انتساب خطاهای شناسایی‌شده در نظرات کاربران در مورد برنامک‌های همراه به توسعه‌دهندگان

مریم یونسی، عباس حیدرنوری و فاطمه قنادی

چکیده: نظراتی که کاربران در فروشگاه‌های برنامک‌های همراه می‌نویسند و خطای برنامک‌ها را گزارش می‌کنند، می‌تواند در بهبود کیفیت نرم‌افزارها تأثیر به‌سزایی داشته باشد. بنابراین در این پژوهش رویکردی بر اساس نظرات کاربران برای انتساب خطای برنامه به توسعه‌دهندگان برنامک‌ها بیان خواهد شد. این رویکرد با استفاده از داده‌های کامیت‌های برنامه تاریخچه‌ای از عملکرد توسعه‌دهندگان به دست می‌آورد و همچنین با استفاده از ایراداتی که توسعه‌دهندگان از قبل در برنامه رفع کرده‌اند در مورد سوابق آنها در رفع خطاهای برنامه اطلاعاتی کسب می‌کند. سپس با استفاده از ترکیب این دو معیار به هر توسعه‌دهنده آن نرم‌افزار برای رسیدگی به هر نظر امتیازی اختصاص می‌دهد تا فهرستی از توسعه‌دهندگان ارائه کند که به ترتیب اولویت برای رسیدگی به نظر مناسب هستند. ارزیابی این پژوهش از جنبه‌های مختلف در نهایت نشان می‌دهد که روش پیشنهادی با دقت ۷۴٪ قادر به شناخت توسعه‌دهنده مناسب برای رسیدگی به نظرات خواهد بود. هدف این پژوهش یک موضوع جدید است که پژوهش دیگری حول آن انجام نگرفته و صرفاً باقی پژوهش‌ها راجع به دسته‌بندی نظرات کاربران بوده‌اند. بنابراین دقت ارزیابی این پژوهش نشان می‌دهد که انتساب اتوماتیک خطاهایی که در نظرات کاربران ذکر شده‌اند می‌تواند مفید واقع شود تا فرایند شناسایی و حل خطا بهبود یابد.

***کلیدواژه:* انتساب خطا، نگهداری نرم‌افزار، یادگیری ماشین، نظرات کاربران.**

۱- مقدمه

فروشگاه‌های برنامک‌های کاربردی مانند گوگل‌پلی[1] و اپ‌استور اپل[2] ارائه‌دهنده انواع برنامک‌ها و سرویس‌ها هستند که تعداد زیادی از کاربران از آنها بهره می‌برند و افزایش محبوبیت گوشی‌های هوشمند این فروشگاه‌ها را به مخازن عظیم نرم‌افزاری تبدیل کرده است. کاربران این برنامک‌ها، تجاربشان را در قالب نظراتی که برای هر محصول قرار می‌دهند به توسعه‌دهندگان نرم‌افزار منتقل می‌کنند تا موجب ارتقای کیفیت این محصولات شوند. تحقیقات نشان داده‌اند که نظرات کاربران حاوی اطلاعات مفیدی است که توسعه‌دهندگان و صاحبین نرم‌افزارها می‌توانند به خوبی از آنها بهره ببرند [۱]. در واقع اهمیت‌دادن به نظرات و درخواست‌های کاربران و جلب رضایت آنها می‌تواند به افزایش امتیاز برنامک کمک کند [۲]. به عنوان مثال کاربران ممکن است که درخواست یک ویژگی جدید در برنامک را در نظرات خود داشته باشند یا ممکن است نظرات حاوی گزارش خطاهایی باشد که در تجربه کاربری با آنها برخورد کرده‌اند. همچنین این منبع عظیم اطلاعات شامل تعداد زیادی از نظرات است که می‌تواند مشکلات مختلفی که در دستگاه‌های مختلف، سیستم‌های عامل گوناگون، اندازه‌های صفحه نمایش و شرایط خاصی از شبکه رخ می‌دهد، به توسعه‌دهندگان معرفی کنند که حتی ممکن است نتوانند دوباره آن شرایط را حین فعالیت‌های توسعه و آزمون نرم‌افزار به وجود آورند [۲]. بنابراین با مطالعه نظرات کاربران، افراد تیم توسعه نرم‌افزار تشویق به این می‌شوند که کیفیت نرم‌افزار را به عنوان مثال از طریق رفع خطاها یا اضافه‌کردن ویژگی‌های درخواست‌شده بهبود بخشند. با این وجود برای استخراج این اطلاعات اگر توسعه‌دهندگان بخواهند که به صورت دستی نظرات کاربران را مطالعه کنند و از محتوای آن آگاه شوند، این کار برای برنامه‌های کاربردی محبوب که روزانه هزاران نظر از کاربرانشان دریافت می‌کنند انجام‌پذیر نیست. بنابراین نیاز به روش‌هایی برای استخراج خودکار اطلاعات ارزشمند از نظرات کاربران احساس می‌شود. از مشکلاتی که در خودکارسازی استخراج اطلاعات از نظرات کاربران با آن مواجه هستیم می‌توان به عامیانه‌بودن جملات، وجود کلمات مخفف و همچنین ساختار نامشخص و غیر رسمی نظرات کاربران اشاره کرد.

تا کنون پژوهش‌های متعددی [۱] و [۳] تا [۹] در این حوزه انجام گرفته که هر یک سعی در درک نظرات کاربران و دسته‌بندی و اولویت‌دهی به آنها داشته‌اند. توسعه‌دهندگان پس از استفاده از این روش‌ها و به دست آوردن اطلاعات لازم راجع به خطاها و درخواست‌های ویژگی جدید، نیاز به این دارند که مناسب‌ترین توسعه‌دهنده را که تجربه کافی در زمینه رفع خطای صورت‌گرفته داشته باشد انتخاب کنند تا به رفع آن ایراد بپردازد و رضایت کاربران را به دست آورد. اما مشکلی که در این زمینه وجود دارد این است که اگر حجم پروژه و تعداد برنامه‌نویسان زیاد باشد این فرایند انتساب خطا سخت می‌شود. تا کنون روش‌ها و مدل‌های مختلفی برای انتساب خطا پیشنهاد شده است ولی هیچ یک در رابطه با انتساب خطاهایی که از نظرات کاربران استخراج می‌شوند، نبوده‌اند.

بنابراین این نیاز احساس می‌شود که به خطاهایی که از نظرات کاربران به دست می‌آوریم، به طور خودکار توسعه‌دهنده مناسبی را نسبت دهیم که به رفع آن ایراد بپردازد. از این رو رویکرد ارائه‌شده در این پژوهش سعی دارد با استفاده از اطلاعات تاریخچه برنامه از قبیل داده‌های کامیت، داده‌های نسخه‌های انتشاریافته برنامه و داده‌های ایرادات رفع‌شده برنامه توسعه‌دهنده مناسبی را برای رسیدگی به نظر معرفی کند. برای بیان بهتر مسأله، فردی را باید در نظر گرفت که در یک گروه از افراد فعالیت دارد که توسعه‌دهندگان یک برنامه موبایل پرطرفدار در فروشگاه گوگل هستند. این برنامه روزانه تعداد زیادی از نظرات را از سوی کاربران دریافت می‌کند. فرد مورد نظر به صورت مکرر نظرات کاربران را بررسی می‌کند

این مقاله در تاریخ ۱۲ اردیبهشت ماه ۱۳۹۸ دریافت و در تاریخ ۹ آبان ماه ۱۳۹۸ بازنگری شد.

مریم یونسی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران، (email: younesi@ce.sharif.edu).

عباس حیدرنوری (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران، (email: heydarnoori@sharif.edu).

فاطمه قنادی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران، (email: ghanadi@ce.sharif.edu).

1. Google Play Store
2. Apple App Store

که این نظرات می‌توانند حاوی تحسین و شکایات کاربران و همچنین اطلاعات مفیدی از قبیل گزارش خطا یا درخواست ویژگی جدید برای نرم‌افزار باشند. زمانی که این فرد نظری را مطالعه می‌کند که حاوی یک خطای نرم‌افزاری است، لزوماً نمی‌تواند در صدد رفع آن خطا برآید زیرا ممکن است که آن خطا مربوط به قسمتی از کد برنامه باشد که آن فرد در نوشتن آن نقشی نداشته است یا تخصصی در آن زمینه ندارد. بنابراین باید این نظر را به شخصی منتقل کند که به کد مربوط به خطای مذکور تسلط دارد و می‌تواند آن را رفع کند. از این طریق نیاز کاربران سریع‌تر برآورده می‌شود و در پی آن به جلب رضایت آنها و افزایش رتبه آن برنامه در فروشگاه گوگل منجر می‌شود. اما حجم بالای نظرات کاربران، پیچیده‌بودن کد برنامه و در بعضی مواقع تعداد زیاد توسعه‌دهندگان باعث می‌شود که این فرایند انتساب خطا بسیار زمان‌بر و مشکل شود. از این رو نیاز به یک فرایند خودکار انتساب خطا برای برنامک‌های کاربردی تلفن همراه با استفاده از نظرات کاربران احساس می‌شود.

چالش‌هایی که در این پژوهش با آنها روبه‌رو هستیم به صورت زیر است:

- برای برنامه‌های کاربردی موجود در فروشگاه‌ها روزانه تعداد زیادی نظر پست می‌شود که طبق پژوهش Chen و همکاران [۷]، تنها یک‌سوم از نظرات دارای اطلاعاتی است که می‌تواند برای بهبود نرم‌افزار به توسعه‌دهندگان کمک کند (به طور متوسط ۳۵/۱٪).
- بخش متنی بازخوردی که کاربران در فروشگاه‌های برنامه‌های کاربردی قرار می‌دهند یک توصیف آزاد است که دارای ساختار مشخص از پیش تعریف شده‌ای نیست.
- بر خلاف «گزارش خطا»، نظرات کاربران به جزئیات پیاده‌سازی اشاره نمی‌کنند. به عبارت دیگر یک عدم تطابق کلمات بین نظرات کاربران و کد برنامه وجود دارد، زیرا کاربران اطلاعی از نحوه پیاده‌سازی برنامه کاربردی ندارند و همین امر، انتساب خطا را پیچیده می‌کند.
- در این پژوهش نظرات کاربران مورد بررسی قرار می‌گیرند و این نظرات ساختاریافتگی و اطلاعات موجود در «گزارش خطا» (فرد رفع‌کننده خطا، نظرات توسعه‌دهندگان در مورد خطا، زمان رفع خطا، این که خطا رفع شده یا خیر و این که خطا به کدام موجودیت از محصول ارتباط دارد) را ندارند. اما پژوهش‌هایی که تا کنون حول موضوع انتساب خطا صورت گرفته‌اند از گزارش خطا استفاده می‌کنند. بنابراین نمی‌توان به طور کامل از روش‌های پژوهش‌های پیشین بهره برد و یا حتی راه حل پیشنهادی این پژوهش را با آنها مقایسه کرد.

