده باید روی گزارش خطا مورد نظر کار کند [۱۵].

۵) تکرار گزارش خطا[۱۸]: گزارش خطاهای تکراری به گزارش خطایی با خطای یکسان که توسط گزارش‌دهندگان مختلف نوشته شده ارجاع دارد، به این دلیل که بسیاری از کاربران در تعامل با یک سیستم هستند و خطاها را گزارش می‌کنند. تشخیص گزارش خطاهای تکراری یک روش تریاگ خطا است که باعث کاهش هزینه سیستم مراقبت نرم‌افزار شده و برای توسعه‌دهندگان موجب صرفه‌جویی در زمان رفع خطاهای مشابه می‌شود [۱۳].

۶) سیستم ردیابی خطا: سیستم ردیابی خطا یا سیستم ردیابی نقص، مدیریت گزارش خطاها و توسعه‌دهندگانی که خطاها را رفع می‌کنند را بر عهده دارد [۱۵]. سیستم‌های ردیابی خطا برای پیگیری خطاهای نرم‌افزار طراحی شده‌اند. خطاها در یک مخزن خطا که جزء اصلی از یک سیستم ردیابی خطا است، ذخیره می‌شوند. برای ارسال و حل خطاها به صورت خودکار، توسعه‌دهندگان از پروژه‌های منبع باز محبوب سیستم‌های ردیابی خطا مانند: Mozilla, Eclipse, Linux kernel استفاده می‌کنند.

۱-۲- انگیزه تحقیق

شناسایی گزارش خطای تکراری به جهت صرفه‌جویی در زمان و تلاش توسعه‌دهندگان از اهمیت زیادی برخوردار است. اخیراً بسیاری از محققان روی این مسئله تمرکز کرده‌اند [۹]. برخی از انگیزه‌های مهم تشخیص گزارش خطای تکراری عبارتند از:

- ممکن است گزارش خطای تکراری به اشتباه به توسعه‌دهندگان مختلف اختصاص داده شود که این امر منجر به اتلاف زمان و تلاش توسعه‌دهندگان می‌شود.
- علاوه بر این هنگامی که یک گزارش خطا اصلاح می‌شود، پرداختن به موارد تکراری به عنوان خطاهای مستقل موجب اتلاف وقت است.
- در نهایت، شناسایی گزارش خطای تکراری می‌تواند در رفع خطاها مفید باشد چون که برخی از گزارش خطاها ممکن است توصیف مفیدتری از تکرارهای خود داشته باشند.

مسئله گزارش خطای تکراری و تأثیر آن در توسعه نرم‌افزار و تأکید نیازمندی یک سیستم به تشخیص گزارش خطای تکراری اخیراً برجسته شده است. در یکی از پژوهش‌ها، مخازن خطا، جستجو و تجزیه و تحلیل خطا در ۸ پروژه‌ی منبع باز در سال ۲۰۰۸ بررسی شده است. یافته‌های آن‌ها در جدول ۱ خلاصه شده است.

متوسط زمان سپری شده در جستجو و تحلیل گزارش خطا قبل از باز کردن یک گزارش خطای جدید ۱۲/۵ دقیقه است. همچنین تقریباً به طور متوسط ۴۸ نفر ساعت در طول روز تنها صرف جستجوی گزارش خطاهای مشابه می‌شود. بیشترین میزان خطاهای تکراری در Bugzilla و ٪۶۸ ذکر شده است.

علاوه بر این، برخی از پروژه‌ها وب سایت‌هایی را برای کسب اطلاعات در مورد گزارش خطاهای تکراری به خود اختصاص داده‌اند. به عنوان مثال پروژه Mozilla شامل یک صفحه Most frequently reported bugs است به منظور ردیابی خطاهایی که اغلب بیشترین تکرار را گزارش داده‌اند و شمارش تعداد موارد تکراری خطاها [۱۰]. صفحه وب به منظور به حداقل رساندن تعداد خطاهای تکراری وارد شده به Bugzilla انجام می‌شود که به نوبه خود موجب صرفه‌جویی زمان مهندسان تضمین کیفیت[۱۹] برای ردیابی خطا می‌شود. از آمارهای گزارش شده بدیهی است که گزارش خطاهای تکراری یک مشکل شایع در پروژه‌های نرم‌افزاری منبع باز است.

۲- گزارش خطای تکثیر

پرداختن به موضوع کثرت گزارش خطاها در فاز نگهداری چرخه‌ی پروژه‌ی نرم‌افزار اهمیت زیادی دارد. محققان برای سهولت در تشخیص نظام‌مند خطا و تصحیح آن سعی در افزایش سیستم‌های ردیابی خطا دارند. ابزارهای گزارش باز و دموکراتیک یک چالش عمده را به همراه دارد: کاربران اغلب مشکلی را ارائه می‌دهند که مشابه است [۱۶]. زمانی که دو خطا یک اشکال مشابه و با مفهوم یکسانی را توصیف کنند به نحوی که حل یکسانی نیز داشته باشند، در این حالت می‌گوییم گزارش خطا تکراری[۲۰] است و خطای قبلی موجود در سیستم تکثیر شده است. از جمله دلایل پرداختن به گزارش خطای تکراری می‌تواند: جستجوی ضعیف ردیاب‌های خطا، بی‌تجربگی کاربران و ارسال دوباره‌ی گزارش‌های خطا به طور اتفاقی باشد [۱۴]. گزارش‌های خطای تکراری در قالب واژه‌ی Duplicate طبقه‌بندی شده و به خطاهای اصلی[۲۱] پیوست می‌شوند. شناسایی گزارش خطاها برای جلوگیری از

اختصاص یک خطا به توسعه‌دهنده‌های متعدد، مورد نیاز است. اغلب گزارش تکراری شامل اطلاعات تکمیلی است که برای اصلاح و تعمیر خطا می‌تواند مفید واقع شود.

سیستم‌های نرم‌افزاری پیچیده نیاز به یک روش خودکار برای ردیابی خطا دارند. کاربران اغلب خطاهایی را که قبلاً گزارش شده را بررسی نمی‌کنند و گزارش خطای تکثیر با ارجاع به همان خطا صورت می‌گیرد و تخصیص آن به چندین توسعه‌دهنده باعث اتلاف زمان[۲۲] می‌شود. سیستم مراقبت نرم‌افزار قبل از اینکه گزارش خطاها را برای اصلاح به توسعه‌دهندگان بدهند باید به صورت دستی تعیین کنند که آیا گزارش خطاهای جدید از روی گزارش خطاهای قبلی تکثیر شده است یا نه؟ شناسایی خودکار گزارش‌های تکراری از میان هزاران گزارش خطا در مخزن خطا[۲۳] موجب افزایش بهره‌وری و صرفه‌جویی سیستم مراقبت نرم‌افزار می‌شود. این بهره‌وری سیستم مراقبت نرم‌افزار نتیجه‌ی کاهش میزان زمان صرف شده به منظور جستجوی گزارش خطای تکراری از گزارش‌های دریافتی است. مطالعات نشان می‌دهد که درصد گزارش خطای تکثیر می‌تواند به ٪۳۰- ۲۵ برسد. این روند به طور قابل ملاحظه‌ای می‌تواند روند تعمیر خطا و نرم‌افزار مربوطه را مختل کند [۹].

۲-۱- رکوردهای گزارش خطا

هر مجموعه داده شامل صدها و یا هزاران گزارش خطا است و هر گزارش خطا شامل رکوردهای اطلاعاتی است. هر رکورد بیانگر توضیحی در مورد خطای رخ داده در سیستم است که در بین تمام سیستم‌های ردیابی خطا از جمله Bugzilla مشترک هستند. در جدول ۲ برخی از اطلاعاتی که سیستم‌های ردیابی خطا در خود نگهداری می‌کنند به همراه تعاریف آن‌ها، آورده شده است. بر حسب نیاز، این سیستم‌ها حتی می‌توانند اطلاعات سفارشی را نیز به سیستم نرم‌افزاری اضافه کنند تا در آن داده‌ها از کاربر دریافت شده و ذخیره گردند. جهت تشخیص گزارش خطاهای تکراری از این اطلاعات استفاده می‌شود. هر کدام از این رکوردهای ذکر شده نوع و مقادیر خاص خود را دارند که در ادامه به بیان آن‌ها پرداخته‌ایم.

- رکوردهای شناسه‌ای[۲۴]: شناسه‌ها جهت یکتا نمودن گزارش خطاهای خطا و همچنین توسعه‌دهنگان استفاده می‌شوند. بدیهی است که شناسه‌ها ملاک مناسبی برای تشخیص تکراری بودن دو گزارش نیستند. در داده‌های بانک اطلاعاتی، داده Merge_id نشان‌دهنده گزارش خطای دیگری است که درباره‌ی همین خطا توضیحاتی دارد. به این ترتیب یک لیست پیوندی از گزارش خطاهای خطا می‌توان داشت که تکراری بودن آن‌ها مشخص است و قبلاً توسط انسان یا نرم‌افزار تشخیص داده شده و قطعی شده‌اند. مقدار داشتن این داده باعث می‌شود بر روی بخشی از بانک اطلاعات بتوان عملیات طبقه‌بندی[۲۵] را انجام داد و معیارهایی جهت یافتن گزارش خطاهای تکراری را پیدا کرد و دقت معیارها را با این داده تشخیص داد. در جدول مذکور رکوردهای Bug ID, Merge_id, CC, Assigned_to جزء رکوردهای شناسه‌ای هستند.
- رکوردهای رسته‌ای[۲۶]: این داده‌ها از نوع انتخابی و معمولاً اسمی هستند. مزیت این نوع داده‌ها این است که به نحوی می‌توانند به خوشه‌بندی[۲۷] داده‌ها کمک کنند و تا حدی حوزه جستجوی داده‌های تکراری را محدود نمایند و موجب تسریع عملیات جستجو شوند. از معایب این داده‌ها این است که چندان قابل اعتماد نیستند. چنانچه از خطای سهوی کاربر صرف نظر کنیم و فرض کنیم کاربران اطلاعات را به درستی وارد کرده‌اند، مشکل دیگر این است که گاهی کاربر به دلیل عدم تبحر ممکن است این اطلاعات را درست وارد نکند مثلاً اطلاع دقیقی از سیستم عامل، سخت‌افزار، نرم‌افزار یا نسخه آن نداشته باشد و موردی را اشتباه پر نماید. همچنین با فرض اینکه باز هم تمام کاربران خبره باشند و چنین اشتباهی ندارند، این امکان وجود دارد که کتابخانه‌های هسته‌ای یک شرکت در چندین محصول استفاده شده باشد و خطای آن کتابخانه‌ها در تمام محصولات مشاهده شود، بنابراین اطلاعات این بخش چندان هم قابل اعتماد نیستند. در جدول مذکور رکوردهای Product, Company, Component, Operating System, Hardware, Type, Version, Severity, Priority, Status, Resolution, جزء رکوردهای رسته‌ای هستند.
- رکوردهای متنی[۲۸]: این دسته از داده‌ها حاوی توضیحات و عناوینی است که کاربر به گزارش خود می‌دهد. این دسته مهم‌ترین اطلاعات جهت تشخیص تکراری بودن گزارشات خطا را دارد. برای تشخیص تکراری بودن دو گزارش باید بررسی نمود که تا چه اندازه محتوای دو گزارش شباهت دارند. در جدول مذکور رکوردهای (Issues) Title, Short_desc, Summary, Description جزء رکوردهای متنی هستند.
- رکوردهای فایلی: هر گزارش می‌تواند فایل‌های ضمیمه نیز داشته باشد. به عنوان مثال یک کاربر می‌تواند هنگام گزارش خطا، تصویری از صفحه نمایش رایانه خود را نیز ضمیمه کند یا هنگامی‌که نرم‌افزار از کار می‌افتد می‌تواند پشته فراخوانی[۲۹] خود را به عنوان یک فایل گزارشی متنی یا ساختاری ارسال کند.
- رکوردهای زمانی: هر گزارش می‌تواند تاریخ وقوع یا رفع شدن داشته باشد. داده‌های Open_date, Close_date از این دسته‌اند.

همانطور که گفته شد، هر کدام از این رکوردهای گزارش خطاها مقادیری را نیز به خود اختصاص داده‌اند که این مقادیر در جدول ۳ نشان داده شده است.

جدول ۱- مشخصات پروژه

حوزه	پروژه	اندازه کد	خطاها	طول عمر (سال)	درصد گزارش خطای تکراری
Bug tracker	Bugzilla	55K	12829	14	68%
IDE	Eclipse	6.5M	130095	7	19%
Browser	Epiphany	100K	10683	6	32%
E-mail client	Evolution	1M	72646	11	43%
Browser	Firefox	80K	60233	9	38%
Compiler	GCC	4.2M	35797	9	18%
E-mail client	Thunderbird	310K	19204	8	48%
Application server	Tomcat	200K	8293	8	8%

جدول ۲- رکوردهای اطلاعاتی گزارش خطاها

Title (Issues): عنوان خطای رخ داده در سیستم.	Bug Id: شماره‌ی خطای رخ داده در سیستم.
Resolution: چه اتفاقی برای این خطا رخ داده است.	Priority: چگونه خطا باید به زودی اصلاح شود.
Company: خطای رخ داده مربوط به چه سازمانی است.	Type: نوع خطای رخ داده در سیستم.
Severity: تأثیر خطا برروی سیستم نرم‌افزار مربوطه.	Component: مربوط به زیرسیستم محصولی که خطای آن گزارش شده.
Summary: توصیف کوتاهی از خطای رخ داده در سسیستم.	Product: نرم‌افزار ویژه‌ای که خطا به آن مربوط است.
Description: مطالب جزئی خطا مانند: چه خطایی و چگونگی رخداد آن.	Open_date: تاریخی که گزارش خطا نمایش داده است.
Merge_id: شناسه گزارش خطای تکثیر دیگر از گزارش خطای موجود.	Status: نشان‌دهنده‌ی وضعیت و حالت اخیر خطا است.
Close _date: تاریخ بسته شدن گزارش خطا.	CC: کاربرانی که علاقمند به پیشرفت این خطا هستند.
Assigned_to: شناسه‌ی توسعه‌دهنده‌ای که خطا به آن داده شده است.	Operating System: سیستم عاملی که خطا را گزارش داده است.
Version: نسخه‌ای از محصول که خطا در آن پیدا شده است.	Short_desc: خلاصه تک خطی از توصیف خطا را نمایش می‌دهد.
Attachment: فایل‌های پیوست نظیر فایل تصویری خطا یا گزارش خطا.	Hardware: سخت‌افزاری که سیستم عامل بر روی آن نصب شده است.