برای رفع این مشکلات، پژوهش‌های متعددی انجام گرفته‌اند که در آنها سعی بر این بوده که اطلاعاتی را که از نظر توسعه‌دهندگان مفید است به صورت خودکار از نظرات کاربران استخراج کنند. اما هیچ یک از این پژوهش‌ها به این موضوع نپرداخته‌اند در صورتی که با روش‌های معرفی‌شده اطلاعات مهم از نظرات کاربران استخراج شوند، باز هم توسعه‌دهندگان با حجم زیادی از نظرات که مفید تشخیص داده شده‌اند مواجه هستند و باید آنها را به صورت دستی مطالعه نمایند. از این رو پژوهش پیش رو فواید زیر را نسبت به کارهای پیشین در پی خواهد داشت:

- حجم نظراتی که هر توسعه‌دهنده با آن روبه‌رو است کاهش می‌یابد و توسعه‌دهنده صرفاً نظراتی را می‌خواند که مربوط به حوزه کاری خودش است و اطلاعات موجود در نظرات می‌تواند برایش مفید باشد.
- خطاهایی که از نظرات استخراج می‌شوند به صورت خودکار به مناسب‌ترین فرد نسبت داده می‌شوند و این باعث می‌شود که آن خطا با صرف کمترین زمان و انرژی ممکن رفع شود و عمل نگهداری نرم‌افزار و رفع خطاها با فرایند بهتری نسبت به قبل، از نظر زمان و انرژی صورت گیرد.

۲- کارهای مرتبط پیشین

در این بخش پژوهش‌های پیشین مرتبط با زمینه انتساب خطا و همچنین نگاشت خطاهای شناسایی‌شده به تغییرات کد را مورد بررسی قرار می‌دهیم.

۲-۱ انتساب خطاهای برنامه به توسعه‌دهندگان

برخی از این پژوهش‌ها از معیار زمان رفع خطا برای انتساب خطا استفاده کرده‌اند [۱۰] و [۱۱]. در این پژوهش‌ها ذکر شده که برای بهبود روند انتساب خطا باید برای هر گزارش خطا، مناسب‌ترین فردی انتخاب شود که می‌تواند مشکل را در کمترین زمان ممکن شناسایی و حل کند. برای این منظور میزان زمانی را که هر توسعه‌دهنده به طور میانگین برای هر دسته از مشکلات صرف می‌کند تخمین زده‌اند.

در گروه دیگری از پژوهش‌ها سعی داشته‌اند بر اساس مکانی از کد برنامه که حدس می‌زنند آن گزارش خطا در آن قسمت رخ داده است انتساب خطا را انجام دهند [۱۲] تا [۱۴]. در این پژوهش‌ها ابتدا باید قسمت‌هایی از کد اصلی را که به گزارش خطا یا تغییر گزارش‌شده مربوط هستند را پیدا کرده و سپس با استفاده از اطلاعات موجود در آن فایل‌ها یا کامنت‌هایشان توسعه‌دهنده تغییردهنده آن خطوط از کد را پیدا می‌کنند. در نهایت نیز میزان مناسب‌بودن هر یک از توسعه‌دهندگان را برای رفع خطای گزارش‌شده بررسی می‌کنند و لیست مرتبی از توسعه‌دهندگان را برای رفع آن خطا پیشنهاد می‌دهند.

علاوه بر دو روش معرفی‌شده، می‌توان برای توسعه‌دهندگان پروفایل عملکرد ساخت و برای انتساب خطا از آنها بهره جست [۱۵] تا [۱۸]. به عبارت دیگر میزان تخصص برنامه‌نویس را بر اساس کارهایی که در گذشته انجام داده است با استفاده از تکنیک‌های یادگیری ماشین و بازیابی اطلاعات مورد بررسی قرار می‌دهند و بر اساس مرتبط‌بودن تخصص کاربران با موضوع گزارش خطا، افراد منتخب برای رفع آن ایراد مشخص می‌شوند و بر اساس پروفایل عملکردشان رتبه‌بندی صورت می‌گیرد. از نکات مثبت این روش این است افرادی که تجربه کمتری نسبت به دیگران دارند، همچنان از این شانس برخوردارند که در رفع مشکل گزارش‌های خطا فعالیت کنند.

در پژوهش دیگری [۱۹] شکری‌پور و همکاران نیز از تأخر گزارش‌های خطای رفع‌شده برای امر انتساب خطا استفاده کرده‌اند. منظور از تأخر این است که به چه میزان به زمان حال نزدیک‌تر است. ایشان در پژوهش خود سعی‌شان بر این بوده که با استفاده از فراداده زمان در تکنیک tf-idf فرایند انتساب اتوماتیک گزارش خطا را بهبود بخشند. در این روش با استفاده از میزان تأخر به کار بردن هر اصطلاح توسط هر برنامه‌نویس نشان‌دهنده‌ی این است که آن فرد چقدر درباره آن موضوع حضور ذهن دارد.

روش دیگری که برای انتساب خطا به کار گرفته شده استفاده از شباهت گزارش خطا با گزارش‌های پیشین است. Xia و همکاران [۲۰] روشی را به نام DevRec مبتنی بر دو نوع تحلیل ارائه دادند. تحلیل اول

مبتنی بر گزارش خطا بوده و فاصله اقلیدسی بین بردار ویژگی‌های گزارش‌های خطا را به عنوان معیار فاصله در نظر گرفته است و در مرحله بعد از الگوریتم نزدیک‌ترین همسایه چندبرچسبی برای پیداکردن نزدیک‌ترین گزارش‌های خطا به گزارش جدید استفاده کرده است. تحلیل دوم مبتنی بر توسعه‌دهنده است که با استفاده از خطاهایی که توسعه‌دهنده قبلاً رفع کرده است، فاصله آن را با گزارش خطای جدید اندازه‌گیری کرده و در نهایت لیستی از مناسب‌ترین توسعه‌دهنده‌ها را برای رفع خطا پیشنهاد می‌دهد.

۲-۲ نگاشت خطاهای شناسایی‌شده به تغییرات کد

پس از بررسی روش‌های انتساب خطاهای برنامه به توسعه‌دهندگان، با در نظر گرفتن این موضوع که ما در این پژوهش تنها نظرات کاربران و کد برنامه را در اختیار داریم و اطلاعاتی که برخی روش‌های انتساب خطا از گزارش‌های خطا به دست می‌آورند و به وسیله آن انتساب خطا را انجام می‌دهند، در نظرات کاربران وجود ندارد، به این نتیجه رسیدیم که پژوهش‌هایی را که حول موضوع ارتباط نظرات با کد برنامه هستند نیز بررسی کنیم. بنابراین در این بخش قصد داریم به بررسی پژوهش‌هایی که سعی کرده‌اند ارتباط بین خطاهای برنامه و تغییراتی که توسعه‌دهندگان در طول زمان در کد برنامه صورت داده‌اند بپردازیم. ابتدا پژوهش‌هایی را معرفی می‌کنیم که نظرات کاربران را که حاوی خطا هستند به تغییرات کد منبع نسبت می‌دهند. سپس با توجه به جدیدبودن این موضوع و وجود تعداد کم پژوهش‌هایی که ارتباط مستقیم با آن داشته باشند، در ادامه پژوهش‌هایی را بررسی می‌کنیم که از گزارش خطا استفاده کرده‌اند تا آنها را به تغییراتی که در کد منبع در طول زمان رخ داده است ارتباط دهند.

۲-۲-۱ روش‌های یافتن تناظر میان نظرات برنامک‌های کاربردی و کامیت‌های برنامه

Palomba و همکاران [۲] روشی را به نام CRISTAL معرفی کردند که در آن نظرات حاوی اطلاعات ارزشمند را توسط ابزار AR-miner [۷] شناسایی کردند و با پیاده‌سازی روش ReLink [۲۱]، آنها را به کد منبع برنامه مرتبط ساختند. سپس و نظارت می‌کنند که توسعه‌دهندگان تا چه میزان بازخورد کاربران را پیگیری کرده‌اند و نظرات کاربران را در نسخه‌های جدید برنامه پیاده‌سازی کرده‌اند که این در بالارفتن امتیاز برنامه نمود پیدا می‌کند. این مقاله به این اشاره می‌کند که تا چه میزان تیم توسعه نرم‌افزار از مکانیزم‌های جمع‌سپاری می‌تواند برای پیش‌بینی تغییرات آینده استفاده کند و چگونه این تغییرات روی رضایت کاربران تأثیر می‌گذارد که آن را از طریق معیار رتبه‌بندی می‌سنجند.

برای حل این مسأله CRISTAL یک روش چندمرحله‌ای پیشنهاد داده که در درجه اول نظرات حاوی اطلاعات مفید را شناسایی می‌کند و سپس نظرات را به متن کامیت‌ها و گزارش‌های خطای موجود در سیستم کنترل نسخه ربط می‌دهد. این کار با استفاده از نرمال‌سازی متن و محاسبه شباهت‌های متنی انجام می‌گیرد. در مرحله بعد ارتباطات اضافی که ایجاد شده بودند فیلتر می‌شوند. ایشان ۱۰۰ برنامه اندروید را با این ابزار بررسی کردند و به این نتیجه رسیدند که به طور میانگین توسعه‌دهندگان ۴۹٪ نظرات کاربران را پیاده‌سازی می‌کنند و هم این که نظرات کاربران واقعاً برای موفقیت اپلیکیشن مهم است زیرا اهمیت‌دادن به آنها باعث افزایش معیار رتبه‌بندی برای برنامه‌ها در فروشگاه گوگل شده است.

۲-۲-۲ روش‌های یافتن تناظر میان گزارش‌های خطا و کامیت‌های برنامه

در این بخش پژوهش‌هایی را معرفی می‌کنیم که به دنبال یافتن تناظری میان گزارش‌های خطای ثبت‌شده و تغییرات ایجادشده در برنامه هستند. در ادامه این پژوهش‌ها را بر اساس اطلاعاتی که برای یافتن تناظر استفاده کرده‌اند دسته‌بندی می‌نماییم:

• تعیین تناظر بر مبنای پیام کامیت و اطلاعات مرتبط با آن

در مطالعه Sliwerski و همکاران [۲۲]، روشی را در دو سطح لغوی و معنایی برای نگاشت گزارش‌های خطا و کامیت‌های برنامه ارائه کردند. در سطح لغوی پیام کامیت‌ها به توکن‌هایی شکسته می‌شوند. یک توکن می‌تواند یک شماره خطا یا یک کلیدواژه مانند «fixed»، «bug» یا غیره باشد. در سطح لغوی تناظرهای احتمالی میان گزارش‌های خطا و کامیت‌های برنامه شناسایی می‌شوند. در سطح معنایی صحت تناظرهای احتمالی به دست آمده در مرحله قبلی مشخص می‌گردد. این اعتبارسنجی با استفاده از ترکیب اطلاعات زیر صورت می‌گیرد:

۱) آیا خطای مورد نظر رفع شده است؟

۲) آیا پیام کامیت مورد نظر شامل توضیحات خطا است؟

۳) آیا نویسنده کامیت همان رفع‌کننده خطای مورد نظر است؟

ارزیابی این روش بر روی پایگاه داده‌های موزیلا و اکلیپس انجام گرفت که توانست بیش از ۴۰٪ از خطاهای رفع‌شده را به تغییرات مرتبط آن متصل کند.

در پژوهش دیگری ReLink [۲۱] معرفی شد که ابزاری است که ارتباط بین خطاهای برنامه و تغییرات کد را از روی شباهت متنی اطلاعات تغییرات و گزارش خطا به صورت خودکار پیدا می‌کند. در این روش ابتدا با استفاده از هیوریستیک‌های سنتی از قبیل وجود کلیدواژه‌های مشخص و وجود شماره خطا در پیام کامیت، تناظرهای صریح موجود در داده‌ها به دست می‌آید. سپس ویژگی‌های این تناظرها از قبیل فاصله زمانی میان کامیت رفع خطا و ثبت رفع آن، تناظرهای میان گزارش‌دهنده خطا و رفع‌کننده آن و شباهت متنی میان گزارش خطا و متن کامیت مرتبط با آن، تحلیل و شناسایی می‌گردند. در این روش برای به دست آوردن میزان شباهت میان گزارش‌های خطا و پیام کامیت‌ها، از مدل VSM استفاده گردیده و شباهت میان دو بردار گزارش خطا و پیام کامیت سنجیده شده است. در نهایت ارزیابی‌ها حاکی از آن بود که Relink توانسته به صورت میانگین ۷۸٪ تناظرهای میان گزارش‌های خطا و کامیت‌ها را با دقت ۸۹٪ شناسایی کند.

• تعیین تناظر بر مبنای پیام کامیت و فایل‌های تغییریافته در کد

روش‌هایی که تا کنون مطرح شد غالباً بر مبنای شباهت متنی میان گزارش‌های خطا و پیام‌های کامیت برنامه هستند. در پژوهش [۲۳]، Nguyen و همکاران، روشی چندلایه به نام MLink را ارائه کردند که در این روش ابتدا با استفاده از هیوریستیک‌های سنتی لینک‌های مشخص موجود میان گزارش‌ها و کامیت‌ها شناسایی می‌شوند. سپس تناظرهای مربوط به گزارش‌های خطا و کامیت‌هایی که از نظر زمانی با یکدیگر سازگار نیستند مورد پالایش قرار می‌گیرند. در مراحل بعدی با توجه به نام موجودیت‌های برنامه که در کامیت‌ها و در گزارش‌های خطا اشاره شده‌اند، تکه‌کدهایی که در گزارش‌های خطا آورده شده است و همچنین با توجه به شباهت متنی میان گزارش‌های خطا و کامیت‌ها (همانند آنچه در ReLink [۲۱] بود) تناظرهای موجود میان گزارش‌های خطا و کامیت‌های برنامه شناسایی می‌گردند. این روش بر روی سه پروژه متن‌باز

مورد مطالعه در [۲۱]، به منظور مقایسه با روش ReLink ارزیابی گردید. نتایج آزمایش‌های صورت‌گرفته نشان می‌دهد به صورت میانگین MLink توانسته است دقت روش‌های موجود از جمله ReLink را حدود ۱۸ درصد بهبود بخشد.

در پژوهش دیگری Le و همکاران [۲۴] برای حل مشکل وجود پیام‌های کامیت ناقص و یا عدم وجود پیام کامیت، روشی به نام RCLinker را ارائه کردند. این روش از دو فاز آموزش و استقرار تشکیل شده است.

در فاز آموزش ابتدا با استفاده از ابزار ChangeScribe [۲۵][1] تغییرات موجود در هر کامیت به صورت خلاصه بیان می‌گردد. خلاصه تولیدشده با پیام آن کامیت که در گذشته توسط توسعه‌دهنده نوشته شده است، ترکیب می‌گردد. پس از آن، شاخص‌های مهم از تناظرهای موجود در مجموعه داده یادگیری استخراج می‌گردد و در نهایت با استفاده از این شاخص‌ها یک مدل پیش‌بینی ساخته می‌شود. در این روش نیز برای محاسبه شباهت متنی میان پیام کامیت و گزارش خطا از مدل VSM استفاده می‌شود.

در فاز استقرار، مدل ساخته‌شده در فاز قبل و مجموعه‌ای از تناظرهای نامشخص به عنوان ورودی گرفته می‌شود و در نهایت مجموعه‌ای از تناظرهای میان کامیت‌ها و گزارش‌های خطا به عنوان خروجی بازگردانده می‌شوند.

• تعیین تناظر بر مبنای پیام کامیت و فایل‌های تغییریافته در کد و مستندات غیر کدی

در این زمینه Sun و همکاران [۲۶]، یک مطالعه بر روی ۱۸ پروژه بزرگ شرکت Apache با ۶۳۷۰ گزارش خطا و ۲۲۷۶۱ کامیت انجام دادند که هدف از این مطالعه بررسی نقش مستندات غیر کدی بر بهبود روش‌های تشخیص تناظر میان گزارش‌های خطا و کامیت‌ها است. نتایج این مطالعه نشان می‌دهد:

۱) در ۵۱٪ موارد، مستندات غیر کدی نسبت به پیام‌های کامیت دارای متن مشابه‌تری با گزارش‌های خطا هستند.

۲) روش‌هایی که مستندات غیر کدی را نیز در بازیابی تناظرهای میان گزارش‌های خطا و کامیت‌ها در نظر می‌گیرند، کارایی بالاتری را نسبت سایر روش‌ها دارند.

۳) در روش‌های بازیابی تناظر میان گزارش‌های خطا و کامیت‌ها، در نظر گرفتن مستندات غیر کدی باعث افزایش میانگین F-measure می‌گردد. در آزمایش‌های انجام‌گرفته بر روی ۱۸ پروژه مذکور در نظر گرفتن این مستندات باعث افزایش ۷۶/۲ تا ۶۳/۲۵ درصدی در معیار F-measure شده است.

بنابراین همین گروه در پژوهش دیگری [۲۷] روشی به نام FRLink را ارائه کردند که بر مبنای دو مسئله مهم ارائه شده است:

۱) در هیچ کدام از روش‌های گذشته مستندات غیر کدی برنامه مانند توضیحات هر انتشار برنامه در نظر گرفته نشده است. در حالی که در اغلب این مستندات اطلاعات جزئی درباره علت ایجاد تغییرات در برنامه (مانند وجود یک نقص) بیان شده است. همچنین تجربه نشان می‌دهد که واژگان مشترک زیادی میان این مستندات و گزارش‌های خطا وجود دارد.

۲) در هر کامیت تمامی فایل‌هایی که تغییر داده شده‌اند نقش یکسانی در رفع خطای مربوط به آن ایفا نمی‌کنند. در نتیجه می‌توان با استفاده از واژگان به کار رفته در گزارش‌های خطا، فایل‌های غیر مرتبط با یک گزارش خطا را پالایش نمود.

در این روش ابتدا با توجه به بازه زمانی گزارش‌های خطا، کامیت‌هایی که مربوط به هیچ خطایی نیستند فیلتر می‌شوند. تناظرهای باقیمانده به عنوان مجموعه داده آموزشی و آزمون استفاده می‌شوند. در مرحله بعد، FRLink میزان شباهت موجود در هر تناظر را بر اساس شباهت مبتنی بر کد و متن محاسبه می‌کند. شباهت مبتنی بر متن شامل دو نوع است. نوع اول از طریق مقایسه متن پیام کامیت‌ها و گزارش‌های خطا به دست می‌آید و نوع دوم از مقایسه متن مستندات غیر کدی (از جمله توضیحات انتشار) با گزارش‌های خطا محاسبه می‌شود که برای انجام این مقایسه از مدل بازیابی متن VSM استفاده می‌شود. در مرحله بعد با توجه به تناظرهای موجود در مجموعه داده آموزشی، آستانه شباهت مورد نیاز برای رسیدن به یک بازیابی حداقلی یادگیری می‌شود و در نهایت مدل به دست آمده برای پیش‌بینی وجود تناظر میان یک گزارش خطا و کامیت استفاده می‌شود.

ارزیابی این روش بر روی شش پروژه مشابه با [۲۴] اجرا گردید که در اغلب موارد FRLink توانست ۹۰٪ تناظرهای جاافتاده را با دقت بیش از ۵۰٪ پیش‌بینی کند و در نهایت توانست پیشرفت ۴۰ درصدی در معیار F-measure نسبت به روش RCLinker داشته باشد.

۳- رویکرد پیشنهادی

در این بخش ابتدا به معرفی چارچوب سطح بالای رویکرد پیشنهادی پرداخته و سپس وارد جزئیات هر یک از مؤلفه‌های آن خواهیم شد.