جدول ۳- مقادیر رکوردهای اطلاعاتی گزارش خطاها

انواع Resolution:	fixed, invalid, won't fix, duplicate, works for me, incomplete
انواع Status:	unconfirmed, new, assigned, reopened, ready, resolved, verified
انواع Severity:	trivial, minor, normal, major, critical, blocker
انواع Priority:	P1 تا P5
انواع Type:	defect, task, feature, enhancements

جدول ۴- انواع حل خطا

انواع حل	توصیف حل
نامعتبر	برطرف‌کننده خطا گزارش را نامعتبر[1] تلقی کرده است.
تعمیر نشده	برطرف‌کننده خطا تعیین می‌کند که خطا تعمیر[1] نخواهد شد.
تعمیر شده	نشان‌دهنده‌ی این است که خطا تعمیر شده[1] است.
تکراری	نشان‌دهنده‌ی این است که خطا، تکراری از خطای دیگر است.
در حال بررسی	برطرف‌کننده نمی‌تواند مشکل را حل کند. خطا قابل تولید مجدد نبوده.
حرکت کرده	نشان‌دهنده‌ی این است که گزارش خطا به مکانی دیگر منتقل شده است.

۲-۲- چرخه حیات گزارش خطا

خطاها از طریق یک سری از حالت‌های طول عمرشان گذر می‌کنند. حالت‌های مختلف یک گزارش خطا در Bugzilla در شکل ۱ آمده است. حالت‌ها بسته به روند توسعه و نیاز پروژه حذف یا اضافه می‌شوند.

به دلیل این‌که چرخه حیات Bugzilla شبیه به دیگر پروژه‌های منبع باز است و در واقع بقیه پروژه‌ها از Bugzilla استخراج شده‌اند، بنابراین ما به تشریح آن می‌پردازیم. Bugzilla یک ابزار ردیابی خطای مبتنی بر وب و توسط Mozilla توسعه یافته است. همانطور که در شکل ۱ نشان داده شده است، هنگامی که یک گزارش خطا توسط کاربر ارائه می‌شود، وضعیت آن به جدید[30] تغییر می‌کند. هنگامی که این گزارش به یک توسعه‌دهنده برای بررسی بیشتر اختصاص داده می‌شود، وضعیت آن به اختصاص داده شده[31] تغییر می‌کند. در نهایت، هنگامی که یک گزارش حل شود، وضعیت آن به حل شده[32] تغییر می‌کند. اگر در ادامه گزارش خطا هیچ موردی نداشت، برچسب وضعیت آن به بسته شده[33] تغییر می‌کند. همچنین اگر به هر دلیلی یک گزارش که خطا قبلاً حل شده و بعد از آن دوباره برای بررسی بیشتر باز شود، وضعیت آن به بازگشایی[34] تغییر می‌کند. بسته به ماهیت خطا روش‌های مختلفی برای حل خطا وجود دارد اما بسیاری از حالت‌های ممکن در آن‌ها مشابه است. انواع این حالت‌ها و توضیح در مورد آن‌ها در جدول ۴ آمده است.

۳- کارهای انجام شده

سیستم‌های ردیابی خطای استفاده شده در پروژه‌های نرم‌افزاری شامل خطاها و یا اشکالات نوشته شده توسط یک طیف گسترده‌ای از گزارش‌دهندگان خطا، با سطوح مختلف دانش در مورد سیستم تحت توسعه هستند. به طور معمول گزارش‌دهندگان، فاقد مهارت و زمان لازم برای جستجوی خطاهای مشابه تازه گزارش شده در سیستم ردیابی خطا هستند. پروژه‌های منبع باز به منظور انتساب گزارش خطا به توسعه‌دهندگان، سیستم‌های ردیابی خطا را با هم یکجا در نظر می‌گیرند. یکی از وظایف سیستم ردیابی خطا، شناسایی تکراری بودن گزارش خطای تازه وارد با یک گزارش خطای از پیش موجود در سیستم است. یک سیستم ردیابی خطا به منظور تشخیص گزارش خطای تکراری یا به حافظه درونی خود و یا به قابلیت‌های جستجو در مخزن خطا متکی است. هر دو روش برای سیستم ردیابی خطا وقت‌گیر است و همچنین ممکن است منجر به تشخیص نادرست تکراری بودن گزارش خطا شود.

علاوه بر این گزارش خطاهای تکراری لزوماً مضر نیستند و گاهی می‌توانند به وسیله‌ی ارائه اطلاعات اضافی به توسعه‌دهندگان برای بررسی خطای موجود مکمل یکدیگر باشند. این انگیزه نیاز به روش‌های خودکار یا نیمه خودکار برای تشخیص خطای تکراری دارد. با توجه به ضرورت فرآیند گزارش خطاهای تکراری به صورت خودکار بیشتر محققان در این زمینه مطالعه کرده‌اند. تقریباً همه مطالعات موجود در دامنه‌ی روش‌های بازیابی اطلاعات و تلاش برای بهبود آن‌ها است. تاکنون چندین روش برای بهبود تشخیص گزارش خطای تکراری و کمک به سیستم مراقبت نرم‌افزار پیشنهاد شده است که به‌طور کلی به دو روش دسته‌بندی می‌شوند. اولین روش فیلتر کردن خودکار گزارش‌های تکراری به منظور جلوگیری از رسیدن آن‌ها به توسعه‌دهندگان است. در این روش، فرض مهم این است که گزارش خطاها به صورت خودکار فیلتر شوند و از رسیدن خطاهای تکراری به توسعه‌دهندگان جلوگیری شود. بنابراین این رویکرد به دخالت هیچ سیستم مراقبت نرم‌افزاری نیاز ندارد و فرض می‌کنیم نفری که باید فیلتر کردن را انجام دهد، دقیق است. روش معناشناسی متنی و خوشه‌بندی گراف از این روش استفاده می‌کند. روش دوم با یک لیست پیشنهادی از بالاترین K گزارش خطای مشابه که

توسط کاربر نهایی اعلام شده سروکار دارد که به سیستم مراقبت نرم‌افزار اجازه می‌دهد گزارش خطای جدید وارد شده را با لیست پیشنهادی مقایسه کند. اگر گزارش خطای پیشنهادی موجود در لیست با گزارش خطای تازه وارد مطابقت داشت، سیستم مراقبت نرم‌افزار آن را به عنوان تکراری از گزارش خطای موجود با برچسب duplicate مشخص می‌کند و گزارش موجود به عنوان گزارش اصلی در نظر گرفته می‌شود. در این روش کیفیت لیست بسیار مهم است. روش‌های پردازش زبان طبیعی، پردازش زبان طبیعی به همراه اطلاعات زمان اجرا، معناشناسی کلمه و روش مدل افتراقی برای این هدف انجام می‌شوند. به‌طور کلی می‌توان کارهای انجام گرفته را در چهار حوزه دسته‌بندی کرد.

۳-۱- تکنیک‌های بازیابی اطلاعات

به منظور به دست آوردن اطلاعات مورد نیاز از یک مجموعه منابع اطلاعاتی که در طیف گسترده‌ای از زمینه‌های مختلف کاربرد دارد، استفاده می‌شود. این تکنیک‌ها برای حل مشکلات مهندسی نرم‌افزار نیز اعمال می‌شوند و در مرحله‌ی نگهداری و ارزیابی چرخه‌ی نرم‌افزار جای می‌گیرند. تحقیقات نشان داده IR دارای عملکرد مفیدی در زمینه‌ی تشخیص گزارش خطای تکراری است. مدل فضای برداری[۳۵] یکی از مهم‌ترین ابزارهای بازیابی اطلاعات است. این مدل یک نمایش ریاضی از اسناد متنی ارائه می‌دهد و معمولاً به منظور مقایسه پرس‌وجوهای متنی و اسناد مورد استفاده قرار می‌گیرد. یکی از روش‌های برجسته تشکیل بردار وزن‌دار [۳۶] متن به نام TF-IDF[۳۷] است. TF-IDF یک عامل وزنی است که نشان می‌دهد چه اندازه یک کلمه در متن یا مخزن اسناد مهم است.

۳-۲- تکنیک‌های ردیابی پشته

در این روش، برای هر گزارش خطای ورودی، دو تشابه متفاوت بین این گزارش و تمام گزارش‌های موجود محاسبه می‌شود. اولین معیار شباهت، تشابه مبتنی بر زبان طبیعی (NL-S) است که در آن خلاصه و توصیف گزارش خطاها به بردار وزن با استفاده از TF-IDF تبدیل شده و توسط معیار تشابه کسینوسی با همدیگر مقایسه می‌شوند. معیار دوم، تشابه براساس اطلاعات زمان اجرا [۳۸] (ES) که در آن از یک مدل فضای برداری براساس اطلاعات زمان اجرا برای محاسبه تشابه گزارش خطاها استفاده می‌شود. با این حال، در این اندازه گیری تشابه، فقط متدهایی که در طول اجرا استفاده شده‌اند، فراخوانی می‌شوند (بدون در نظر گرفتن اینکه هر متد چند بار فراخوانی شده است). در نهایت، ترکیبی از NLS و ES در طبقه‌بندی مشابه‌ترین گزارشات برای هر گزارش خطای ورودی خاص کمک می‌کند. نتیجه تجربی نشان می‌دهد که این روش قادر به تشخیص ٪۶۷ الی ٪۹۳ گزارش خطای تکراری در مخزن خطای فایرفاکس است [۸].

۳-۳- تکنیک‌های تشابه متنی و رسته‌ای

در مقاله‌ی [۹] تکنیکی پیشنهاد شده است که به طور خودکار، کار طبقه‌بندی و فیلتر کردن ورود گزارش خطاهای تکراری را انجام می‌دهد. یکی از اهداف این روش صرفه‌جویی در وقت triagers است. طبقه‌بندی روش ترکیبی از ویژگی‌های سطحی از گزارش خطاها (ویژگی‌های غیرمتنی مانند: severity, operating system)، معیارهای شباهت متن و الگوریتم‌های خوشه‌بندی گراف برای شناسایی گزارش خطاهای تکراری است. این طبقه‌بندی روی ویژگی‌های گزارش خطاها یک رگرسیون خطی اعمال می‌کند. هر سند با یک بردار وزن‌دار مشخص می‌شود که در آن وزن هر بردار با فرمول $W_i = 3 + 2\log_2(freq)$ محاسبه شده که در آن W_i وزن کلمه i در سند و freq تعداد تکرار کلمه i در سند است. شباهت متنی بین هر دو سند توسط معیار شباهت کسینوس زیر محاسبه می‌شود. نتیجه‌ی اندازه‌گیری شباهت اساس ایجاد یک گراف شباهت و یک الگوریتم خوشه‌بندی گراف است و در نهایت، ویژگی‌های سطحی جهت شناسایی گزارش‌های تکراری استفاده می‌شوند. ارزیابی بر روی یک زیرمجموعه از گزارش خطاهای موزیلا انجام شده و این رویکرد توانسته ٪۸ گزارش‌های تکراری را به صورت خودکار شناسایی و فیلتر کند.

$$\text{cosine}_{\text{sim}} = \frac{\sum_{i=1}^{n} C1_i \times C2_i}{\sqrt{\sum_{i=1}^{n}(C1_i)^2} \times \sqrt{\sum_{i=1}^{n}(C2_i)^2}} \quad (۱)$$

۳-۴- تکنیک‌های مدل موضوعی[۳۹]

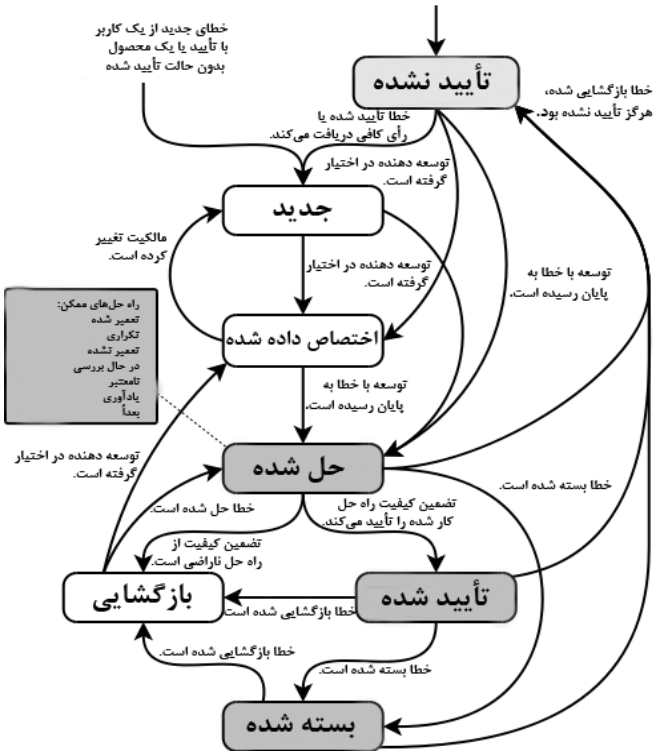
در مقاله‌ی [۱۱] یک تکنیک جدید به نام DBTM براساس تکنیک‌های IR و استخراج موضوع[۴۰] به منظور تشخیص گزارش خطاهای تکراری پیشنهاد شده است. برای مجموعه آموزش[۴۱] DBTM، گزارش خطاهای موجود در مخزن و اطلاعات تکراری آن‌ها استفاده شده است. برای پیش‌بینی گزارش خطاهای تکراری، DBTM روی گزارش خطاهای جدید اجرا می‌شود به این طریق که از پارامترهای آموزش به منظور برآورد شباهت بین گزارش خطای جدید و گزارش‌های موجود از نظر ویژگی‌های متنی و موضوعی استفاده می‌شود. در مقاله‌ی ذکر شده همچنین یک تکنیک مبتنی بر LDA به نام T-Model برای استخراج موضوعات از گزارش خطاها ارائه شده است. T-Model در مرحله‌ی آموزش از کلمات موجود در گزارش خطاها و تکرارهای مربوط در میان آن‌ها به منظور برآورد موضوعات، ویژگی‌های موضوع و ویژگی‌های محلی موضوع استفاده می‌کند. در مرحله‌ی پیش‌بینی، برای هر گزارش خطای جدید b_{new} با استفاده از پارامترهای آموزش، گروه تکراری G را که بیشترین شباهت را با b_{new} از نظر موضوع دارد، پیدا می‌کند. این شباهت با استفاده از فرمول زیر محاسبه می‌شود که در آن $\text{topicsim}(b_{new}, b_i)$ تشابه ویژگی موضوعی بین گزارش خطاهای b_i و b_{new} است:

$$\text{topicsim}(b_{new}, G) = \max_{b_i \in G}(\text{topicsim}(b_{new}, b_i)) \quad (۲)$$

در مقاله‌ی ذکر شده برای اندازه‌گیری شباهت متنی بین گزارش خطاها از روش BM25F استفاده شده است. برای ترکیب معیارهای موضوعی و متنی از یک تکنیک یادگیری ماشین به نام Ensemble Averaging استفاده می‌شود. محاسبه‌ی تابع y، ترکیب خطی از دو معیار ذکر شده (متنی و موضوعی) به صورت زیر است که در آن y_1 و y_2 معیارهای تشابه متنی و موضوعی، α_1 و α_2 پارامترهای کنترلی در فرآیند شناسایی گزارش خطای تکراری هستند. این رویکرد یک لیستی از بیشترین k گزارش خطای مشابه با گزارش خطای جدید را فراهم می‌کند. ارزیابی این روش روی مجموعه داده‌های اپن آفیس، موزیلا و اکلیپس نشان‌دهنده‌ی بهبود دقت ٪۲۰ است.

$$y = \alpha_1 \times y_1 + \alpha_2 \times y_2 \quad (۳)$$

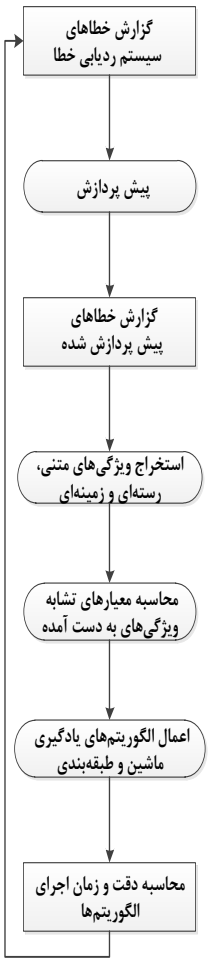
در آخر، با توجه به ضرورت فرآیند خودکار تشخیص گزارش خطاهای تکراری، تعداد زیادی از محققان در این زمینه مطالعه کرده‌اند. تقریباً همه مطالعات در دامنه‌ی روش‌های بازیابی اطلاعات هستند و هر یک سعی در بهبود این روش‌ها دارند. این مطالعات را می‌توان در کل به ۴ گروه حوزه‌بندی کرد. جدول ۵ دسته‌بندی روش‌های تشخیص گزارش خطاهای تکراری را نشان می‌دهد.



شکل ۱- چرخه حیات Bugzilla [۲]

جدول ۵- دسته‌بندی کارهای انجام شده

	عنوان مقاله	تکنیک مقایسه	تکنیک بازیابی	معیار ارزیابی
روش‌های اعمال تکنیک‌های انحصاری IR	تشخیص گزارش خطای تکثیر با استفاده از پردازش زبان طبیعی [۱].	اعمال مدل فضای برداری و معیار شباهت کسینوسی معیار تشابه با در نظر گرفتن زمان فریم‌ها	لیست تکرارهای کاندید	Recall Rate
	یک روش مدل متمایز برای بازیابی دقیق گزارش خطای تکراری [۳].	اعمال SVM به منظور پیش‌بینی تکرارها براساس معیارهای مقایسه متنی	لیست تکرارهای کاندید	Recall Rate
	مدل مبتنی بر معیارهای تشابه وزنی، روشی شیءگرا به منظور کاوش مجموعه داده‌های خطا برای تشخیص تشابه و خطاهای تکراری [۴].	اعمال مدل فضای برداری و معیار تشابه کسینوسی به منظور مشخص کردن تکرارها براساس یک آستانه‌ی خاص	فیلترکردن خودکار	Recall Rate, Precision
	تشخیص گزارش خطای تکراری با استفاده از ویژگی‌های مبتنی بر کاراکترهای N-gram [۶].	ساختن کاراکترهای N-gram توصیف و عنوان خطاها و مقایسه‌ی آنها براساس تعداد کارکترهای N-gram مشترک	لیست تکرارهای کاندید	Recall Rate
	تشخیص گزارش خطای تکراری [۷].	اعمال مدل فضای بردای، معیار تشابه کسینوسی و خوشه‌بندی به منظور تشخیص تکرارها براساس یک آستانه‌ی خاص	لیست تکرارهای کاندید	Recall rate, Precision
روش‌های مبتنی بر ردیابی پشته	روشی برای تشخیص گزارش خطای تکراری با استفاده از زبان طبیعی و اطلاعات زمان اجرا [۸].	مقایسه‌ی متنی گزارش‌های خطا با استفاده از TF-IDF و معیارهای تشابه کسینوسی و همچنین اجرای اطلاعات و ترکیب این معیارها	لیست تکرارهای کاندید	Recall Rate
روش‌های مبتنی بر تشابه متنی و رسته‌ای	تشخیص گزارش خطای تکراری خودکار برای سیستم‌های ردیابی [۹].	اعمال مدل فضای برداری، معیار تشابه کسینوسی با استفاده از ویژگی‌های ظاهری و خوشه‌بندی گزارش خطاها	لیست تکرارهای کاندید	Recall Rate and Area Under the ROC Curve (AUC)
	بازیابی دقیق‌تر گزارش خطای تکراری [۱۰].	اعمال یک مجموعه از ۷ مقایسه شامل BM25F و معیارهای تشابه رسته‌ای	لیست تکرارهای کاندید	Recall Rate and Mean Reciprocal Rank (MRR)
روش‌های مبتنی بر مدل موضوعی	تشخیص گزارش خطای تکراری با ترکیب اطلاعات بازگشتی و مدل موضوعی [۱۱].	اعمال BM25F و LDA براساس معیار تشابه استخراج موضوع و ترکیب معیارها با استفاده از Ensemble Averaging	لیست تکرارهای کاندید	Recall Rate



شکل ۲- جریان کار تشخیص گزارش‌های تکراری

۴- راهکار پیشنهادی

با وجود ازدیاد ابعاد ویژگی‌های به‌دست آمده از رکوردهای اطلاعاتی درون گزارش خطاها، بهتر است آن‌ها را کاهش داد. کاهش بعد ویژگی‌ها سبب می‌شود تا سرعت و کارایی تشخیص گزارش‌های تکراری افزایش یابد. دلیل استفاده از کاهش بعد این است که بسیاری از ویژگی‌ها ممکن است شامل اطلاعات اضافی و حشو باشند که چندان تأثیری در تشخیص تکراری بودن یا نبودن ندارند و در واقع ارزش اطلاعاتی آن‌ها ناچیز است، از طرفی ممکن است بردار یک ویژگی با بردار ویژگی دیگری هم‌راستا باشد و از این رو میزان همبستگی این ویژگی‌ها مشخص نباشد. با توجه به اینکه در مطالعات قبلی تنها استخراج ویژگی‌ها مختلف از گزارش خطاها به منظور دقت بیشتر حائز اهمیت بود و کمتر در مورد میزان اهمیت و با معنا بودن ویژگی‌ها بحث شده، نیاز هست که کاهش بعد در این زمینه بررسی شود.

در این قسمت بیان نوآوری پژوهش در زمینه‌ی ترکیب ویژگی‌های استخراج شده و کاهش ابعاد ویژگی‌ها می‌پردازیم. هدف اصلی کاهش ابعاد[۴۲]، ترکیب ویژگی‌های اصلی به‌طوری‌که بین کلاس‌ها تمایز وجود داشته باشد. به طور مثال هر ویژگی یک بعد را به خود اختصاص می‌دهد که با ترکیب دوتا از آن ویژگی‌ها، می‌تواند به یک بعد تقلیل یابد. با یک افزونه‌ی مناسب، می‌توان کاهش بعد را در ماتریس سند اعمال کرد به‌طوری‌که ساختار خوشه‌ای خود را حفظ کند. این کار با پیدا کردن یک تبدیل خطی انجام می‌شود و ترکیبی از ویژگی‌ها را ارائه می‌دهد. همچنین برخی از ویژگی‌های به‌دست‌آمده اطلاعات اضافه‌ای درون خود دارند به عبارتی ممکن است بردار ویژگی آن‌ها، هم راستا با بردار ویژگی‌های دیگر باشد، بنابراین باید میزان ارزش اطلاعاتی هر ویژگی بررسی گردد و در صورت امکان، ویژگی‌های کم اهمیت‌تر حذف گردند تا بتوان با این عمل، کاهش بعد انجام داد. بدین صورت با کاهش ویژگی‌ها، تعداد مقایسه‌ها و سرعت انجام محاسبات بهبود می‌یابد و فضای ذخیره‌سازی کمتری نیاز است. هدف از این روند بهبود کارایی سیستم‌های تشخیص گزارش خطاهای تکراری و همچنین تشخیص همبستگی بین ویژگی‌ها و شناسایی ویژگی‌های کم اهمیت است.

۴-۱- روش رایج در تحقیقات قبلی

در مطالعات انجام گرفته تاکنون سعی بر آن بوده تا با استخراج ویژگی‌های بیشتر، دقت تشخیص گزارش خطاهای تکراری را بهبود بدهند. در برخی مطالعات تنها ویژگی‌های متنی استخراج شده مورد بررسی قرار گرفته که مسلماً تنها با این معیار دقت خوبی به دست نمی‌آمد. سپس محققان با استخراج ویژگی‌های بیشتر از جمله ویژگی‌های رسته‌ای و زمینه‌ای دقت مورد نظر را بهبود بخشیده‌اند. جریان کار روش‌های قبلی در شکل ۲ آمده است.

همانطور که در شکل ۲ مشاهده می‌کنید، از گزارش خطاهای پیش پردازش شده، ویژگی‌های زیادی استخراج می‌شود که ممکن است با اهمیت و یا کم اهمیت باشند. سپس این ویژگی‌های استخراج شده ورودی الگوریتم‌های یادگیری ماشین هستند و روی آن‌ها طبقه‌بندی به منظور به دست آمدن دقت بهتر انجام می‌شود. این روش به دلیل استخراج تعداد زیادی از ویژگی‌های کم اهمیت زمان اجرای الگوریتم‌های طبقه‌بندی را طولانی می‌کند و ممکن است در دقت خللی ایجاد کند. به این دلیل در روش پیشنهادی سعی در حذف ویژگی‌های کم اهمیت با روش‌های کاهش بعد داریم که در ادامه توضیح داده می‌شود.

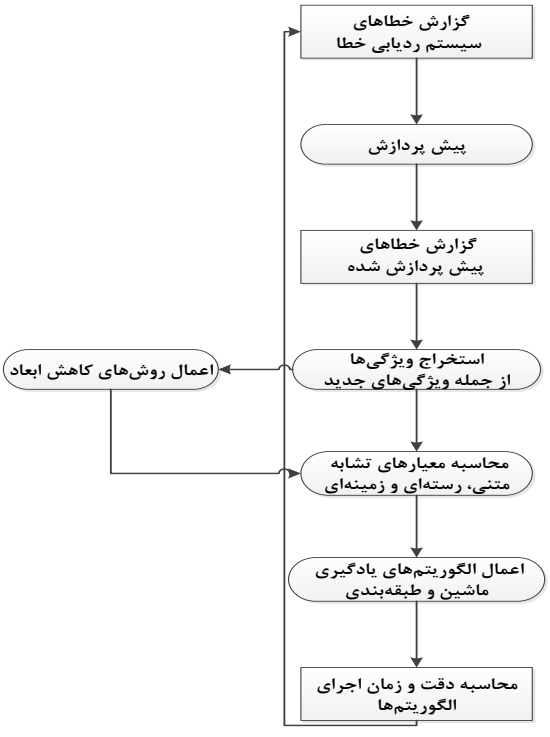
روش‌های کاهش داده برای به دست آوردن نمایش مختصر مجموعه داده‌ها که از لحاظ حجم خیلی کوچک‌تر و در عین حال صحت و جامعیت داده اصلی را داراست، به کار می‌روند. با این روش، کاوش داده‌ی کاهش یافته کارآمدتر و منجر به تولید همان نتایج اصلی می‌شود. کاهش بعد یکی از مراحل کاهش داده است که در برخی مواقع به عنوان یک مرحله پیش‌پردازش در ابتدای یک فرآیند داده‌کاوی انجام می‌شود. کاهش ابعاد از طرفی می‌تواند به عنوان یک کار انتخاب ویژگی[۴۳] در نظر گرفته شود. هنگامی‌که نشانه‌گذاری[۴۴] اسناد متنی، اغلب با داده‌های با بعد بسیار بالا مواجه است، ویژگی‌ها به آسانی قابل رسیدگی و دستکاری نیستند. بنابراین انتخاب ویژگی نقش مهمی را ایفا می‌کند. آستانه‌ی تکرار سند[۴۵]، دستیابی به کاهش ابعاد را با در نظر گرفتن حذف تکرارهای بسیار کم و یا زیاد فراهم می‌کند. اصطلاحاتی (کلمه‌ها، نشانه‌ها) که تقریباً در تمام اسناد یک مجموعه رخ می‌دهد هیچ‌گونه اطلاعات متمایزی به دست نمی‌دهند به عنوان مثال کلمه "از" به عنوان یک حرف ربط ممکن است در متن بارها تکرار شود اما برای به دست آوردن تشابه بین دو متن، اطلاعات مفیدی ارائه نمی‌کند. آستانه تکرار سند اغلب به عنوان اولین گام برای کاهش ابعاد استفاده می‌شود.