۳-۱ چارچوب سطح بالای رویکرد پیشنهادی

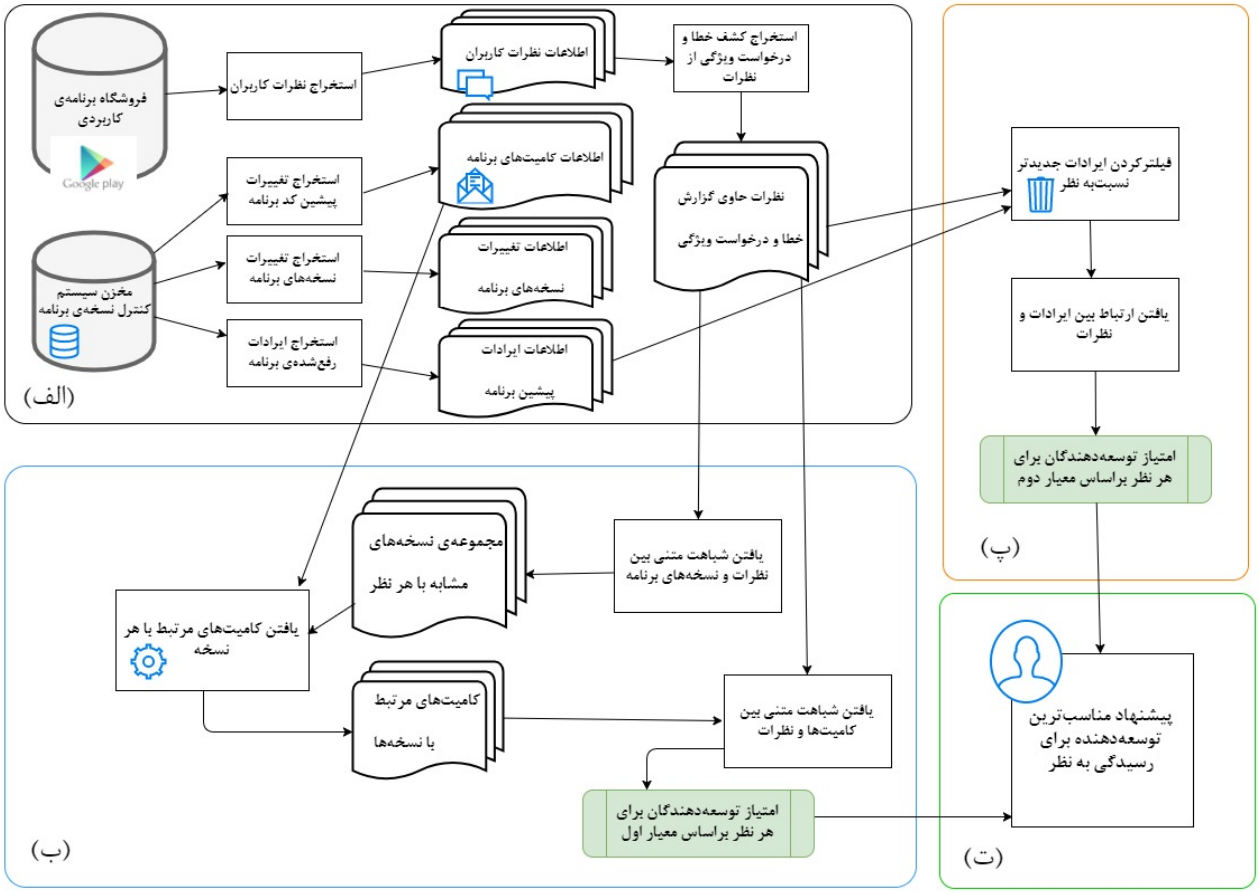
رویکرد پیشنهادی از شباهت بین نظرات برنامک و سابقه کارهایی که توسعه‌دهندگان در طول ساخت برنامک انجام داده بودند استفاده می‌کند تا توسعه‌دهنده مناسبی را برای رسیدگی به آن نظر پیشنهاد دهد. منظور از سابقه توسعه‌دهنده کارهایی از قبیل کامیت‌هایی که کرده‌اند و مشکلاتی از برنامه که در حین توسعه نرم‌افزار با آنها مواجه شده بودند است که نشان می‌دهند توسعه‌دهنده در چه بخش‌هایی از تولید آن نرم‌افزار حضور داشته و اگر مشکلی در آن مورد ایجاد شود برای حل آن آشنایی کافی را داراست. بنابراین راهکار پیشنهادی را بر اساس دو معیار بیان می‌کنیم که در شکل ۱ به آنها اشاره شده است:

قسمت الف بخش جمع‌آوری مجموعه داده است که اطلاعات مربوط به نظرات را استخراج می‌کنیم و سپس گزارش‌های خطا و پیشنهاد ویژگی‌ها را از دل نظرات بیرون می‌کشیم. همچنین اطلاعات ایرادات پیشین و کامیت‌ها و نسخه‌های برنامه را استخراج می‌کنیم.

در قسمت ب امتیاز توسعه‌دهندگان بر حسب معیار اولمان را محاسبه کردیم که این معیار میزان شباهت نظر با کامیت‌هایی است که هر توسعه‌دهنده انجام داده که در محاسبه این معیار اطلاعات متنی هر نسخه از برنامه را دخالت دادیم تا صرفاً کامیت‌هایی را بررسی کنیم که مربوط به نسخه‌های شبیه به نظر هستند. منظور از اطلاعات نسخه برنامه، متنی است که توسعه‌دهندگان در هنگام انتشار نسخه جدید از برنامه در فروشگاه گوگل می‌نویسند تا کاربران از تغییراتی که در نسخه جدید رخ داده است اطلاع یابند.

در قسمت پ از شکل نیز امتیاز توسعه‌دهندگان را بر حسب معیار دوم محاسبه کردیم که این معیار نشان‌دهنده این است که خطای مورد

۱. یک افزونه اکلیپس است که با استفاده از خلاصه‌سازی کد، برای تغییرات صورت‌گرفته در برنامه پیام کامیت مناسب تولید می‌کند.



شکل ۱: چارچوب سطح بالای رویکرد پیشنهادی.

بررسی‌مان به چه میزان به ایرادات پیشینی که کاربر در برنامه با آنها مواجه شده و آنها را رفع کرده است شباهت دارد. این شباهت نیز شباهت کسینوسی بین متن نظر و متن توصیف ایرادات پیشین برنامه است.

سپس در قسمت ت اولویت هر توسعه‌دهنده برای رسیدگی به آن نظر را بر حسب امتیازی که از دو معیار اول و دوم کسب کرده بود محاسبه می‌نماییم و توسعه‌دهنده‌ای را که بیشترین مجموع امتیاز را داشته باشد به عنوان یک توسعه‌دهنده مناسب برای رسیدگی به نظر معرفی می‌کنیم.

۳-۲ استخراج نظرات نشان‌دهنده کشف خطا

نظرات کاربران دارای ساختار مشخصی نیستند و متن آنها را افراد غیر متخصص به صورت عامیانه نوشته‌اند. بنابراین ابتدا بایستی کارهایی از جنس پیش‌پردازش روی داده‌ها انجام گیرد. این کارها عبارتند از حذف غلط‌های املایی و مخفف‌نویسی‌ها، حذف کلمات مانع و نشانه‌گذاری‌ها، ریشه‌یابی کلمات و جایگذاری کلمات هم‌معنی و حذف نظرات غیر انگلیسی.

با توجه به تعداد زیاد نظرات و این که حجم زیادی از آنها حاوی اطلاعات مفیدی نیستند، باید نظراتی که نشان‌دهنده کشف خطای برنامه هستند از باقی جدا شوند که تا کنون پژوهش‌های متعددی حول موضوع دسته‌بندی نظرات کاربران صورت گرفته‌اند. از بین ابزارهای معرفی‌شده در این پژوهش‌ها برای جداسازی گزارش‌های خطا از باقی نظرات از ابزار ARdoc [۶] استفاده شد زیرا روش جدیدی است و نسبت به روش‌های دیگر دقت مناسب‌تری را داراست. این ابزار با ترکیب سه روش پردازش زبان طبیعی، پردازش متن و تحلیل عقاید[1] نظرات کاربران را به پنج دسته کشف خطا، درخواست ویژگی جدید، جستجوی اطلاعات، ارائه‌ی اطلاعات و سایر نظرات دسته‌بندی می‌نماید.

۳-۳ انتساب خطا بر حسب شباهت نظر با تغییرات پیشین کد

در این معیار قصد بر این بوده خطاهایی را که کاربران در نظرات گزارش کرده‌اند از لحاظ متنی با پیام‌های کامیت مقایسه کنیم و ارتباط بین آنها را بیابیم. اگر یک خطا با یک تغییری که یک توسعه‌دهنده در برنامه ایجاد کرده ارتباط داشته باشد بدین معنی است که آن توسعه‌دهنده نسبت به دیگر افراد با آن قسمت از کد آشنایی بیشتری دارد و می‌تواند ایرادی را که مربوط به آن باشد، راحت‌تر از دیگران رفع کند. اما نکته‌ای که وجود دارد این است که این نظرات توسط افراد غیر متخصص نوشته شده، بنابراین ممکن است عباراتی که آنها به کار برده‌اند با تغییرات کد برنامه سنخیتی نداشته باشد و با توجه به تعداد زیاد کامیت‌ها به نتیجه درستی نرسیم. بنابراین برای این که فضای خالی بین نظرات کاربران و تغییرات کد برنامه را از بین ببریم از تغییرات نسخه‌های برنامه به عنوان پل ارتباطی بهره می‌جوییم. زیرا متنی که توسعه‌دهندگان در فروشگاه برنامک کاربردی به عنوان تغییراتی که در نسخه برنامه رخ داده نوشته‌اند، طوری است که برای کاربر قابل فهم باشد و کاربر بتواند به راحتی دریابد که برنامه نسبت به قبل چه تغییراتی کرده و آیا نسخه جدید برنامه ارزش نصب دارد یا خیر. در پژوهشی نیز Sun و همکاران [۲۷] به این نتیجه رسیده بودند که استفاده از مستندات غیر کدی از قبیل تغییرات نسخه‌های برنامه توانسته سبب بهبود دقت یافتن ارتباط بین خطاهای برنامه و تغییرات کد منبع شود.

1. Sentiment Analysis

بنابراین ابتدا شباهت متنی بین نظرات خطادار و نسخه‌های برنامه را به دست می‌آوریم و سپس کامیت‌های نسخه‌هایی را که مشابه با نظرات تشخیص داده شده‌اند استخراج می‌کنیم و در آخر شباهت متنی کامیت‌های نسخه مورد نظر را با نظر خطادار مورد بررسی می‌سنجیم. سپس بر اساس این که چه کسانی کامیت مشابه با نظر را انجام داده‌اند به هر توسعه‌دهنده از لحاظ مناسب‌بودن برای رسیدگی به هر نظر امتیازی اختصاص می‌دهیم.

برای یافتن شباهت متنی بین نظرات و نسخه‌های برنامه و همچنین نظرات و کامیت‌های نسخه‌های مشابه از معیار Tf-idf و شباهت کسینوسی استفاده می‌کنیم. برای این منظور از مدل فضای بردار استفاده می‌کنیم که در آن هر متن توسط یک بردار nبعدی $\langle w_1,\ldots,w_n\rangle$ نمایش داده می‌شود که در آن n تعداد عبارات و w_i وزن هر یک از عبارات را نشان می‌دهد

$$w_i = tf_i \times idf_i \quad (۱)$$

$$idf_i = \log \frac{|D|}{|d : t_i \in d|} \quad (۲)$$

ایده اصلی این روش این است که وزن هر عبارت در یک متن با فرکانس تکرار آن عبارت در آن متن افزایش و با فرکانس تکرار آن در متن‌های دیگر کاهش می‌یابد. پس از محاسبه وزن عبارات با استفاده از شباهت کسینوسی این بردارها میزان شباهت‌های متنی را محاسبه می‌نماییم

$$Sim = \frac{\sum_{i=1}^{n} w_{1i} w_{2i}}{\sqrt{\sum_{i=1}^{n} w_{1i}^2 \times \sum_{i=1}^{n} w_{2i}^2}} \quad (۳)$$

۴-۳ انتساب خطا بر حسب شباهت نظر با ایرادات پیشین برنامه

موضوعاتی که توسعه‌دهندگان در قسمت Issues سیستم کنترل نسخه گیت در موردشان با هم صحبت می‌کنند می‌تواند برای ما از این نظر که هر توسعه‌دهنده در گذشته در چه حوزه‌هایی فعالیت داشته و چه ایراداتی از برنامه را رفع کرده است کمک‌کننده باشد. در پژوهش‌های دیگری که در حوزه انتساب خطا انجام شدند یک سامانه ردیابی خطا[1] وجود داشت که وقتی خطای جدیدی پیش می‌آمد و می‌خواستند آن را به توسعه‌دهنده مناسب نسبت دهند، از طریق آن سامانه به تاریخچه خطاهای پیشین دست می‌یافتند و مشاهده می‌کردند که خطای مورد نظرشان به کدام یک از خطاهای قبلی برنامه شباهت بیشتری دارد و فردی که آن خطاها را رفع کرده را به عنوان فردی که می‌تواند خطای جدید را رفع کند پیشنهاد می‌دادند. اما در پژوهش ما چنین سامانه‌ای برای برنامه‌های اندروید وجود ندارد و تنها چیزی که در دست داریم همین قسمت از سیستم گیت است.