علاوه بر این، روش‌هایی مانند بهره اطلاعاتی[۴۶] ویژگی‌های مهم را قبل از انتخاب آن‌ها به منظور کاهش هزینه‌ی محاسبات به ما می‌دهد. چند روش کاهش بعد در ادامه به اختصار توضیح داده شده است [۱۷].

۱) تحلیل جداکننده‌ی خطی[۴۷]: LDA نتیجه‌ی جداکننده‌ی خطی Fisher's است که یک روش آماری، تشخیص الگو و یادگیری ماشین به حساب می‌آید. LDA به منظور مشخص کردن کلاس‌های اشیاء و یا رخدادها، یک ترکیب خطی

از ویژگی‌ها را به ما ارائه می‌کند. نتیجه‌ی ترکیب به عنوان یک طبقه‌بند خطی یا کاهش بعد، قبل از طبقه‌بندی بعدی مورد استفاده قرار می‌گیرد. LDA به تحلیل واریانس [۴۸] و تحلیل رگرسیون[۴۹] مربوط می‌شود و هدفش آشکارسازی یک متغیر وابسته به عنوان یک ترکیب خطی از ویژگی‌های دیگر است. تحلیل واریانس از متغیرهای مستقل مطلق و یک متغیر وابسته پیوسته استفاده می‌کند درحالی‌که LDA متغیر مطلق را توسط مقادیر متغیرهای مستقل پیوسته مشخص می‌کند. هدف اصلی LDA ترکیب ویژگی‌ها از داده‌های اصلی به‌طوری‌که بیشترین تأثیر را در جداسازی کلاس‌ها داشته باشد. این عمل توسط یک افزونه‌ی مناسب در سند ماتریس مانند انجام می‌گیرد. در این روش مجموعه‌ای از ویژگی‌ها به صورت بردار $\vec{x}$ از کلاس مشخص شده y در نظر گرفته می‌شوند. مجموعه‌ای از نمونه‌ها را مجموعه آموزش در نظر می‌گیریم و سپس طبقه‌بندی باید یک پیش‌بینی خوب از کلاس y ارائه دهد. روش LDA این مشکل را با فرض تابع چگالی احتمال شرطی $p(\vec{x}|y=0)$ و $p(\vec{x}|y=1)$ که هر دو توزیع نرمال با پارامترهای میانگین μ_0 و کوواریانس Σ_0 هستند، رفع می‌کند [۱۵].

۲) تحلیل مؤلفه‌ی اصلی[۵۰]: با توجه به توضیحات بالا، در این بخش، رویکرد تشخیص گزارش خطای تکراری تشریح می‌شود. در روش‌های قبلی زمان زیادی صرف استخراج ویژگی‌ها می‌شد و ممکن بود خیلی از آن ویژگی‌ها کم اهمیت باشند و در تشخیص تکرارها کمکی نکنند، به همین منظور در این پژوهش ابتدای ویژگی‌های جدیدی استخراج می‌شود و سپس با روش‌های کاهش بعد ویژگی‌های کم اهمیت حذف می‌شوند تا زمان اجرای الگوریتم‌های طبقه‌بندی کم شود و صحت تشخیص گزارش خطاهای تکراری بهبود یابد. اول از همه روش پیش‌پردازش گزارش خطاها بررسی می‌شود. سپس روش‌های اندازه‌گیری تشابه، به منظور مقایسه گزارش خطاها با استفاده از میانگین مشخصه‌های متنی، رسته‌ای و زمینه‌ای گزارش خطاها، بیان می‌شود. پس از آن، روش بازیابی گزارش خطای تکراری براساس معیارهای تشابه گزارش خطاها انجام می‌شود. شکل ۳ جریان کار روش فعلی را نشان می‌دهد.



شکل ۳- فلوچارت جریان کار روش پیشنهادی

۴-۱-۱- پیش‌پردازش

پس از استخراج گزارش خطاها، روش پیش‌پردازش متشکل از مراحل زیر بر روی آن‌ها اعمال می‌شود:

۱- اولین گام نشانه‌گذاری رکوردهای متنی گزارش خطاها و حذف کلمات توقف است. به عنوان مثال رکوردهای عنوان و توصیف گزارش خطا.

۲- دومین گام شامل سازمان‌دهی گزارش خطاها درون لیستی از دسته‌ها است. همه‌ی گزارش خطاها در همان دسته‌ای درج می‌شوند که گزارش خطای اصلی آن‌ها در آن دسته موجود است. در اینجا منظور از گزارش خطای اصلی Merge_ID است. گزارش خطایی که در اول لیست قرار دارد، به عنوان گزارش خطای اصلی دسته در نظر گرفته می‌شود. سپس گزارش خطاها به مجموعه‌ای از اشیاء با مشخصات زیر تبدیل می‌شوند: ID گزارش خطای اصلی دسته و شامل گزارش خطای اخیر است.

۴-۱-۲- استخراج ویژگی

مخازن خطا سیستم‌هایی هستند که مدیریت گزارش خطای ارسال شده توسط یک جامعه گسترده‌ای از کاربران را بر عهده دارند. معمولاً، کاربران دارای دانش‌ها، مهارت و سطح واژگان متفاوت برای تدوین و فرموله کردن گزارش خطا هستند. در نتیجه، یک سیستم ردیابی خطا پر از گزارش است که بسیاری از آن‌ها تکراری هستند، و سیستم مراقبت نرم‌افزار آن‌ها وقت‌گیر و مستعد خطا است. استخراج ویژگی از گزارش خطاها به‌طوری‌که بتوان مقدار تشابه دو گزارش خطا را در نظر گرفت در تشخیص کمک مؤثری دارد. با توجه به انواع داده‌های موجود در گزارش خطاهای کاملاً روشن است که این داده‌ها برای تعیین شباهت بین گزارش خطاها اهمیت زیادی دارند. به همین منظور روش‌های داده‌کاوی، متن‌کاوی، بازیابی اطلاعات[۵۱] و پردازش زبان طبیعی در این زمینه برای تعیین معیارهای تشابه[۵۲] در میان گزارش خطاها استفاده می‌شود. تاکنون شیوه‌های مختلفی جهت تشخیص گزارش‌های تکراری مبتنی بر روش‌های داده‌کاوی و علوم مرتبط با آن استفاده شده است تا ویژگی‌های[۵۳] جدیدی از گزارش‌ها استخراج شود و از طریق این ویژگی‌های جدید بتوان شباهت بین گزارش‌ها را بهتر یافت. با توجه به اینکه شیوه‌های مختلفی جهت استخراج ویژگی‌های متنی و ترکیبی وجود دارد، تاکنون چند دسته ویژگی از متون گزارش‌ها استخراج شده است. ویژگی‌های استخراج شده از مجموعه داده‌ها به چهار دسته‌ی مجزا تقسیم‌بندی می‌شوند: ویژگی‌های رسته‌ای[۵۴]، ویژگی‌های متنی[۵۵]، ویژگی‌های معنایی[۵۶] ویژگی‌های ساختاری[۵۷]. در زیر هر کدام از انواع ویژگی‌ها تفسیر شده است.

✓ معیارهای تشابه متنی

در این نوع ویژگی تعداد دقیق کلمات و واژه‌های یکسان با استفاده از معیارهای تشابه متنی و الگوریتم‌های خوشه‌بندی گراف[۵۸] به منظور تشخیص تکرارها محاسبه می‌شود. به منظور دقت بیشتر این ویژگی‌ها معمولاً حروف ربط و کلمات زائد و پرکاربرد حذف می‌شوند تا بدین ترتیب از ایجاد شباهت بی‌معنا بین گزارش خطاها جلوگیری شود. بعد از پیش‌پردازش تمام گزارش خطاهای داخل مخزن به مرحله استخراج ویژگی می‌رسیم. برای اندازه‌گیری شباهت متنی بین یک جفت گزارش خطا از روش BM25F معرفی شده توسط [۱۲] استفاده می‌شود. BM25F یک معیار تعریف شده‌ی مقایسه متن به منظور مقایسه‌ی اسناد است. در واقع یک تابع محاسبه‌ی تشابه متنی است. این مدل ارزیابی احتمالی شامل چندین متغیر است: محدوده‌ی تکرار در سند، طول سند، محدوده‌ی تکرار در پرس‌وجو، محدوده‌ی تکرار قبل از مرحله‌ی وزنی. این روش ساده‌تر و قابل تفسیرتر و

محاسبات را با سرعت و عملکرد بالاتری انجام می‌دهد. برای پرس‌وجوهای کوتاه که معمولاً فاقد کلمات تکراری هستند، طراحی شده است. به عنوان مثال، پرس‌وجوهای موتورهای جستجو معمولاً حداکثر شامل ده کلمه مجزا می‌باشند. با این حال، در زمینه‌ی بازیابی گزارش خطا تکراری هر پرس‌وجو یک گزارش خطا محسوب می‌شود. ساختار پرس‌وجو شامل خلاصه و توصیف (طولانی) است البته در برخی موارد می‌تواند بسیار طولانی باشد. بنابراین، اندازه‌گیری معیار شباهت متنی در این پژوهش براساس نسخه‌ی گرفته‌شده از BM25F به صورت رابطه ۴ تعریف می‌شود [۵]:

$$BM25F(d,q) = \sum_{t\in d\cap q} IDF(t) \times \frac{TF_D(d,t)}{K_1 + TF_D(d,t)} \times W_Q \quad (۴)$$

$$W_Q = \frac{(K_3+1)\times TF_Q(q,t)}{K_3 + TF_Q(q,t)} \quad (۵)$$

$$TF_Q(q,t) = \sum_{f=1}^{k} W_f \times occurences(q[f],t) \quad (۶)$$

$$TF_D(d,t) = \sum_{f=1}^{k} \frac{W_f \times occurences(d[f],t)}{1 - b_f + \frac{b_f \times length_f}{average-length_f}} \quad (۷)$$

$$IDF(t) = \log \frac{N}{N_d} \quad (۸)$$

رابطه ۴ برای هر اصطلاح مشترک t بین یک سند d و پرس‌وجوی q رابطه‌های ۵ و ۶ و ۷ و ۸ محاسبه می‌شود. در رابطه ۴ $TF_Q(q,t)$ تکرار وزن‌دار اصطلاح t در پرس‌وجوی q و $TF_D(d,t)$ تکرار وزن‌دار اصطلاح t در سند d و W_Q یک متغیر میانی برای تشابه دو سند با BM25F می‌باشد که هر کدام به صورت جداگانه تفسیر می‌شود.

- $TF_D(d,t)$ برای یک اصطلاح t در سند d که تراکم اهمیت t در هر رکورد متنی d هست. در رابطه ۷ برای هر رکورد f، wf_f رکورد وزن، $occurences\ (d[f],t)$ تعداد اصطلاح t در رکورد f، $length_f$ اندازه‌ی بسته‌ی $d[f]$، $average - length_f$ میانگین اندازه‌ی دسته‌ی $d[f]$ در تمام اسناد مجموعه است. پارامتر bf $(0 \le bf \le 1)$ که تعیین کننده مقیاس توسط رکورد طول: $bf = 1$ مربوط به طول کامل نرمال شده، $bf = 0$ مربوط به وزن اصطلاح که با طول نرمال نشده است.
- W_Q شامل وزن پرس‌وجوی محاسبه شده $TF_Q(q,t)$ است. پارامتر آزاد $k_3\ (k3 \ge 0)$ وزن اصطلاح پرس‌وجو را کنترل می‌کند. برای مثال: اگر $K_3 = 0$ ، آنگاه اصطلاح پرس‌وجو هیچ وزنی ندارد و عبارت W_Q همیشه برابر با یک می‌شود. $TF_Q(q,t)$ شامل تکرار اصطلاح t در پرس‌وجوی q است. در رابطه ۵ wf_f وزن رکورد متنی f در پرس‌وجوی q، $occurences\ (q[f],\ t)$ تکرار اصطلاح t در رکورد متنی f از q است.
- آخرین مورد $IDF(t)$ که یک رابطه معکوس با تکرار اصطلاح t در تمام اسناد موجود در مخزن دارد. در رابطه ۸ Nd تعداد اسناد شامل اصطلاح t و N تعداد کل اسناد است.

در کل می‌توان گفت ویژگی‌های متنی را با استفاده از تشابه[۵۹] TF-IDF استخراج می‌کنیم. فرآیند استخراج ویژگی جفت ویژگی‌های تکراری و غیر تکراری را استخراج می‌کند و سپس این ویژگی‌ها به عنوان ورودی به الگوریتم یادگیری SVM برای ساخت یک مدل در نظر گرفته می‌شوند. مهندسی ویژگی و استخراج برای پیدا کردن شباهت یا عدم شباهت بین دو گزارش انجام می‌شود. ویژگی‌های محدود شده تمایز بین دو مجموعه داده مجزا را سخت‌تر می‌کند: جفت‌های تکراری و جفت‌های غیر تکراری. از این رو مجموعه‌ای از ویژگی به اندازه کافی غنی مورد نیاز است تا بازیابی گزارش‌های تکراری دقیق‌تر باشد. تشابه متنی می‌تواند به عنوان ویژگی در نظر گرفته شود. دو نوع مقایسه برای اندازه‌گیری شباهت متنی توسط BM25F ارائه شده است [۱۲]. در روش پیشنهادی این دو مقایسه به هفت مقایسه تبدیل شده است.

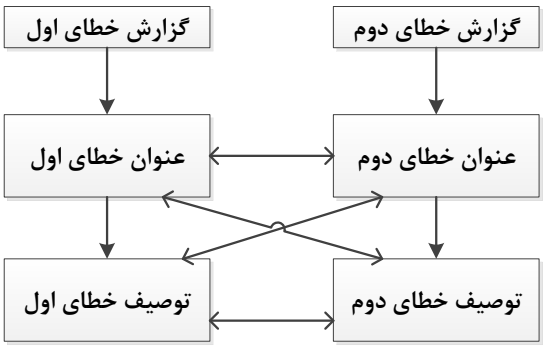
$$feature_1(d,q) = BM25_{ext}(d,q) \text{ of unigrams} \quad (۹)$$

$$feature_2(d,q) = BM25_{ext}(d,q) \text{ ofbigrams} \quad (۱۰)$$

اولین مقایسه تعریف شده در رابطه ۹ اندازه‌گیری شباهت متنی بین دو گزارش خطا روی رکوردهای عنوان و توصیفات گزارش خطا که توسط BM25F محاسبه شده است. دومین مقایسه مشابه به اولین مقایسه است با این تفاوت که ویژگی‌های عنوان و توصیف با bigrams نشان داده می‌شوند. یک unigrams متشکل از یک کلمه و یک bigrams متشکل از دو کلمه متوالی است. در unigrams مقایسه بین تک‌تک کلمه‌ها صورت می‌گیرد اما در bigrams مقایسه کلمات دوبه‌دو انجام می‌شود.

✓ ویژگی‌های جدید

با توجه به تشخیص بهتر گزارش خطاهای تکراری به سبب استخراج ویژگی‌های بیشتر سعی بر آن داشتیم تا بتوانیم ویژگی‌های بیشتری را استخراج کنیم. در مطالعات دیگر با استخراج تنها ویژگی‌های متنی (تشابه متن جفت گزارش خطا) و یا ویژگی‌های رسته‌ای (شامل تشابه اجزای تشکیل‌دهنده‌ی هر دو جفت گزارش خطای موجود در مخزن داده) و یا ویژگی‌های زمینه‌ای که با لیست کلماتی که از زمینه‌های نرم‌افزار به دست آورده‌اند قدرت تشخیص را بهتر کرده‌اند. استخراج ویژگی‌های قبلی تنها روی رکورد عنوان و توصیفات هر جفت گزارش خطا انجام می‌گرفت. در این پژوهش ترکیب رکوردهای عنوان و توصیف هر جفت گزارش خطا نیز محاسبه می‌شود. مجموعه داده‌های در نظر گرفته شده در این پژوهش فاقد رکورد خلاصه برای هر جفت گزارش خطا هستند و اگر نه همین کار برای رکورد خلاصه نیز انجام می‌گرفت. شکل ۴ شمای ترکیبات مختلف را نشان می‌دهد.



شکل ۴- شمای کلی ترکیبات رکوردهای عنوان و توصیف جفت گزارش خطا

ما این مقایسه را بین عنوان‌ها، توصیفات و ترکیبی از عنوان و توصیف گزارش خطاها اعمال کردیم. به جای مقایسه اول، سه مقایسه انجام شد که در رابطه ۱۱ آمده است.

$$bmf1Desc\ (d,q) = BM25(d,q) \text{ of unigrams} \quad (۱۱)$$

$$\text{bmf1TitleDesc}(d,q) = \text{BM25}(d,q) \text{ of unigrams} \quad (۱۲)$$

$$\text{bmf1Title}(d,q) = \text{BM25}(d,q) \text{ of unigrams} \quad (۱۳)$$

به جای مقایسه دوم نیز، سه مقایسه انجام شد. منظور از عدد یک در مقایسه‌ها unigrams و عدد دو bigrams است. این مقایسه‌ها در رابطه ۱۴ آمده است.

$$\text{Bmf2Desc}(d,q) = \text{BM25}(d,q) \text{ of bigrams} \quad (۱۴)$$

$$\text{bmf2TitleDesc}(d,q) = \text{BM25}(d,q) \text{ of bigrams} \quad (۱۵)$$

$$\text{bmf2Title}(d,q) = \text{BM25}(d,q) \text{ of bigrams} \quad (۱۶)$$

ترکیبات نمایش داده شده در شکل ۴ به شرح زیر است:

۱) عنوان گزارش خطای اول با عنوان گزارش خطای دوم
۲) توصیف گزارش خطای اول با توصیف گزارش خطای دوم
۳) عنوان گزارش خطای اول با توصیف گزارش خطای دوم
۴) توصیف گزارش خطای اول با عنوان گزارش خطای دوم
۵) ترکیب عنوان و توصیف گزارش خطای اول با ترکیب عنوان و توصیف گزارش خطای دوم
۶) عنوان گزارش خطای اول با ترکیب عنوان و توصیف گزارش خطای دوم
۷) توصیف گزارش خطای اول با ترکیب عنوان و توصیف گزارش خطای دوم
۸) عنوان گزارش خطای دوم با ترکیب عنوان و توصیف گزارش خطای اول
۹) توصیف گزارش خطای دوم با ترکیب عنوان و توصیف گزارش خطای اول

هر ترکیب به عنوان یک ویژگی منفرد محسوب می‌شود. بنابراین تعداد کل ویژگی‌های مختلف برابر با ۹ می‌شود. با سه دسته‌ی ذکر شده سه نوع IDF می‌توان محاسبه کرد. IDF که شامل مجموعه‌ای از عنوان‌هاست، IDF که شامل مجموعه‌ای از توصیفات است و همچنین IDF که شامل مجموعه‌ای از ترکیب عنوان و توصیف گزارش خطا است. هر دسته شامل ۹ ویژگی است بنابراین ۲۷=۳*۹. از طرفی کلمات دوبخشی (دو کلمه متوالی) نیز در نظر گرفته می‌شود در نتیجه ۵۴=۲*۲۷. به منظور داشتن یک آمار کلی از معیارهای کلمه‌ای در کل متن، از شاخص‌های میانگین، کمترین مقدار و بیشترین مقدار را نیز در نظر گرفتیم. به عبارت دیگر برای هر ویژگی استخراج شده سه مقدار میانگین، کمترین مقدار و بیشترین مقدار نیز محاسبه شده است و به عنوان یک ویژگی جدید در نظر گرفته شده است. یک مجموعه داده با ۱۶۲ ویژگی داریم. تعداد زیادی از این ویژگی‌ها فاقد اهمیت و تکراری است و در تشخیص هیچ کمکی اضافه‌ای نمی‌کنند و فقط زمان اجرای الگوریتم‌های طبقه‌بندی را زیاد می‌کنند و سرباری برای حافظه هستند.

✓ معیارهای تشابه رسته‌ای

ویژگی‌های رسته‌ای به صورت مستقیم از روی داده‌های رسته‌ای به دست می‌آیند. میزان شباهت بین دو مقدار یک داده رسته‌ای را می‌توان بر حسب یک ماتریس شباهت بیان نمود و یا داده‌های رسته‌ای را به نحوی با معنی به صورت عددی درآورد و از اختلاف عددی آن‌ها به عنوان یک معیار عدم شباهت استفاده نمود.

برای مقایسه ویژگی‌های رسته‌ای یک جفت گزارش خطا، معیار شباهت بین آن‌ها براساس رکوردهای اصلی آن‌ها اندازه‌گیری می‌شود. در جدول ۲ رکوردهای گزارش خطاها نام برده شد. به عنوان مثال: component, type, priority, product and version. با توجه به این‌که گزارش خطاهای تکراری ویژگی‌های رسته‌ای مشابه دارند و این انگیزه‌ای برای استفاده از ویژگی‌های رسته‌ای در تشخیص گزارش خطاهای تکراری به شمار می‌رود. رابطه ۱۷ معیارهای تشابه رسته‌ای را نشان می‌دهد. در اینجا d, q هر کدام اطلاعات مربوط به یک گزارش خطا هستند [۱۰].

$$\text{feature}_1(d,q) = \begin{cases} 1, & \text{if } d.\text{prod} = q.\text{prod} \\ 0, & \text{otherwise} \end{cases} \quad (۱۷)$$

$$\text{feature}_2(d,q) = \begin{cases} 1, & \text{if } d.\text{comp} = q.\text{comp} \\ 0, & \text{otherwise} \end{cases} \quad (۱۸)$$

$$\text{feature}_3(d,q) = \begin{cases} 1, & \text{if } d.\text{type} = q.\text{type} \\ 0, & \text{otherwise} \end{cases} \quad (۱۹)$$

$$\text{feature}_4(d,q) = \frac{1}{1+|d.\text{prio}-q.\text{prio}|} \quad (۲۰)$$

$$\text{feature}_5(d,q) = \frac{1}{1+|d.\text{vers}-q.\text{vers}|} \quad (۲۱)$$

در اولین رابطه مقایسه بین product گزارش خطاها است و برای مخزن خطای Android قابل اجرا نیست چون رکورد product برای همه گزارش خطاهای Android مشخص نشده است. بنابراین مقدار این ویژگی برای همه گزارش خطاهای Android صفر در نظر گرفته شده است. همچنین مطابق روش [۱۲] ویژگی version برای مخازن خطای Eclipse, Mozilla, and OpenOffice در نظر گرفته نشده است.

دومین مقایسه، مقایسه ویژگی‌های component گزارش خطاها است. Component یک گزارش خطا ممکن است به یک لایه معماری و یا یک ماژول خاص در یکی از لایه‌های معماری اختصاص داده شود. مقدار این مقایسه اگر دو گزارش خطا متعلق به Component یکسان باشند برابر یک است و در غیر این صورت به صفر تنظیم می‌شود.

سومین مقایسه، مقایسه بین type دو گزارش خطا است، به عنوان مثال در سیستم ردیابی خطای Android این مقدار مشخص می‌کند که آیا هر دو گزارش خطا هنوز نقص هستند یا به مرحله پیشرفت[۶۰] رسیده‌اند. مقدار این مقایسه اگر دو گزارش خطا متعلق به type یکسان باشند برابر یک است و در غیر این صورت به صفر تنظیم می‌شود.

دو مقایسه آخر، مقایسه بین priority و version دو گزارش خطا است. این اندازه‌گیری مقادیر بین ۰ و ۱ از جمله یک را داراست.

برای مؤثر بودن نتایج آزمایش‌ها از چهار ویژگی رسته‌ای دیگر مطرح شده در مقاله [۱۱] استفاده شده است. رابطه ۲۲ مقایسه‌ی این ویژگی‌ها را نشان می‌دهد.

$$\text{feature}_1(d,q) = \frac{1}{1-|d.\text{prio}-q.\text{prio}|} \quad (۲۲)$$

$$\text{feature}_2(d,q) = \frac{1}{1-|d.\text{vers}-q.\text{vers}|} \quad (۲۳)$$

$$\text{feature}_3(d,q) = \text{abs}(d.\text{open\_Date} - q.\text{Open\_Date}) \quad (۲۴)$$

$$\text{feature}_4(d,q) = \text{abs}(d.\text{bug}_{\text{id}} - q.\text{bug}_{\text{id}}) \quad (۲۵)$$

نتیجه این مقایسه‌ها یک مجموعه داده شامل تمام جفت گزارش خطاهایی است که اندازه‌گیری ویژگی‌های رسته‌ای روی آن‌ها انجام گرفته است.

✓ معیارهای تشابه زمینه‌ای

برخی از این روش‌ها علاوه بر متن، ویژگی‌های زمینه‌ای گزارش خطاها را نیز در نظر می‌گیرند. در اینجا رویکرد اندازه‌گیری تشابه زمینه‌ای در میان گزارش خطاها را توصیف می‌کنیم. این روش‌ها معتقد هستند که اندازه‌گیری شباهت‌های جدید می‌تواند به پیدا کردن گزارش خطاهای تکراری کمک کند. این ویژگی‌ها معمولاً سعی در به دست آوردن موضوع و مفهوم یک گزارش دارند. به عنوان مثال یک لغت‌نامه از واژه‌های مربوط به ویژگی‌های کیفی نرم‌افزار مانند: قابلیت اطمینان، کارایی، امنیت، قابلیت آزمون، قابلیت حمل تهیه می‌کنند و میزان کاربرد این واژگان را در متن گزارش بررسی می‌کنند و برحسب آن مشخص می‌شود که این گزارش خطا بیشتر با چه زمینه‌ای از نرم‌افزار مرتبط است. کلمات مهم مربوط به این زمینه در آمده است. با در نظر گرفتن لغت‌نامه ایجاد شده، می‌توان شباهت گزارش‌هایی که در یک زمینه هستند را به دست آورد. لیست کلمات[۶۱] مفهومی مختلف به منظور بررسی تأثیر زمینه‌های مختلف مهندسی نرم‌افزار بر روی دقت و تشخیص گزارش خطای تکراری استفاده می‌شود. تعدادی از این لیست کلمه‌ها در جدول ۶ زیر آمده است [۱۳].

جدول ۶- لیست کلمات

۱	کلمات معماری[۶۲]
۲	کلمات نیازمندی‌های غیر وظیفه‌مندی نرم‌افزار[۶۳]
۳	کلمات موضوعی استخراج شده به روش تخصیص دیری کله نهفته[۶۴]
۴	کلمات موضوعی استخراج شده به روش LDA و LDA برچسب‌دار[۶۵]
۵	کلمات فرهنگ لغت انگلیسی به صورت تصادفی
۶	مستندات Mozilla
۷	مستندات Android
۸	مستندات OpenOffice
۹	مستندات Eclipse
۱۰	مهندسی نرم‌افزار کلی

در این پژوهش به دلیل مؤثر بودن زمینه‌های مهندسی نرم‌افزار در بهبود تشخیص گزارش خطاهای تکراری، مجموعه داده‌های زمینه‌ای ارائه شده را مورد ارزیابی قرار داده‌ایم. این لیست کلمات در مقاله علیپور ارائه شده است [۵]. این مجموعه داده‌ها شامل لیستی از کلمات در زمینه‌ی مهندسی نرم‌افزار است. ابتدا این ویژگی‌ها استخراج شده و سپس مقادیر آن با سایر ویژگی‌های متنی و رسته‌ای گزارش خطاها مقایسه می‌شود.

- کلمات معماری: برای هر یک از مخازن خطا، مجموعه کلمات معماری که در آن هر لیست مربوط به یک لایه معماری است.

در این پژوهش به دلیل مؤثر بودن زمینه‌های مهندسی نرم‌افزار در بهبود تشخیص گزارش خطاهای تکراری، مجموعه داده‌های زمینه‌ای ارائه شده را مورد ارزیابی قرار داده‌ایم. این لیست کلمات در مقاله علیپور ارائه شده است [۵]. این مجموعه داده‌ها شامل لیستی از کلمات در زمینه‌ی مهندسی نرم‌افزار است. ابتدا این ویژگی‌ها استخراج شده و سپس مقادیر آن با سایر ویژگی‌های متنی و رسته‌ای گزارش خطاها مقایسه می‌شود.

- کلمات نیازمندی‌های غیر وظیفه‌مندی (NFR): یک روش خودکار.