بنابراین در این روش در گام اول ایراداتی را از برنامه که نسبت به نظر ثبت‌شده جدیدتر هستند پالایش می‌کنیم زیرا از لحاظ منطقی در نظر گرفتن آنها درست نیست. سپس مانند معیار قبلی که ذکر شد از شباهت کسینوسی میزان شباهت بین نظر و ایرادات را می‌سنجیم و به توسعه‌دهندگان برای رسیدگی به آن نظر امتیاز می‌دهیم

$$S_{final}(d,r) = S_{Commits}(d,r) + S_{Issues}(d,r) \quad (۴)$$

1. Bug Tracking System

۵-۳ پیشنهاد مناسب‌ترین توسعه‌دهنده

حال می‌توانیم بر اساس دو معیاری که بیان شد مناسب‌ترین توسعه‌دهنده را برای رسیدگی به خطای ذکرشده در نظر کاربر پیشنهاد دهیم. برای این منظور همان گونه که در برابری ۴ نشان داده شده اولویت هر یک از توسعه‌دهندگان برای رسیدگی به هر نظر $(S_{final}(d,r))$ را برابر با جمع امتیازاتی که از دو معیار ذکرشده کسب کرده است قرار می‌دهیم.

٤- ارزیابی

در این بخش به شرح مجموعه داده، روش ارزیابی و سپس ارائه نتایج به دست آمده خواهیم پرداخت.

٤-١ مجموعه داده برای ارزیابی

برای انتخاب برنامک‌هایی که در این پژوهش مورد بررسی قرار دادیم باید ویژگی‌هایی را در نظر می‌گرفتیم. اول این که باید به این نکته توجه می‌کردیم که برنامه رایگان باشد و از سیستم کنترل نسخه گیت استفاده کند تا ما بتوانیم به اطلاعات کد منبع آن از قبیل کامیت‌های پیشین برنامه، اطلاعات نسخه‌های انتشاریافته برنامه و اطلاعات ایرادات پیشین آن دسترسی داشته باشیم. در مرحله بعد باید در نظر می‌گرفتیم که تعداد توسعه‌دهندگان برنامه کم نباشد زیرا اگر به عنوان مثال تعداد توسعه‌دهندگان یک برنامه یک یا دو نفر باشد انتساب خطا به توسعه‌دهنده مناسب، معنی خود را از دست می‌دهد و هیچ کارایی برای آن برنامه و توسعه‌دهنده‌اش نخواهد داشت.

پس از این مورد با توجه به رویکردمان و استفاده از اطلاعات نسخه‌های برنامه، باید بررسی می‌کردیم تعداد نسخه‌هایی که از برنامه منتشر شده کم نباشد تا بتوانیم استفاده از اطلاعات نسخه برنامه را نیز توسط این داده‌ها مورد ارزیابی قرار دهیم. از طرفی به صورت مشابه باید در نظر می‌گرفتیم که تعداد ایراداتی که توسعه‌دهندگان در قسمت Issues سیستم گیت مطرح کرده‌اند از حدی بیشتر باشد که هم بدانیم استفاده از این بخش گیت برای توسعه‌دهندگان مهم بوده و هم بتوانیم از آن داده‌ها نیز استفاده کنیم و مورد ارزیابی قرارشان دهیم. به طور مشابه کامیت‌های توسعه‌دهندگان برنامه را بررسی کردیم که پیام‌های کامیتی که نوشته‌اند از کیفیت حداقلی برای مقایسه با نظرات برخوردار باشند. ویژگی دیگری که در مجموع برای برنامه‌هایمان در نظر گرفتیم این بود که سعی داشتیم برنامه‌ها از دسته‌های مختلفی باشند تا تنوع کارکردهای برنامه‌ها را در داده‌هایمان حفظ کرده باشیم.

در جدول ۱ برنامک‌های انتخاب‌شده برای ارزیابی رویکرد پیشنهادی آورده شده‌اند.

٤-٢ سؤال‌های پژوهشی

برای ارزیابی رویکرد پیشنهادی سه سؤال پژوهشی پیش رو داریم:

- **تحلیل دسته‌بندی نظرات کاربران**

همان گونه که گفته شد برای دسته‌بندی نظرات کاربران که از برنامه‌ها استخراج نمودیم از ابزار ARdoc استفاده نمودیم. بنابراین نیاز است عملکرد ابزار را روی برنامه‌هایمان مورد سنجش قرار دهیم تا مطمئن شویم ابزار دقت خوبی در انتخاب نظرات گزارش‌دهنده خطا یا درخواست ویژگی داشته باشد. زیرا اگر دقت این ابزار در این امر پایین باشد در نتیجه نهایی رویکرد ما تأثیر به‌سزایی خواهد داشت. بنابراین سؤال پژوهشی که

جدول ۱: اطلاعات برنامک‌های مورد بررسی.

نام برنامه	تعداد توسعه‌دهندگان فعال	تعداد مشکلات رفع‌شده	تعداد کامیت‌ها	تعداد نسخه‌ها
AF-Wall	۶	۴۳۸	۱۰۸۳	۵۴
AntennaPod	۲۲	۱۲۳۱	۳۹۶۳	۷۷
Atarashii	۷	۱۵۲	۲۳۲۵	۷۷
FrostWire	۹	۱۸۶	۳۷۷۷	۲۵۲
Gibberbot	۹	۳۱۲	۲۹۰۷	۱۱۷
K-۹-Mail	۳۴	۹۷۹	۷۳۳۵	۳۵۴
OpenPGP-Keychain	۱۴	۱۴۱۵	۶۴۲۰	۶۱
Owncloud	۱۳	۱۰۳۱	۶۱۵۶	۶۸
RedReader	۸	۱۹۴	۱۱۲۵	۶۲
Simple-Last.fm-Scrobbler	۵	۲۴۰	۵۸۸	۴۰

جدول ۲: شرح امتیاز ارزیاب به ارتباط بین نظرات و کامیت‌ها.

امتیاز	معنی	شرح
امتیاز ۰	نامرتبط	به عقیده من این کامیت و نظر با هم هیچ ارتباطی ندارند.
امتیاز ۱	حدوداً نامرتبط	به عقیده من کامیت و نظر حدوداً نامرتبط هستند اما کسی که این کامیت را انجام داده ممکن است درباره رسیدگی به این نظر چیزهایی بداند.
امتیاز ۲	حدوداً مرتبط	به عقیده من این کامیت و نظر حدوداً با هم ارتباط دارند و کامیت‌کننده ممکن است در رسیدگی به نظر موفق باشد.
امتیاز ۳	کاملاً مرتبط	به عقیده من این کامیت و نظر کاملاً با هم ارتباط دارند و کامیت‌کننده در رسیدگی به نظر موفق خواهد بود.

در این بخش می‌خواهیم به آن پاسخ دهیم این است که «آیا ابزاری که برای دسته‌بندی نظرات استفاده کرده‌ایم دقت کافی را برای انتخاب نظرات کشف خطا و درخواست ویژگی روی مجموعه داده ما دارد؟»

• ارزیابی تأثیر استفاده از اطلاعات نسخه‌های پیشین برنامه

در این بخش از ارزیابی بخشی از رویکرد پیشنهادی را مورد سنجش قرار می‌دهیم که در آن از اطلاعات نسخه‌های انتشاریافته برنامه استفاده کردیم. در واقع قصد داریم به این سؤال پژوهشی پاسخ دهیم که «آیا اطلاعات نسخه‌های انتشاریافته برنامه برای نگاشت نظرات کاربران به کامیت‌های توسعه‌دهندگان مفید بوده است؟»

• ارزیابی و مقایسه معیارهای انتساب خطا

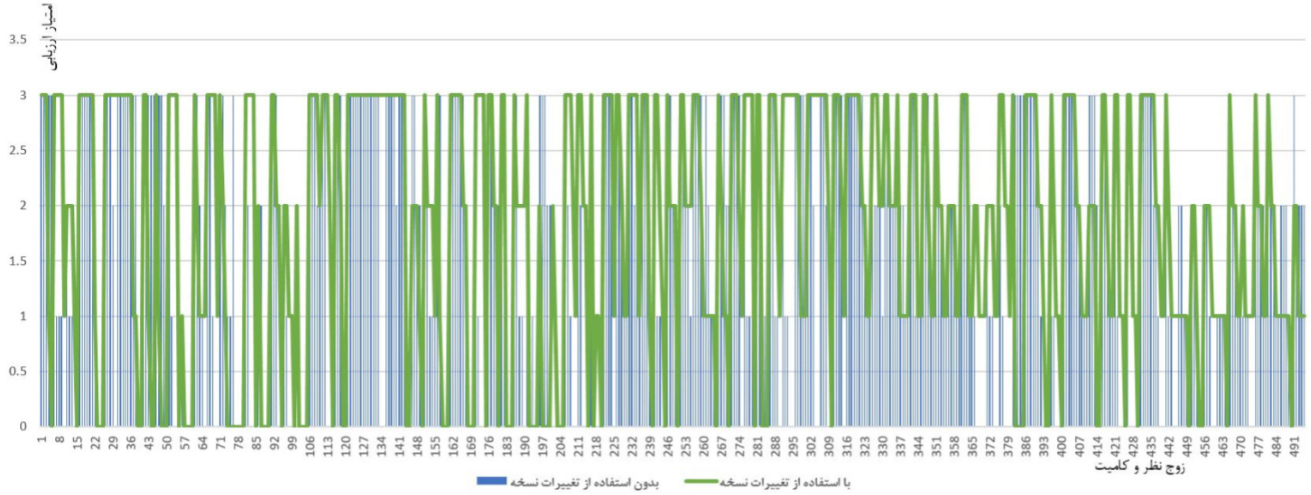
هدف سؤال پژوهشی سوم که با آن روبه‌رو هستیم این است که کیفیت انتساب خطاهای برنامک به برنامه‌نویسان را مورد ارزیابی قرار دهد: «تا چه میزان لیستی از توسعه‌دهندگان که ارائه کرده‌ایم می‌تواند از نظر برنامه‌نویسان ارزیابی‌کننده صحت داشته باشد؟»

۳-٤ روش ارزیابی

ارزیابی راهکار پیشنهادی توسط ۵ نفر برنامه‌نویس با سابقه برنامه‌نویسی حداقل ۵ سال انجام شد که هر یک بخشی از این پژوهش را مورد ارزیابی قرار دادند. برای پاسخ به سؤال پژوهش اول هدفشان این بود نظراتی را که ابزار به عنوان گزارش خطا معرفی کرده بررسی کنند که آیا این نظرات در واقع گزارش خطا هستند یا خیر. بنابراین بدین طریق می‌توان دقت ابزار دسته‌بندی نظرات را برای پیداکردن نظرات خطادار سنجید.