استخراج موضوع برچسب‌گذاری شده براساس LDA و نظرات وارد شده در سیستم‌های کنترل منبع پیشنهاد کرده‌اند [۱۸]. این روش موضوعات برچسب‌گذاری شده را، از تعمیم طبقه‌بندی متقابل پروژه، متشکل از نیازهای غیر وظیفه‌مندی مانند: portability, maintainability, efficiency به دست می‌آورد. آن‌ها مجموعه داده‌ای از کلمات NFR نرم‌افزار در شش فهرست واژه با برچسب‌های بهره‌وری، عملکرد، نگهداری، قابلیت حمل، قابلیت اطمینان و قابلیت استفاده را ایجاد کرده‌اند.

- کلمات موضوعی LDA: LDA نشان‌دهنده ساختار موضوع و ارتباط موضوع در بین گزارش خطاها است. دو گزارش خطای تکراری باید با موضوع‌های فنی یکسان آدرس‌دهی شوند. انتخاب موضوع یک گزارش خطا تحت تأثیر موضوعی است که برای آن خطای مورد نظر گزارش شده است. مدل‌های تجزیه و تحلیل موضوعی LDA و LDA برچسب‌دار برای گزارش خطای Android به‌کار رفته‌اند [۱۹].
- کلمات تصادفی انگلیسی: برای بررسی تأثیر لیست کلمات زمینه‌ای در دقت تشخیص گزارش خطاهای تکراری، یک مجموعه‌ی تصادفی از کلمات فرهنگ لغت انگلیسی فرهنگ ایجاد کرده است [۵]. به عبارت دیگر، به این لیست‌ها، لیست مصنوعی[۶۶] گفته می‌شود. به عنوان مثال اگر داده‌های مخدوش به ویژگی‌های گزارش خطاها اضافه شود، می‌تواند وضعیت تکراری بودن خطا را بهبود بخشد حتی اگر داده‌های اضافه شده در یک زمینه‌ی معتبر نباشند.

در این بخش، با توجه به لیست کلماتی که در بالا بیان شد، اندازه‌گیری تشابه زمینه‌ای گزارش خطاها توصیف می‌شود. به عنوان مثال استفاده از زمینه‌ی کلمات معماری Android توضیح داده می‌شود. همانطور که قبلاً اشاره شد، این مجموعه کلمات معماری شامل پنج فهرست واژه با برچسب‌های برنامه‌های کاربردی، چارچوب، کتابخانه‌ها، زمان اجرا و هسته است. هر یک از این فهرست واژه‌ها به عنوان یک پرس‌وجو به حساب می‌آید و شباهت بین هر پرس‌وجو و هر گزارش خطای متنی با استفاده از BM25F محاسبه می‌شود. در مورد زمینه‌ی کلمات معماری، پنج مقایسه BM25F برای هر گزارش خطا وجود دارد، که نتیجه آن پنج ویژگی جدید برای هر گزارش خطا می‌باشد.

هر ستون شباهت زمینه‌ای بین گزارش خطا و هر یک از فهرست واژه کلمات معماری را نشان می‌دهد. به عنوان مثال، گزارش خطا با شماره ۳۷۶۴۴ به نظر می‌رسد بیشتر مربوط به برچسب زمان اجرا باشد تا برچسب‌های دیگر و بالعکس گزارش خطا با شماره ۳۷۶۴۲ بیشتر به زمینه‌ی برنامه‌های کاربردی مربوط است. این مقایسه‌ها برای تک‌تک مجموعه داده‌ها با هر یک از لیست کلمات انجام شده است. در پایان، پنج جدول زمینه‌ای مختلف برای هر مخزن خطای داریم.

✓ ترکیب معیارها

در این مرحله از فرآیند، تمام جدول ایجاد شده تاکنون برای جفت گزارش خطاها از جمله: جداول متنی، رسته‌ای و زمینه‌ای مربوط به هر مخزن خطا با هم ترکیب می‌شوند. در این مرحله، هدف تولید پنج جدول مختلف، هر یک شامل مقایسه دوبه‌دو گزارش خطا (یک) ویژگی‌های متنی (ب) ویژگی‌های رسته‌ای و (ج) یک مجموعه‌ای از ویژگی‌های زمینه‌ای. یکی از این جداول، مربوط به ویژگی‌های متنی است که برای مخزن خطای Android در نظر گرفته شده است. در نهایت با ترکیب جداول ۴ جدول کلی که هر کدام مربوط به یک مخزن خطا و شامل همه‌ی ویژگی‌ها است را تولید می‌کنیم. برای هر مخزن خطا ویژگی‌ها به همین ترتیب هستند که ابتدا شماره دو گزارش خطا، سپس ویژگی‌های محاسبه شده و در ستون آخر شباهت تکراری بودن یا نبودن دو گزارش خطا را نشان می‌دهد. ستون آخر یا همان معیار شباهت، مقدار فاصله بین دو گزارش خطا را اندازه‌گیری می‌کند. رابطه ۲۶ محاسبه این شباهت نشان داده شده است:

$$\text{cosine}_{\text{sim}} = \frac{\sum_{i=1}^{n} C1_i \times C2_i}{\sqrt{\sum_{i=1}^{n}(C1_i)^2} \times \sqrt{\sum_{i=1}^{n}(C2_i)^2}} \qquad (۲۶)$$

در رابطه ۲۶، n تعداد لیست کلمات زمینه‌ای که در مورد کلمات معماری برابر با پنج بود. C_{2i}, C_{1i}؛ برابر با i امین تعدادی ویژگی‌های زمنیه‌ای اضافه شده به هر گزارش خطا به ترتیب اولین و دومین جفت گزارش خطا است. ویژگی شباهت کسینوسی در جداول ما میزان شباهت هر جفت گزارش خطا را نشان می‌دهد. سپس مقدار کلاس همه جداول توسط طبقه‌بند یادگیری ماشین در مرحله بعدی پیش بینی شده است. به عبارت دیگر طبقه‌بندی تصمیم می‌گیرد که آیا این دو گزارش خطا تکراری از یکدیگر هستند یا خیر؟

✓ **کاهش بعد**

برای کاهش بعد، روش‌های زیادی وجود دارد. در اینجا ابتدا ویژگی‌های دارای همبستگی زیاد یا واریانس کم حذف گردیدند. از روش PCA برای کاهش بعد و انتخاب ویژگی‌ها استفاده شده است.

۴-۱-۳- پیش‌بینی

در این بخش، این‌که چگونه گزارش خطاهای تکراری بازیابی می‌شوند و چگونه این روش‌های بازیابی ارزیابی می‌شوند توصیف می‌شود. در رویکرد اول، با استفاده از طبقه‌بندهای عمومی به کاربرده شده در یادگیری ماشین مانند درخت تصمیم و بیزین ساده، به پیش‌بینی اینکه آیا دو گزارش خطای خاص تکراری هستند یا خیر می‌پردازیم (با توجه به معیار شباهت به‌دست آمده از آن‌ها). در روش دوم هر گزارش خطای ورودی با تمام گزارش‌های موجود در مخزن مقایسه می‌شود و لیستی از تکرارهای کاندید را به صورت مرتب شده برای سیستم مراقبت نرم‌افزار فراهم می‌کند. در نهایت سیستم مراقبت نرم‌افزار می‌تواند در مورد تکرارهای واقعی تصمیم بگیرد.

۴-۱-۴- طبقه‌بندی

در این بخش الگوریتم‌های طبقه‌بندی را روی مجموعه داده‌های مختلف با در نظر گرفتن معیارهای مقایسه اعمال می‌کنیم. این کار به منظور تصمیم‌گیری که آیا یک جفت گزارش خطا تکراری هستند یا خیر انجام می‌شود. هدف از استفاده یادگیری ماشین، برای تقویت تأثیر کار سیستم مراقبت نرم‌افزار به عنوان تشخیص‌دهنده خطاهای تکراری است. طبقه‌بند این امکان را دارد که چگونه به بهتر شدن تشخیص تکرارها کمک کند و ممکن است برای ساده‌تر شدن کاندیدهایی را به سیستم مراقبت نرم‌افزارها پیشنهاد دهد.

برای بازیابی گزارش خطاهای تکراری از الگوریتم‌های طبقه‌بندی یادگیری ماشین استفاده می‌شود. در هر آزمایش، یک جدول شامل جفت گزارش خطاها با ترکیب خاصی از معیارهای شباهت (به عنوان مثال ویژگی‌های متنی، رسته‌ای و زمینه‌ای) به طبقه‌بندی منتقل می‌شود. هر جدول به‌دست آمده برای هر مخزن خطای مجزا (که قبلاً شرح داده شده است) شامل همه ورودی‌های لازم برای طبقه‌بندی است. طبقه‌بند باید در مورد مقدار ستون کلاس هر مجموعه داده و هر جفت گزارش خطا تصمیم‌گیری کند. به عبارت دیگر، با توجه به هر جفت گزارش خطا، طبقه‌بندی باید تصمیم بگیرند اگر مقدار کلاس یا همان ستون آخر جداول dup است، جفت گزارش خطاها در دسته یکسان قرار گرفته‌اند و اگر non است، جفت گزارش خطاها در دسته یکسانی نیستند. برای جلوگیری از اتصالات زیاد در آموزش و ارزیابی، از روش اعتبارسنجی متقابل ۱۰ برابر ۶۷ برای طبقه‌بندی در RapidMiner استفاده شد. RapidMiner یک ابزار شناخته شده یادگیری ماشین و پیاده‌سازی شده در جاوا است. این ابزار تعداد زیادی از وظایف داده‌کاوی مانند: پردازش، خوشه‌بندی، طبقه‌بندی، رگرسیون، مصورسازی و انتخاب ویژگی را پشتیبانی می‌کند. الگوریتم‌های طبقه‌بندی یادگیری ماشین مورد استفاده در این کار به شرح زیر است. الگوریتم K-NN, Naïve Bayes, Decision Tree, Lib SVM, Random Forest, Fast Large margin (SVM), Linear Regression.

۴-۱-۵- معیارهای ارزیابی

به منظور ارزیابی عملکرد روش، معیارهای اندازه‌گیری دقت که از جمله معیارهای خروجی هستند در نظر گرفته شده است. چند معیار اندازه‌گیری دقت در فرآیند طبقه‌بندی مانند: صحت[۶۸]، دقت[۶۹]، سطح زیر نمودار ویژگی عملیاتی دریافت کننده[۷۰]، کاپا[۷۱] که در ادامه توضیح داده خواهد شد. دقت نسبت به نتایج درست (طبقه‌بندی درست dup و non) در میان همه جفت‌های طبقه‌بندی شده است. در واقع دقت برابر است با تعداد نسبی نمونه‌هایی که به درستی طبقه‌بندی شده‌اند یا به عبارت دیگر درصد پیش‌بینی درست. مقدار این پارامتر ارزیابی از نوع بولین است. رابطه ۲۷ دقت در زیر آمده است.

$$\text{accuracy} = \frac{|\text{true dup}| + |\text{true non}|}{|\text{true dup}| + |\text{false dup}| + |\text{true non}| + |\text{false non}|} \quad (۲۷)$$

false dup و True dup جفت گزارش خطاهایی هستند که درست یا غلط با طبقه‌بندی به عنوان dup مشخص شده‌اند، همچنین True "non" and false "non" جفت گزارش خطاهایی هستند که درست یا غلط به عنوان non (تکراری نبودن) مشخص شده‌اند.

صحت[۷۲]: تعداد نسبی نمونه‌هایی که به درستی طبقه‌بندی شده‌اند در میان تمام نمونه‌های طبقه‌بندی مثبت.

$$\text{Precision} = \frac{\text{Positives Correctly Classified}}{\text{Total Predicted Positives}} \quad (۲۸)$$

نرخ فراخوانی[۷۳]: این پارامتر نیز به نرخ درستی مثبت اشاره دارد.

$$\text{Recall} = \frac{\text{Positives Correctly Classified}}{\text{Total Positives}} \quad (۲۹)$$

$$\text{recall rate} = \frac{N_{\text{detected}}}{N_{\text{total}}} \quad (۳۰)$$

کاپا یک معیار آماری برای طبقه‌بندی و توافق ارزیابی بینابین است. تصور بر این است که کاپا یک معیار قوی‌تر نسبت به محاسبه درصد پیش‌بینی درست ساده است. به عنوان مثال کاپا نشان می‌دهد نرخ داده شده توسط ارزیاب چقدر همگن است. در کاپا پیش‌بینی درست همراه با شانس اتفاق می‌افتد. مقدار این پارامتر ارزیابی از نوع بولین است. رابطه کاپا به شرح زیر است:

$$\text{kappa} = \frac{P_r(a) - P_r(e)}{1 - P_r(e)} \quad (۳۱)$$

در رابطه ۳۱ $P_r(a)$ توافق مشاهده شده نسبی در میان ارزیاب‌ها، $P_r(e)$ احتمال فرضی شانس توافق با استفاده از داده‌ی مشاهده شده برای محاسبه احتمالات هر مشاهده کننده تصادفی هر دسته می‌باشد. اگر توافق برقرار و کامل باشد مقدار کاپا برابر یک می‌شود و در صورت عدم توافق در میان آن‌ها به غیر از آنچه که با شانس مورد انتظار است مقدار کاپا صفر می‌شود.

AUC سطح زیر منحنی ROC است. ROC یک روش مصورسازی[۷۴]، سازمان‌دهی و انتخاب طبقه‌بندها براساس عملکرد آن‌ها است. وزن نمونه‌های داده شده نیز در نظر گرفته می‌شوند. منحنی ROC با رسم کسری از تشخیص درست dup جدایی از همه dupهای تشخیص داده شده (نرخ مثبت درست) در مقابل کسری از تشخیص نادرست dup جدایی از همه nonهای تشخیص داده شده (نرخ مثبت نادرست) توسط طبقه‌بندی ایجاد می‌شود. AUC احتمالی است که یک طبقه‌بند، نمونه انتخاب تصادفی dup را بالاتر از انتخاب تصادفی یک non رتبه‌بندی می‌کند. (فرض بر این است که کلاس dup رتبه بالاتری نسبت به کلاس non دارد).

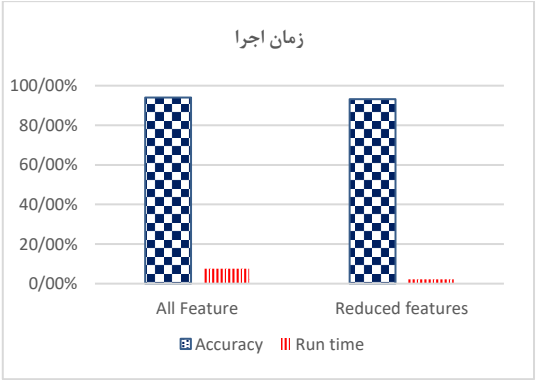
بسته به نوع سیستم، ویژگی‌های گزارش خطاها متفاوت می‌باشد اما به‌طور کلی این ویژگی‌ها مشابه هستند. همانطور که ملاحظه شد، رکوردهای اطلاعاتی گزارش خطاها در جدول ۲ آمده است با این تفاوت که مجموعه داده‌های استفاده شده در پژوهش شامل تمامی این رکوردها نیستند. مجموعه داده‌های تنها شامل رکوردهایی از جمله: description, Open_date,Title, status, component, priority, type, version, product, Bug Id, Merge_ID هستند ویژگی status می‌تواند مقادیر مختلف از جمله Duplicate را دارا باشد بدان معنی که گزارش خطا به عنوان یک گزارش تکراری توسط سیستم مراقبت نرم‌افزار شناخته شده است. توضیح عملکرد Merge_ID با مثالی تفسیر می‌شود. فرض کنید گزارش خطای A به عنوان یک تکرار از گزارش خطای B توسط سیستم مراقبت نرم‌افزار شناخته شده است، ویژگی Merge_ID به شماره گزارش خطای B (Bug_ID) اشاره دارد. در این حالت می‌گوییم B اصلی فوری[۷۵] A است.

۵- آزمایش ها و ارزیابی نتایج

پس از بررسی ویژگی‌های مسأله، در این بخش به ارائه نتایج شبیه سازی روش پیشنهادی، تحلیل و مقایسه‌ی نتایج آن با کارهای گذشته خواهیم پرداخت. استخراج تمام ویژگی‌ها از جمله ویژگی‌های متنی، رسته‌ای و زمینه‌ای از مجموعه ها داده ها با استفاده از زبان برنامه نویسی جاوا در نرم‌افزار NetBeans IDE8.0 پیاده سازی شد. ارزیابی نتایج براساس ویژگی‌های کاهش یافته نهایی با روش‌های کاهش بعد نظیر LDA و PCA، و همچنین زمان اجرای الگوریتم‌های طبقه‌بندی با حفظ دقت انجام گردید. به جهت بررسی تأثیر ویژگی‌های استخراج شده در حفظ دقت تشخیص گزارش خطاهای تکراری، الگوریتم‌های طبقه‌بندی بر روی ۴ سیستم ردیابی خطای Android, Eclipse, Mozilla, OpenOffice اعمال گردید.

شکل ۵ میانگین درصد زمان اجرای الگوریتم طبقه‌بندی Naïve Bayes در نظر گرفتن تمام ویژگی ها در مقایسه با ویژگی های کاهش یافته را نشان می دهد. همانطور که در شکل(۵) مشاهده می‌کنید در مرحله‌ای که ویژگی‌های بدون استفاده و زائد را کاهش داده‌ایم، زمان اجرای الگوریتم‌ها به نسبت قابل توجهی کاهش یافته است، بدون اینکه در درصد دقت تشخیص گزارش خطاهای تکراری خللی وارد کند. به عبارت دیگر با حفظ دقت، زمان اجرای الگوریتم را کاهش داده‌ایم. همانطور که قبلاً بیان شد، الگوریتم‌های طبقه‌بندی به خوبی می‌توانند گزارش خطاهای تکراری را از مخازن خطا شناسایی کنند. به منظور مقایسه‌ی بهتر و دقیق‌تر روش پیشنهادی، شبیه‌سازی بر روی ۸ الگوریتم طبقه‌بندی یادگیری ماشین دیگر نیز اعمال شده است.

برای بررسی روش پیشنهادی یک نمونه‌ای از گزارش خطاهای Android به نام Android-Samples را ایجاد کردیم. سپس یک آزمایش با استفاده از همه‌ی ویژگی‌ها و یک آزمایش با استفاده از ویژگی‌های کاهش یافته انجام شده و در نهایت به مقایسه میان آن دو پرداخته‌ایم. همانطور که در شکل ۶ مشاهده می‌کنید الگوریتم درخت تصمیم توانسته با زمان اجرای نسبتاً کمتری دقت خوبی را نسبت به سایر الگوریتم‌های طبقه‌بندی برجای بگذارد. به بیانی دیگر بالاترین بهبود مربوط به الگوریتم طبقه‌بندی درخت تصمیم است و درخت تصمیم کارایی بهتری از لحاظ زمان اجرای کمتر و حفظ دقت نسبت به سایر الگوریتم‌ها دارد.



شکل ۵- مقایسه میانگین زمان اجرای الگوریتم طبقه‌بندی Naïve Bayes

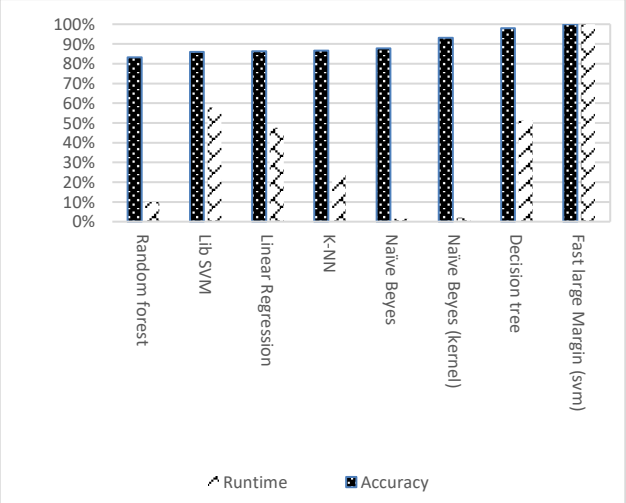
همانطور که قبلاً ذکر شد، برخی از ویژگی‌های جدید به هر مخزن گزارش خطا اضافه گردید. تعداد این ویژگی‌های در مقایسه با ویژگی‌های متنی، رسته‌ای و زمینه‌ای ارائه شده در کارهای قبلی خیلی کمتر است. در این بخش، تأثیر تعداد ویژگی‌های اضافه شده به گزارش خطاها را بر روی روند تکراری بودن تحلیل می‌کنیم. در اینجا پاسخ به سؤال زیر را دریافت می‌کنیم؟ جدول ۷ تعدادی از ویژگی‌های باقیمانده را نشان می‌دهد. تعداد ویژگی‌های اضافه شده در تحقیقات قبلی لزوماً در بهبود تشخیص گزارش خطاهای تکراری مؤثر نیست بلکه میزان اهمیت ویژگی مهم است. در روش پیشنهادی با کاهش این ویژگی‌ها به دقتی بالاتری هرچند ناچیز رسیدیم. همانطور که در جدول ۷ مشاهده می‌کنید ویژگی‌های استخراج‌شده با روش پیشنهادی بااهمیت‌تر از سایر ویژگی‌ها بوده است. بیشتر ویژگی‌های باقیمانده ویژگی‌های tf و idf هستند و ترکیب عنوان و توصیف گزارش خطاها (ویژگی‌های استخراج‌شده در روش پیشنهادی) است. در جدول ۷ منظور از عدد یک در ویژگی‌ها، کلمه یک بخشی و عدد دو کلمه دوبخشی یا دوکلمه متوالی است.

جدول ۷- تعدادی از ویژگی‌ها پس از کاهش بعد

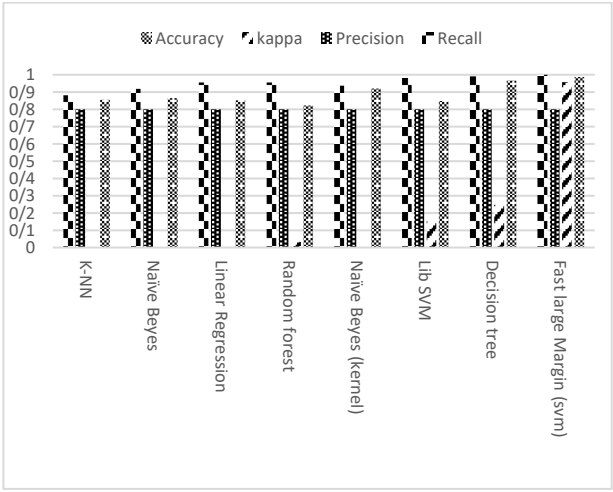
bmf2gT	bmf1gD
Idf2gCorpus TdocDdocDMN	Tf1gCorpus TdocDdocTMN
Idf2gCorpus TdocDdocDMX	Tf1gCorpus TdocDdocTMX
Idf2gCorpus TdocDdocTDMA	Tf1gCorpus TdocTDdocTDMN
Idf1gCorpus TdocDdocDMX	Tf2gCorpus TdocTdocTDMN
Idf1gCorpus TdocTDdocTMX	Tf2gCorpus TdocTDdocTDMN
Idf1gCorpus TdocTdocTDMX	
Idf2gCorpus DdocDdocDMX	
Idf2gCorpus DdocTDdocTDMX	

شکل ۷ سایر پارامترهایی کارایی از جمله کاپا، دقت، صحت و نرخ فراخوانی را برای ۸ الگوریتم طبقه‌بندی در مجموعه داده‌ی Android-Samples نشان می‌دهد. همانطور که مشاهده می‌شود کارایی الگوریتم Decision Tree و Fast

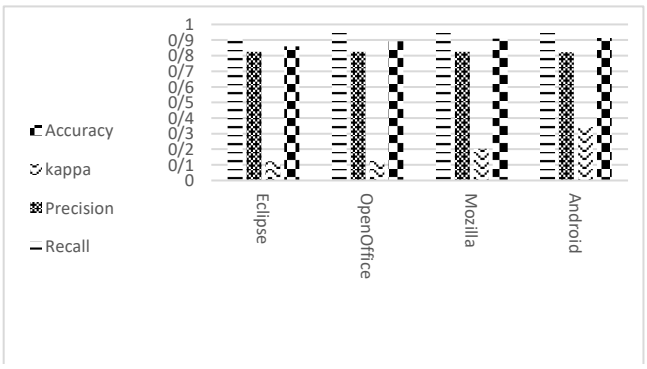
large Margin نسبت به بقیه الگوریتم‌ها بهتر است. شکل ۸ پارامترهای کارایی الگوریتم‌های طبقه‌بندی یادگیری ماشین را برای هر ۴ مخزن داده‌ی ذکر شده نشان می‌دهد. روش پیشنهادی در جدول ۸ با روش علیپور [۵] مقایسه شده است. براساس جدول ۸ مشاهده می‌کنیم که میزان دقت در هر مجموعه داده بهبود یافته است. همچنین میزان کاپا نسبت به روش علیپور به یک نزدیک‌تر است.



شکل ۶- مقایسه میانگین زمان اجرا و دقت الگوریتم‌های طبقه‌بندی



شکل ۷- معیارهای ارزیابی الگوریتم‌های طبقه‌بندی یادگیری ماشین



شکل ۸- پارامترهای کارایی الگوریتم‌های طبقه‌بندی یادگیری ماشین براساس مجموعه داده‌ها

جدول ۸- مقایسه روش پیشنهادی با علیپور [۵]

مجموعه داده	روش علیپور		روش پیشنهادی	
Alghrithm	Accuracy	Kappa	Accuracy	Kappa
Android	92.10%	0.7553	96.65%	0.998
Exlipse	96.75%	0.8968	97.89%	0.998
OpenOffice	93.67%	0.7925	96.26%	0.988
Mozilla	94.78%	0.8318	96.68%	0.988

روش پیشنهادی در جدول ۹ با روش دانش دامنه در مهندسی نرم‌افزار مقایسه شده است. براساس جدول ۹ مشاهده می‌کنیم که میزان دقت در هر مجموعه داده بهبود یافته است. همچنین میزان کاپا نسبت به روش دانش دامنه در مهندسی نرم‌افزار به یک نزدیک‌تر است.