نحوه ارزیابی برای سؤال پژوهش دوم بدین صورت است که یک بار نظرات کاربران را به طور مستقیم و با محاسبه شباهت متنی به کامیت‌ها ارتباط می‌دهیم و یک بار دیگر از اطلاعات نسخه‌های پیشین به عنوان پل ارتباطی برای نگاشت نظرات به کامیت‌ها استفاده می‌کنیم. در نهایت برنامه‌نویسان می‌سنجند که از لحاظ معنایی میزان شباهت نظر به کامیت در هر دو حالت به چه میزان بوده است. در واقع بین این دو روش مقایسه می‌نماییم و می‌سنجیم که در کدام حالت کامیت‌های مرتبطتری به نظرات کاربران نگاشت شده است. برنامه‌نویسان ارزیابی‌کننده در این مورد آموزش داده شدند و ۷ برنامه را مورد ارزیابی قرار دادند. هدف ایشان این بود که از لحاظ معنایی شباهت بین کامیت‌ها و نظرات را پیدا کنند. به عبارت دیگر به آنها تأکید شد که نظرات و کامیت‌هایی که به آنها نگاشت شده را بخوانند و با این دید که آیا کسی که آن کامیت را انجام داده می‌تواند به آن نظر رسیدگی کند یا خیر، به آن زوج مرتب نظر و کامیت از ۰ تا ۳ نمره‌دهی کنند. شرح معنی هر یک از این نمرات در جدول ۲ آورده شده است.

برای پاسخ‌گویی به سؤال سوم از برنامه‌نویسان خواسته شد پس از مطالعه کامل و دقیق توصیف هر برنامه به ارزیابی ارتباطات بین نظرات و کامیت‌ها با واسطه نسخه‌های برنامه و همچنین ارتباطات بین نظرات و ایرادات پیشین برنامه که استخراج کرده بودیم بپردازند. به عبارت دیگر دو معیاری را که در رویکرد پیشنهادی معرفی شد مورد ارزیابی قرار می‌دهند تا نتیجه بگیریم که ترکیب این دو معیار تا چه میزان در روند انتساب خطا مفید واقع شده است. مانند بخش قبل افراد ارزیابی‌کننده به این ارتباطات امتیاز در بازه ۰ تا ۳ دادند. پس از این که امتیازات هر یک از این معیارها مشخص شد، هر یک از توسعه‌دهندگان به ازای هر یک از نظرات امتیازی گرفته است که ما برای محاسبه امتیاز یک توسعه‌دهنده به ازای یک نظر خاص، بین امتیازاتی که بر اساس دو معیار برای آن نظر کسب کرده، امتیاز بیشینه را به عنوان امتیاز نهایی در نظر می‌گیریم و بدین صورت میزان مناسب‌بودن توسعه‌دهنده برای رسیدگی به آن نظر را مورد ارزیابی قرار می‌دهیم. این ارزیابی به دو صورت انجام می‌گیرد که در درجه اول محاسبه می‌کنیم که چند درصد از افرادی که رویکرد پیشنهادی به عنوان حل‌کننده مشکل معرفی کرده توانسته‌اند نمره ۳ را از ارزیاب‌ها با توجه به کامیت‌ها و ایرادات رفع‌شده‌شان بگیرند. در درجه بعد روند صعودی‌بودن نمراتی که توسعه‌دهندگان فهرست پیشنهادی توانسته‌اند از افراد ارزیاب دریافت نمایند را مورد بررسی قرار می‌دهیم. بدین منظور که آیا پنج نفری که برای رسیدگی به نظر پیشنهاد دادیم به همان ترتیبی هستند که افراد ارزیابی‌کننده انتخاب کرده‌اند؟



شکل ۲: مقایسه نمرات ارزیابی به شباهت بین کامیت و نظر در دو حالت باواسطه و بدون واسطه اطلاعات تغییرات نسخه‌های پیشین.

جدول ۳: عملکرد ابزار ARDOC روی نظرات کاربران.

نام برنامه	درصد نظرات گزارش خطا	درصد نظرات درخواست ویژگی	درصد نظرات غیر واضح	درصد نظرات تحسین	مجموع گزارش خطا و درخواست ویژگی
AF-Wall+	۷۳	۵/۵	۱۶	۵/۵	۷۸/۵
AntennaPod	۴۷/۰۵	۴۷/۰۵	۵/۸۸	۱۱/۷۶	۹۴/۱
Atarashii!	۸۲/۳۵	۰	۱۷/۶۴	۰	۸۲/۳۵
FrostWire	۴۳/۲۸۳	۴۱/۷۹	۱۰/۴۴	۵/۹۷	۸۵/۰۷
Gibberbot	۶۶/۶۶	۲۰/۸۳	۴/۱۶	۸/۳۳	۸۷/۵
K-۹-Mail	۸۵/۷۱	۸/۵۷	۲/۸۵	۵/۷۱	۹۴/۲۸
OpenPGP-Keychain	۶۲/۵	۲۰/۸۳	۰	۱۶/۶۶	۸۳/۳۳
Owncloud	۸۶/۲۰	۸/۶۲	۵/۱۷	۰	۹۴/۸۲
RedReader	۳۴/۶۱	۲۶/۹۲	۷/۶۹	۳۰/۷۶	۶۱/۵۳
Simple-last.fm-Scrobbler	۹۰	۰	۱۰	۰	۹۰
میانگین	۶۷/۱۳۶	۱۸/۰۱۱	۷/۹۸۳	۸/۴۶۹	۸۵/۱۴۸

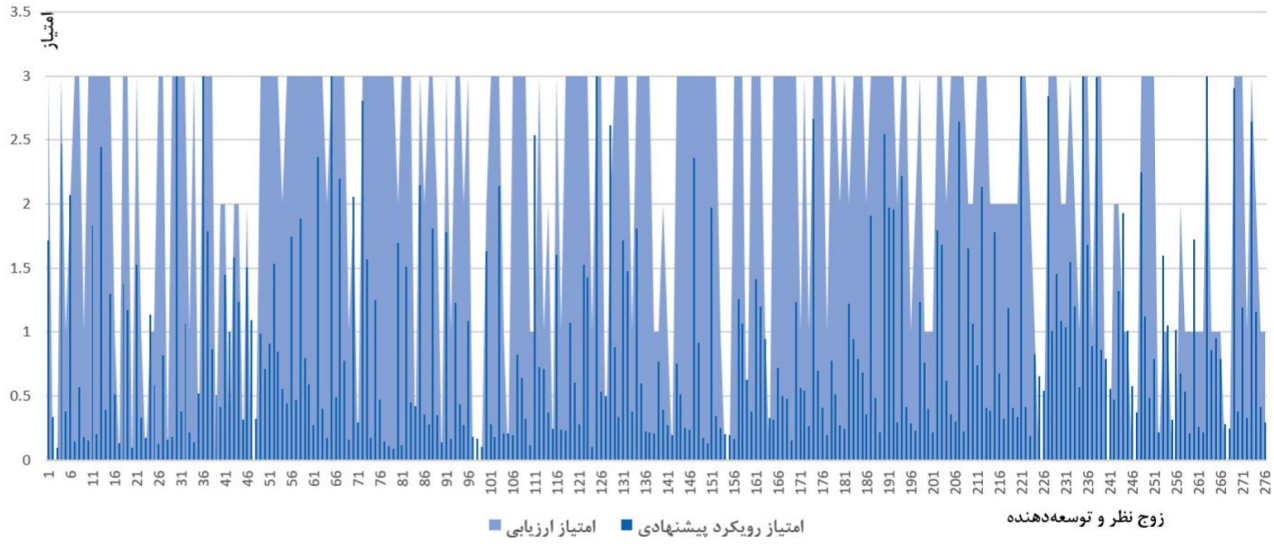
٤-٤ نتایج ارزیابی

نتایج ارزیابی برای پاسخ به سؤال پژوهشی اول در جدول ۳ آورده شده است. در این جدول درصد دقت دسته‌بندی نظرات به چهار گروه گزارش خطا، درخواست ویژگی، تحسین و نظرات غیر واضح برای ۱۰ برنامه محاسبه شده که در کل در سطر آخر جدول، میانگین آنها آورده شده است. همان طور که قابل مشاهده است در ستون آخر جدول میزان دقت این ابزار در محاسبه نظرات گزارش خطا و درخواست ویژگی آمده که حاکی از آن است که چه درصدی از نظراتی که ARdoc به عنوان گزارش خطا و درخواست ویژگی به ما معرفی کرده در واقع از نوع گزارش خطا و درخواست ویژگی بوده است.

نتایج ارزیابی برای سؤال پژوهشی اول حاکی از آن است که این ابزار در تشخیص نظرات گزارش خطا و درخواست ویژگی برای برنامه‌های مختلف دقت بین ۶۱/۵۳ تا ۹۴/۸۲ درصد را داراست که در کل این دقت به طور میانگین برای ۱۰ برنامه برابر ۸۵/۱۴۸ درصد است. این میانگین نشان‌دهنده این است که ۸۵/۱۴۸ درصد نظراتی که ARdoc به عنوان نظرات نشان‌دهنده گزارش خطا و درخواست ویژگی معرفی کرده به راستی حاوی گزارش خطا یا درخواست ویژگی بوده‌اند و باقی آنها یا صرفاً تحسین و رضایت کاربر را نشان داده‌اند یا به صورت واضح منظور خود را از خطا بیان نکرده‌اند.