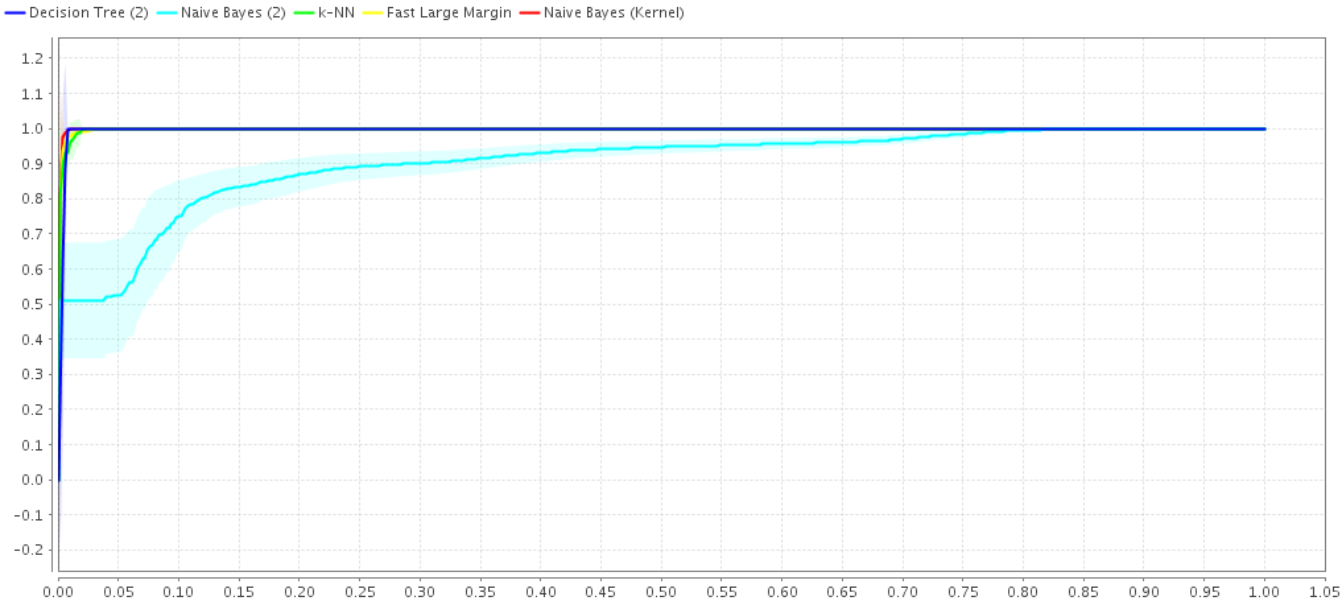
جدول ۹- مقایسه روش پیشنهادی با دانش دامنه در مهندسی نرم‌افزار [۲]

مجموعه داده	روش دانش دامنه مهندسی نرم‌افزار		روش پیشنهادی	
Alghrithm	Accuracy	Kappa	Accuracy	Kappa
Android	96.31%	0.790	98.65%	0.998
Exlipse	95.79%	0.754	97.89%	0.998
OpenOffice	96.22%	0.774	96.26%	0.988
Mozilla	95.49%	0.717	96.68%	0.988

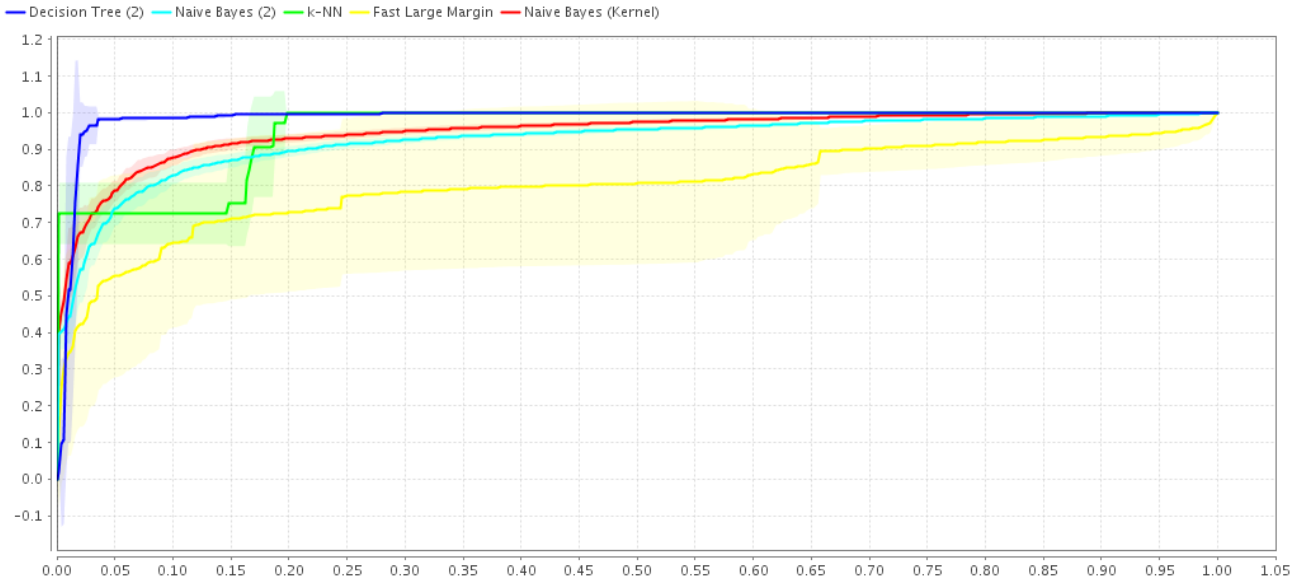
شکل‌های ۹ و ۱۰ مقایسه الگوریتم‌های طبقه‌بندی در ۲ مخزن خطاهای Android,Eclipse, را نشان می‌دهد. هرگاه سطح زیر منحنی ROC بیشتر باشد نشان دهنده بهتر بودن الگوریتم است. در هر ۲ مخزن خطا الگوریتم درخت تصمیم عملکرد بهتری دارد چون سطح زیر نمودار بیشتری دارد.

۶- نتیجه‌گیری

همانطور که گفته شد برای جلوگیری از وضعیت اختصاص یک گزارش خطا به چندین توسعه دهنده نیاز است که خطاهای تکراری حذف شوند. همچنین گزارش‌های تکراری حاوی اطلاعات تکمیلی است که می‌تواند برای تعمیر خطا مفید واقع شود. به‌طورکلی دو روش اصلی برای حل این مشکل ارائه شده است. روش اول، از رسیدن گزارش‌های تکراری به توسعه‌دهندگان با فیلتر کردن خودکار گزارش خطاها جلوگیری می‌کند و در روش دوم سیستم ردیابی خطا با لیستی از K بالاترین گزارش خطاهای مشابه سروکار دارد. این روش اجازه می‌دهد سیستم ردیابی خطا گزارش خطاهای دریافتی را با لیست ارائه شده مقایسه کند. در ابتدا فصل مفاهیم پیش‌زمینه‌های مختلف مرتبط در زمینه‌ی تشخیص گزارش خطاهای تکراری ارائه شده است. در این پژوهش، نخست مجموعه داده‌ی اصلی که همه‌ی ویژگی‌های متنی، رسته‌ای و زمینه‌ای را شامل می‌شد، با الگوریتم‌های طبقه‌بندی پیاده‌سازی و نتایج مورد بررسی قرار گرفت. در ادامه پیاده‌سازی دیگری با ویژگی‌های کاهش یافته انجام شد و دقت و زمان اجرا در هر دو حالت بررسی و مقایسه شد. نتایج نشان داد که با کاهش ویژگی‌ها نه‌تنها زمان اجرا کاهش یافته بلکه ویژگی‌های جدید باعث بهبود دقت در حدود ۱ الی ۷٪ شده است. ایده و نوآوری این پژوهش در ترکیب‌های مختلف داده‌های مجموعه خطاها بوده که باعث دقت بیشتری در تشخیص گزارش خطاها شد. کاهش بعد نیز نوآوری دیگری بود که تأثیر مثبت ویژگی‌های کاهش یافته را نسبت به ویژگی‌های اولیه از لحاظ زمان اجرا و دقت الگوریتم‌های طبقه‌بندی نشان داد. نتایج نموداری نشان داده است که می‌توان با حفظ دقت، زمان اجرای الگوریتم‌های طبقه‌بندی یادگیری ماشین را با کاهش ویژگی‌های کم اهمیت، کاهش داد. الگوریتم درخت تصمیم در میان بقیه الگوریتم‌های طبقه‌بندی در این آزمایش دقت بهتر و زمان اجرای کمتری را از خود نشان داد.



شکل ۹- نمودار ROC برای مخزن خطای Android



شکل ۱۰- نمودار ROC برای مخزن خطای Eclipse

مراجع

[1] J. Sutherland, "Business objects in corporate information systems," *ACM Computing Surveys (CSUR)*, vol. 27, pp. 274-276, 1995.

[2] K. Aggarwal, T. Rutgers, F. Timbers, A. Hindle, R. Greiner, and E. Stroulia, "Detecting duplicate bug reports with software engineering domain knowledge," in *IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2015, pp. 211-220.

[3] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful… really?," in *IEEE International Conference on Software Maintenance (ICSM)*, 2008, pp. 337-345.

[4] J. Anvik, L. Hiew, and G. C. Murphy, "Coping with an open bug repository," in *Proceedings of the OOPSLA workshop on Eclipse technology eXchange*, 2005, pp. 35-39.

[5] A. Alipour, A. Hindle, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, 2013, pp. 183-192.

[6] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," *Proceedings of*

the 26th IEEE/ACM International Conference on Automated Software Engineering, 2011, pp. 253-262.

[7] A. Sureka, and P. Jalote, "Detecting duplicate bug report using character n-gram-based features," in *17th Asia Pacific Software Engineering Conference (APSEC)*, 2010, pp. 366-374.

[8] M. W. Berry, and M. Castellanos, "Survey of text mining," *Computing Reviews*, vol. 45, p. 548, 2004.

[9] S. Banerjee, B. Cukic, and D. Adjeroh, "Automated duplicate bug report classification using subsequence matching," in *IEEE 14th International Symposium on High-Assurance Systems Engineering (HASE)*, 2012, pp. 74-81.

[10] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in *Proceedings of the 30th international conference on Software engineering*, 2008, pp. 461-470.

[11] A. Lazar, S. Ritchey, and B. Sharif, "Improving the accuracy of duplicate bug report detection using textual similarity measures," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 308-311.

[12] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2011, pp. 253-262.

[13] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, "Duplicate bug report detection with a combination of information retrieval and topic modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2012, pp. 70-79.

[14] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in *29th International Conference on Software Engineering (ICSE)*, 2007, pp. 499-510.

[15] N. Jalbert, and W. Weimer, "Automated duplicate detection for bug tracking systems," in *IEEE International Conference on Dependable Systems and Networks (DSN) With FTCS and DCC*, 2008, pp. 52-61.

[16] T. Zimmermann, R. Premraj, J. Sillito, and S. Breu, "Improving bug tracking systems," in *ICSE Companion*, 2009, pp. 247-250.

[17] F. Šarić, G. Glavaš, M. Karan, J. Šnajder, and B. D. Bašić, "Takelab: Systems for measuring semantic text similarity," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, 2012, pp. 441-448.

[18] N. K. Nagwani, and P. Singh, "Weight similarity measurement model based, object oriented approach for bug databases mining to detect similar and duplicate bugs," in *Proceedings of the International Conference on Advances in Computing, Communication and Control*, 2009, pp. 202-207.

[19] K. Vijayakumar, and V. Bhuvaneswari, "How much effort needed to fix the bug? A data mining approach for effort estimation and analysing of bug report attributes in Firefox," *International Conference on Intelligent Computing Applications (ICICA)*, 2014, pp. 335-339.

زهرا امین‌الرعایائی تحصیلات تکمیلی خود را در رشته مهندسی کامپیوتر گرایش نرم‌افزار به اتمام رسانده و اکنون فارغ التحصیل مقطع کارشناسی‌ارشد و پژوهشگر در حوزه‌ی داده کاوی است. تحقیقات مورد علاقه‌ی نامبرده مهندسی نرم‌افزار، داده کاوی و صحت و دقت گزارش خطاهای تکراری نرم‌افزار می‌باشد.

آدرس پست‌الکترونیکی ایشان عبارت است از:

z.aminoroaya@naeini.ac.ir

بهزاد سلیمانی نیسیانی دانشجوی سال پنجم دکتری در رشته مهندسی نرم‌افزار کامپیوتر است. حوزه تحقیقاتی که تا کنون در آنها پژوهش انجام داده و مقاله نیز به چاپ رسانده است بیشتر شامل الگوریتم‌های داده کاوی در زمینه قواعد انجمنی، طبقه‌بندی، سیستم‌های پیشنهاد دهنده بوده است. از جمله زمینه‌های تحقیقاتی دیگر ایشان پیاده‌سازی الگوریتم‌های مسیریابی ربات‌های هوشمند و خودمختار، محاسبات توزیع شده بوده است. وی هم‌اکنون مدرس دانشگاه کاشان و دانشگاه آزاد اسلامی واحد اصفهان (خوراسگان) است که بر روی الگوریتم‌های متن کاوی و کشف گزارش‌های متنی تکراری در پروژه دکترای خود کار می‌کند.

آدرس پست‌الکترونیکی ایشان عبارت است از:

b.soleimani@Grad.Kashanu.ac.ir

محمد حسین ندیمی استادیار و رئیس مرکز تحقیقاتی مه داده در دانشگاه ازاد اسلامی واحد نجف‌آباد است. موضوعات پژوهشی مورد علاقه‌ی ایشان داده کاوی، کاوش شبکه‌های اجتماعی، سیستم‌های هوشمند و کاوش داده‌های پزشکی می‌باشد.

آدرس پست‌الکترونیکی ایشان عبارت است از:

nadimi@iaun.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۱۳۹۵/۱۱/۲۳

تاریخ اصلاح: ۱۳۹۶/۰۸/۲۰

تاریخ قبول شدن: ۱۳۹۷/۰۶/۰۹

نویسنده مرتبط: زهرا امین‌الرعایائی، موسسه آموزش عالی علامه نائینی، نائین، اصفهان، ایران.

[1]Web Browser
[2]Bug Reports
[3]Triagers

[4]Defect
[5]Issue
[6]Development
[7]Testing
[8]Maintenance
[9]Meta Data
[10]Spam
[11]Error
[12]Flaw
[13]Fault
[14]Debugging
[15]Title
[16]Severity
[17]Bug-Report Triage
[18]Bug-Report Duplication
[19]Quality Assurance
[20]Duplicate
[21]Master
[22]Time Waste
[23]Bug Repository
[24]Identifier
[25]Classification
[26]Categorize
[27]Clustering
[28]Textual
[29]Call Stack
[30]New
[31]Assigned
[32]Resolved
[33]Closed
[34]Reopened
[35]Vector Space Model (VSM)
[36]Weight-Vector
[37]Term Frequency-Inverse Document Frequency
[38]Execution Information
[39]Topic Model
[40]Topic Extraction
[41]Train Set
[42]Dimensionality Reduction
[43]Feature Selection
[44]Tokenizing
[45]Document Frequency Thresholding
[46]Information Gain
[47]Linear Discriminant Analysis (LDA)
[48]Analysis of Variance (ANOVA)
[49]Regression Analysis
[50]Principal Component Analysis (PCA)
[51]Information Retrieval (IR)
[52]Similarity Measurements
[53]Features
[54]Categorical Features
[55]Textual Features
[56]Semantic Features
[57]Structural Features
[58]Graph-Clustering
[59]Term Frequency-Inverse Document Frequency
[60]Enhancements
[61]Words List
[62]Architectural Words
[63]Software Non-Functional Requirements Words (NFR)
[64]Latent Dirichlet Allocation (LDA)
[65]Labeled-LDA
[66]Artificial Context
[67]10-Fold Cross Validation
[68]Accuracy
[69]Precision
[70]Area Under Receiver Operation Characteristic (ROC)
[71]Kappa
[72]Precision
[73]Recall
[74]Visualizing
[75]Immediate Master