همچنین همان طور که در جدول قابل مشاهده است در برنامه‌های مختلف درصدهایی که برای نتیجه ابزار ARdoc به دست آورده‌ایم متفاوت است. به عبارت دیگر این گونه نیست که بتوان گفت در تمامی برنامه‌ها این ابزار به صورت یکسان عمل کرده است زیرا ماهیت این برنامه‌ها متفاوت است و بنابراین که برنامه در چه سطحی از توسعه‌یافتگی و کیفیت باشد می‌تواند نظرات مختلفی داشته باشد. به عنوان مثال برای برنامه RedReader تقریباً یک‌سوم نظراتی که ابزار استخراج کرده خطا و درخواست ویژگی نبوده، بلکه صرفاً نشان‌دهنده تحسین و میزان رضایت کاربران از برنامه است که همین باعث شده دقت ابزار در شناسایی نظرات حاوی گزارش خطا و درخواست ویژگی پایین بیاید. دلیل این امر این است که برنامه از کیفیت بالایی برخوردار بوده و نیاز کاربران را رفع می‌کرده و درصد بالایی از نظراتی که کاربران برای آن برنامه قرار داده‌اند نشان‌دهنده تحسین برنامه بوده است که همین موضوع باعث شده دقت ARdoc کاهش یابد و به حدود ۶۰٪ برسد. به جز این برنامه که نظراتش بیشتر نشان‌دهنده رضایت کاربران بوده، دقت ARdoc در باقی برنامه‌ها برای تشخیص نظرات گزارش خطا و درخواست ویژگی بالا است و طبق نتیجه میانگین مجموع نظراتی که این ابزار به طور صحیح به عنوان گزارش خطا و درخواست ویژگی تشخیص داده است می‌توان استنباط کرد که این ابزار برای استفاده در این پژوهش مفید واقع شده و به طور میانگین دقت بالایی دارد.

برای پاسخ به سؤال پژوهشی دوم نتایج ارزیابی در شکل ۲ و جدول ۴ آورده شده است.



شکل ۳: مقایسه امتیازات توسعه‌دهندگان برای نظرات از نظر رویکرد پیشنهادی و فرد ارزیابی‌کننده.

جدول ۴: نتایج ارزیابی استفاده از اطلاعات تغییرات نسخه‌های پیشین.

نام برنامه	میانگین امتیازات با استفاده از تغییرات نسخه	میانگین امتیازات بدون استفاده از تغییرات نسخه	نمودار سبز بالای نمودار آبی	نمودار سبز پایین نمودار آبی	دو نمودار منطبق بر هم
AF-Wall+	۱٫۸	۱٫۸	٪۲۲٫۶۶	٪۲۲٫۶۶	٪۵۴٫۶۶
AntennaPod	۱٫۷۲	۱٫۵۰	٪۲۰	٪۹٫۰۹	٪۷۰٫۹
Atarashii!	۱٫۷۴	۱٫۵۷	٪۲۴٫۷	٪۱۶٫۴۷	٪۵۸٫۸۲
FrostWire	۲٫۱۲	۱٫۹۰	٪۳۰٫۴۷	٪۱۹٫۰۴	٪۵۰٫۴۷
Gibberbot	۱٫۹۳	۱٫۶۱	٪۳۱٫۶۶	٪۳۳٫۸	٪۶۰
OpenPGP-Keychain	۱٫۸	۱٫۷۸	٪۲۵٫۴۵	٪۱۴٫۵۴	٪۶۰
ownCloud	۱٫۳۶	۱٫۳۰	٪۲۵	٪۱۸٫۳۳	٪۵۶٫۶۶
کل	۱٫۸۱	۱٫۶۶	٪۲۶٫۶۶	٪۱۶٫۱۶	٪۵۷٫۷۷

جدول ۵: نتایج ارزیابی انتساب خطای برنامه به توسعه‌دهندگان.

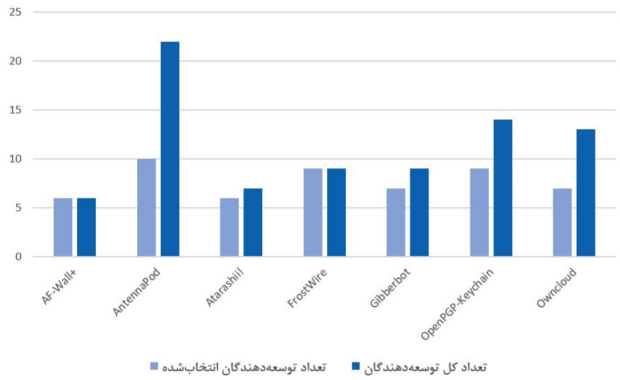
نام برنامه	دقت انتساب خطا طبق اولین توسعه‌دهنده پیشنهادشده	تعداد توسعه‌دهندگان پیشنهادشده	تعداد کل توسعه‌دهندگان
AF-Wall+	٪۸۷٫۵	۶	۶
AntennaPod	٪۶۳٫۶۳	۱۰	۲۲
Atarashii!	٪۸۲٫۳۵	۶	۷
FrostWire	٪۸۰٫۹۵	۹	۹
Gibberbot	٪۷۵	۷	۹
OpenPGP-Keychain	٪۶۳٫۶۳	۹	۱۴
OwnCloud	٪۶۶٫۶۶	۷	۱۳
میانگین کل	٪۷۴٫۲۴	۷٫۷	۱۱٫۴۲

محور افقی نمودار زوج‌های کامیت و نظر را نشان می‌دهد و محور عمودی آن میزان امتیازی را که هر یک از زوج‌های کامیت و نظر بر اساس دو معیار مختلف از افراد ارزیابی‌کننده دریافت کرده‌اند بیان می‌کند. جدول ۴ نیز نتایج ارزیابی را به صورت کمی بیان می‌کند که میزان بهبود در صورت استفاده از اطلاعات نسخه‌ها به درستی مورد بررسی قرار گیرد.

بررسی نتایج ارزیابی حاکی از آن است که در کل در ٪۲۶ موارد استفاده از اطلاعات تغییرات نسخه توانسته در میزان امتیازات بهبود ایجاد کند و در ٪۱۶ موارد باعث کاهش امتیاز از طرف افراد ارزیاب شده است. بنابراین در کل باعث شده در امتیازاتی که افراد ارزیاب به ارتباط بین نظرات و کامیت‌ها داده‌اند بهبود ایجاد شود. همچنین با بررسی میانگین امتیازات در هر دو حالت متوجه می‌شویم که استفاده از اطلاعات نسخه‌های پیشین باعث بالارفتن میانگین امتیازات شده است.

در شکل ۲ نیز امتیاز ارزیابی هر دو روش آورده شده که امتیاز روش با استفاده از تغییرات نسخه با رنگ سبز و امتیاز روش دیگر با رنگ آبی نشان داده شده است. در این نمودار قابل مشاهده است که نمودار نمراتی که با واسطه تغییرات نسخه دریافت کرده است تقریباً در بالای نمودار بدون واسطه تغییرات نسخه قرار گرفته و این بدین معنی است که به صورت میانگین امتیاز روشی که از تغییرات نسخه استفاده کرده بیشتر بوده است. بنابراین تغییرات نسخه توانسته در یافتن ارتباط بین نظرات و کامیت‌های برنامه مفید واقع شود.

برای پاسخ به سؤال پژوهشی سوم نتایج ارزیابی در جدول ۵ آورده شده که نشان می‌دهد برای هر برنامه در چند درصد از نظرات، فردی که رویکرد پیشنهادی به عنوان توسعه‌دهنده مناسب معرفی کرده با فردی که طبق نظر ارزیاب مناسب بوده مطابقت دارد. همچنین در شکل ۳ نتیجه



شکل ۴: تعداد توسعه‌دهندگان انتخاب‌شده از بین تمام توسعه‌دهندگان برنامه.

مقایسه امتیازات توسعه‌دهندگان برای نظرات از نظر رویکرد پیشنهادی و فرد ارزیابی‌کننده برای ۷ برنامه آورده شده است.

نتایج حاکی از آن است که رویکرد پیشنهادی با دقت ۷۴٫۲۴٪ می‌تواند انتساب خطا را انجام دهد به گونه‌ای که اگر اولین فردی که در فهرست پیشنهادی برای هر نظر قرار دارد از فرد ارزیابی‌کننده برای آن نظر بتواند نمره ۳ دریافت کند، به عنوان یک فرد مناسب شناخته می‌شود و نتیجه رویکرد پیشنهادی در مورد او درست بوده است. همچنین جدول ۵ و شکل ۴ نشان می‌دهند که توسعه‌دهندگان یک برنامه همه به میزان خوبی شانس انتخاب‌شدن داشته‌اند و این گونه نبوده که رویکرد پیشنهادی صرفاً افراد خاصی را برای رسیدگی به نظرات و رفع خطا برگزیند.

شکل ۳ نیز به صورت شهودی حاکی از آن است که تقریباً روند صعودی و نزولی‌بودن امتیازاتی که رویکرد پیشنهادی به توسعه‌دهندگان داده با نظر فرد ارزیاب مطابقت داشته است. به عبارت دیگر در اغلب موارد اگر رویکرد پیشنهادی امتیاز بالایی برای یک توسعه‌دهنده به دست آورده، نتیجه ارزیابی نیز آن فرد را به عنوان توسعه‌دهنده مناسب برای رسیدگی به نظر تشخیص داده است.

۵- نتیجه‌گیری

در این مقاله سعی ما بر این بوده ایراداتی را که از نظر کاربران در برنامک‌های همراه وجود دارد از دل نظراتی که در فروشگاه گوگل نوشته‌اند بیرون بکشیم و برای اولین بار آنها را به یک توسعه‌دهنده مناسب که در آن زمینه فعالیت دارد نسبت دهیم تا به آن رسیدگی کند. برای این منظور پس از کارهای مقدماتی پالایش داده و تحلیل نظرات توسط پردازش زبان طبیعی، با استفاده از داده‌های کامیت‌های برنامک تاریخچه‌ای از عملکرد توسعه‌دهندگان به دست آوردیم و همچنین با استفاده از ایراداتی که توسعه‌دهندگان از قبل در برنامک رفع کرده‌اند در مورد سوابق آنها در رفع خطاهای برنامک اطلاعاتی کسب کردیم. سپس با استفاده از ترکیب این دو معیار به هر توسعه‌دهنده برای رسیدگی به هر نظر امتیازی اختصاص دادیم که نشان‌دهنده صلاحیت آن توسعه‌دهنده از نظر ما در رسیدگی به آن نظر است. در نهایت با توجه به امتیازات کسب‌شده به ازای هر نظر فهرستی از توسعه‌دهندگان ارائه کردیم که به ترتیب اولویت، برای رسیدگی به نظر مناسب هستند. ارزیابی این روش نشان‌دهنده این بود که خروجی این روش می‌تواند در انتساب خطای ذکرشده در نظرات کاربران به توسعه‌دهندگان برنامک‌ها مفید واقع شود و همچنین با توجه به این که روی موضوع این پژوهش تا کنون پژوهش دیگری انجام نشده بود نتیجه می‌گیریم که می‌توان بیشتر روی این موضوع برای بهبود روش پیشنهادی کار کرد و این می‌تواند هدف مهمی برای بهبود کیفیت نرم‌افزار باشد.

با وجود این راه‌های متعددی برای بهبود رویکرد پیشنهادی وجود دارد. برای انتساب خطا از سابقه توسعه‌دهندگان در پروژه‌های دیگر نیز استفاده شود تا اگر توسعه‌دهنده تازه‌واردی به عضویت تیم درمی‌آید سوابقش در پروژه‌های قبلی باعث شود شانس برای انتساب خطا به او وجود داشته باشد. همچنین با این روش اگر یک برنامک به تازگی ساخته شده باشد، کم‌بودن تعداد کامیت‌ها باعث نمی‌شود که نتوانیم انتساب خطا انجام دهیم. علاوه بر این می‌توان نظرات را خوشه‌بندی کرد تا نظرات با مفهوم یکسان در یک خوشه قرار بگیرند و همچنین بتوان به خوشه‌ها اولویت‌دهی کرد تا مشکلات حیاتی‌تر سریع‌تر شناسایی شوند و برای حلشان اقدام صورت گیرد. همچنین می‌توان داده‌هایی را که از آنها استفاده می‌کنیم غنی‌تر کنیم. به عنوان مثال می‌توان با توجه به تغییرات کد پیام کامیت مناسب تولید کرد [۲۸] و همچنین از نظراتی که توسعه‌دهندگان در قسمت Issues می‌گذارند استفاده کرد تا دقت نهایی کار بهبود یابد. علاوه بر این در آینده می‌توان به جای روش tf-idf از روش‌های نوین پردازش متن استفاده کرد.

مراجع

[1] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proc. of the 38th Int. Conf. on Software Engineering, ICSE'16*, pp. 14-24, Austin, TX, USA, 14-22 May 2016.

[2] F. Palomba, *et al.*, "User reviews matter! tracking crowdsourced reviews to support evolution of successful apps," in *Proc. of the 31st IEEE Int. Conf. on Software Maintenance and Evolution, ICSME'15*, pp. 291-300, Bremen, Germany, 29 Sept.-1 Oct. 2015.

[3] A. Ciurumelea, A. Schaufelbuhl, S. Panichella, and H. C. Gall, "Analyzing reviews and code of mobile apps for better release planning," in *Proc. IEEE 24th Int. Conf. on Software Analysis, Evolution and Reengineering, SANER'17*, pp. 91-102, Klagenfurt, Austria, 20-24 Feb. 2017.

[4] A. Di Sorbo, S. Panichella, C. V. Alexandru, C. A. Visaggio, and G. Canfora, "SURF: summarizer of user reviews feedback," in *IEEE/ACM 39th Int. Conf. on Software Engineering Companion, ICSE-C'17*, pp. 55-58, Buenos Aires, Argentina, 20-28 May 2017.

[5] S. Scalabrino, *et al.*, "Listening to the crowd for the release planning of mobile apps," *Trans. on Software Engineering*, vol. 45, no. 1, pp. 1-19, Jan. 2017.

[6] S. Panichella, et al., "ARdoc: app reviews development oriented classifier," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, pp. 1023-1027, Seattle, WA, USA, 15-17 Nov. 2016.

[7] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: mining informative reviews for developers from mobile app marketplace," in *Proc. of the 36th Int. Conf. on Software Engineering, ICSE'14*, pp. 767-778, Hyderabad, India, 31 May-7 Jun. 2014.

[8] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proc. 40th Int. Conf. on Software Engineering, ICSE'18*, pp. 48-58, Gothenburg, Sweden, 27 May-3 Jun. 2018.

[9] C. Gao, *et al.*, "Emerging app issue identification from user feedback: experience on WeChat," in *Proc. IEEE-ACM 41st Int. Conf. Softw. Eng.: Softw. Eng. In Pract.*, pp. 279-288, Montreal, QC, Canada,, 25-31 May 2019.

[10] T. T. Nguyen, A. T. Nguyen, and T. N. Nguyen, "Topic-based, time-aware bug assignment," *ACM SIGSOFT Softw. Eng. Notes*, vol. 39, no. 1, pp. 1-4, Feb. 2014.

[11] J. Park, M. Lee, J. Kim, S. Hwang, and S. Kim, "CosTriage: a cost-aware triage algorithm for bug reporting systems," in *Proc. of the 25th Int. Conf. on Artificial Intelligence, AAAI'11*, pp. 139-144, San Francisco, CA, USA, 7-11 Aug. 2011.

[12] R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani, "Why so complicated? simple term filtering and weighting for location-based bug report assignment recommendation," in *Proc. of the 10th Working Conf. on Mining Software Repositories, MSR'13*, pp. 2-11, Hyderabad, India, 31 May-7 Jun. 2014.

[13] F. Servant and J. A. Jones, "WhoseFault: automatic developer-to-fault assignment through fault localization," in *Proc. of the 34th Int. Conf. on Software Engineering, ICSE'12*, pp. 36-46, Zurich, Switzerland, 2-9 Jun. 2012.

[14] M. Linares-Vasquez, *et al*., "Triaging incoming change requests: bug or commit history, or code authorship?," in *Proc. of the 28th IEEE Int. Conf. on Software Maintenance, ICSM'12*, pp. 451-460, Trento, Italy, 23-28 Sept. 2012.
[15] H. Naguib, N. Narayan, B. Br, and D. Helal, "Bug report assignee recommendation using activity profiles," in *Proc. of the 10th Working Conf. on Mining Software Repositories, MSR'13*, pp. 22-30, San Francisco,, CA, USA, 18-19 May 2013.
[16] H. Hu, H. Zhang, J. Xuan, and W. Sun, "Effective bug triage based on historical bug-fix information," in *Proc. of the 25th IEEE Int. Symp. on Software Reliability Engineering, ISSRE'14*, pp. 122-132, Naples, Italy, 3-6 Nov. 2014.
[17] O. Baysal, M. W. Godfrey, and R. Cohen, "A bug you like: a framework for automated assignment of bugs," in *Proc. of the 17th IEEE Int. Conf. on Program Comprehension, ICPC'09*, pp. 297-298, Vancouver, BC, Canada, 17-19 May 2009.
[18] A. Yadav, S. K. Singh, and J. S. Suri, "Ranking of software developers based on expertise score for bug triaging," *Inf. Softw. Technol.*, vol. 112, pp. 1-17, Aug. 2019.
[19] R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani, "A time-based approach to automatic bug report assignment," *J. Syst. Softw.*, vol. 102, pp. 109-122, Apr. 2015.
[20] X. Xia, D. Lo, X. Wang, and B. Zhou, "Accurate developer recommendation for bug resolution," in *Proc. of the 20th Working Conf. on Reverse Engineering, WCRE'13*, pp. 72-81, Koblenz, Germany, 14-17 Oct. 2013.
[21] R. Wu, H. Zhang, S. Kim, and S. C. Cheung, "ReLink: recovering links between bugs and changes," in *Proc. of the 19th ACM Int. Symp. on the Foundations of Software Engineering, FSE'11*, pp. 15-25, Szeged, Hungary, 5-9 Sept. 2011.
[22] J. Sliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1-5, Jul. 2005.
[23] A. T. Nguyen, T. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Multi-layered approach for recovering links between bug reports and fixes," in *Proc. of the 20th ACM Int. Symp. on the Foundations of Software Engineering, FSE'12*, 11pp., Washington, DC, USA, 19-21 Mar. 2012.
[24] M. Linares-Vasquez, D. Lo, T. B. Le, and M. Linares-v, "RCLinker: Automated Linking of Issue Reports and Commits Leveraging Rich Contextual Information," in *Proc. IEEE 23rd Int. Conf. on Program Comprehension*, 12 pp., Florence, Italy, 18-19 May, 2015.
[25] M. Linares-Vasquez, L. F. Cortes-Coy, J. Aponte, and D. Poshyvanyk, "ChangeScribe: a tool for automatically generating commit messages," in *Proc. of the 37th IEEE/ACM Int. Conf. on Software Engineering, ICSE'15*, pp. 709-712, Firenze, Italy, 16-24 Mar. 2015.
[26] Y. Sun, Q. Wang, and M. Li, "Understanding the contribution of non-source documents in improving missing link recovery," in *Proc. of the 10th ACM/IEEE Int. Symp. on Empirical Software Engineering and Measurement, ESEM'16*, 10 pp., Ciudad Real Spain, 8-9 Sept. 2016.
[27] Y. Sun, Q. Wang, and Y. Yang, "FRLink: improving the recovery of missing issue-commit links by revisiting file relevance," *Inf. Softw. Technol.*, vol. 84, pp. 33-47, Apr. 2017.
[28] T. D. B. Le, M. Linares-Vasquez, D. Lo, and D. Poshyvanyk, "RCLinker: automated linking of issue reports and commits leveraging rich contextual information," in *Proc. of the 23rd IEEE Int. Conf. on Program Comprehension, ICPC'15*, pp. 36-47, Florence, Italy, 18-19 May 2015.

مریم یونسی مدارک کارشناسی و کارشناسی ارشد مهندسی نرم‌افزار خود را به‌ترتیب در سال‌های ۱۳۹۴ و ۱۳۹۶ از دانشگاه تهران و دانشگاه صنعتی شریف دریافت نموده است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: مهندسی نرم‌افزار، و داده‌کاوی در مهندسی نرم‌افزار.

عباس حیدرنوری مدرک دکترای خود در علوم کامپیوتر را در سال ۱۳۸۸ از دانشگاه واترلو کانادا و مدارک کارشناسی و کارشناسی ارشد در مهندسی نرم‌افزار خود را به‌ترتیب در سال‌های ۱۳۷۸ و ۱۳۸۰ از دانشگاه صنعتی شریف دریافت نموده است. بعد از اتمام دوره دکترا، ایشان به مدت یک سال به عنوان پژوهشگر پسادکترا در دانشگاه لوگانو سوییس مشغول به انجام پژوهش بوده است. ایشان سپس به مدت یک سال به عنوان مهندس ارشد نرم‌افزار در زمینه تولید نرم‌افزارهای هوشمند همراه در شرکت Xtreme Labs Inc. در تورنتو کانادا مشغول به کار بوده است. دکتر حیدرنوری در حال حاضر به عنوان عضو هیأت علمی در دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف مشغول به فعالیت می‌باشد. زمینه‌های پژوهشی ایشان عبارتند از: مهندسی نرم‌افزار، کاربردهای هوش مصنوعی و علم داده‌ها در مهندسی نرم‌افزار، و مهندسی نرم‌افزار تجربی.

فاطمه قنادی مدارک کارشناسی و کارشناسی ارشد مهندسی نرم‌افزار خود را به ترتیب در سال‌های ۱۳۹۴ و ۱۳۹۸ از دانشگاه تهران و دانشگاه صنعتی شریف دریافت نموده است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: مهندسی نرم‌افزار، و هوش مصنوعی در مهندسی نرم‌افزار.