پیش‌بینی مفید بودن نظرات بررسی کد با استفاده از الگوریتم‌های یادگیری ماشین

عاطفه محمدی[۱]، محمد امین فضلی[۲]

[۱] کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران
atefemohammadi@ce.sharif.edu
[۲] استادیار، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران
fazli@sharif.edu

چکیده

امروزه بررسی کد همکار[۱] در پروژه های متن‌باز و تجاری، به طور وسیعی استفاده می‌شود. این اصل با تشخیص زودهنگام عیوب کد و نقض استانداردهای کدنویسی در فازهای ابتدایی توسعه، به حفظ کیفیت کد کمک می کند. بر اساس مطالعات انجام‌شده، بخش قابل توجهی از نظرات غیر مفید هستند یعنی منجر به تغییر در کد نمی‌شوند و توسط توسعه‌دهنده نادیده گرفته می‌شوند. بنابراین وجود ابزاری که بتواند به صورت خودکار نظرات مفید را تشخیص دهد تا در زمان توسعه‌دهندگان صرفه‌جویی کند، احساس می‌شود. در این پژوهش ابتدا عوامل مؤثر بر کیفیت نظرات بررسی کد در دو دسته عوامل مربوط به تجربه توسعه‌دهنده و ویژگی‌های متنی نظرات استخراج شد. سپس با توجه به عدم وجود مجموعه داده[۲] مناسبی که شامل این عوامل باشد یک مجموعه داده جمع‌آوری شد. در مرحله بعد یک مدل پیش‌بینی‌کننده نظرات مفید با کمک الگوریتم XGBoost پیاده‌سازی و عملکرد آن با سایر کارهای انجام‌شده در این زمینه مقایسه شد. نتایج به‌دست‌آمده نشان می‌دهد که روش پیشنهادی با در نظر گرفتن دو مجموعه داده مجزا و با توجه به معیارهای صحت[۳]، فراخوانی[۴] و امتیاز اف-وان[۵] حدود سه درصد و با توجه به معیار دقت[۶] حدود یک درصد نسبت به تنها روش موجود، بهتر عمل کرده است.

کلمات کلیدی

بررسی کد همکار، نگه‌داری نرم‌افزار، ویژگی‌های متنی، تجربه توسعه‌دهنده، یادگیری‌ماشین

۱- مقدمه

امروزه با توجه به افزایش نرم‌افزارهای متن‌باز و تجاری، رقابت برای ماندن در دنیای کسب‌وکار شدت گرفته است. تا زمانی که یک نرم‌افزار، مناسب با نیازمندی‌های کاربران تغییر کند، به اصطلاح زنده است و می‌تواند در این رقابت باقی بماند؛ بنابراین فاز نگهداری برای ایجاد تغییر در نرم‌افزار برای رفع نیازهای کاربران، یک فاز ضروری است. مطالعات نشان می‌دهد بیش از ۷۰ درصد هزینه‌های مربوط به ایجاد نرم‌افزار صرف فاز نگهداری نرم‌افزار می‌شود [17]. برای کاهش هزینه‌های مربوط به این فاز لازم است تا از ایجاد خطا در نرم‌افزار جلوگیری شود. یکی از راه‌های جلوگیری از ایجاد خطا در نرم‌افزار به‌کارگیری بررسی کد همکار است. مطالعات نشان می‌دهد بررسی کد همکار باعث صرفه‌جویی در بیش از ۵۰ درصد از هزینه‌های نگه‌داری نرم‌افزار می‌شود [2]. اما حجم قابل توجهی از نظرات بررسی کد مفید نبوده (۷ تا ۳۵ درصد) و باعث ایجاد تغییرات در کد نمی‌شود [14]؛ بنابراین ساخت ابزاری که با شناسایی عوامل مؤثر بر مفید بودن نظرات، به‌صورت خودکار نظرات مفید (نظراتی که موجب تغییر در کد می‌شوند) را شناسایی کند، موجب ذخیره زمان و تلاش افراد درگیر در پروژه می‌شود.

در [3] عوامل مؤثر در مفید بودن نظرات بررسی کد در مایکروسافت بررسی شده است. طبق تعریف این مقاله اگر نظری منجر به تغییر در همسایگی ۱۰ خطی آن بخش از کدی که مورد بررسی قرار گرفته است، شود به عنوان نظر مفید تلقی می‌شود. در [16] به پیش‌بینی مفید بودن نظرات بررسی کد با استفاده از ویژگی‌های متنی و عوامل مربوط به تجربه بازنگر پرداخته شده است. تعریف ارائه‌شده برای تعیین مفید بودن نظرات همان تعریف ارائه‌شده در [3] است.

در [10] کیفیت بررسی کد و تأثیر افراد در بررسی کد در موزیلا مورد بررسی قرار گرفت. بر اساس مطالعه صورت گرفته ۵۴ درصد از تغییرات مورد بررسی، اشکالات[۷] کد را نشان می‌دهد. همچنین در این مقاله تأثیر ویژگی‌های مختلف در کیفیت بررسی کد و تشخیص اشکالات کد توسط بازنگر مورد بررسی قرار گرفت. در [11] عوامل فنی[۸] و غیر فنی مؤثر بر فرایند

بررسی کد با تمرکز بر زمان و مثبت بودن (خروجی) در پروژه‌های بلینک و وب‌کیت مورد بررسی قرار گرفت. در [24,4] نیز به بررسی عوامل مؤثر در کیفیت بررسی کد پرداخته شده است.

در [27] به بررسی عواملی که بررسی کد را آسان‌تر می‌کند، پرداخته شده است. عواملی که مرتبط با درک بازنگر از کد و ساختار تغییرات است. برای استخراج این عوامل از نتایج مقالات، بلاگ‌های منتشرشده و از ۱۰ توسعه-دهنده پروژه‌های متن‌باز و تجاری کمک گرفته شد. در نهایت ابزاری توسعه داده شد که توسعه‌دهندگان به تغییرات اعمال‌شده از نظر قابلیت بررسی امتیاز اختصاص دادند. مهم‌ترین عوامل تعیین‌کننده در قابلیت بررسی بر اساس این مقاله توضیح تغییرات (مستندسازی مناسب درباره تغییرات و تأثیرات آن‌ها، نحوه به‌کارگیری تغییرات به کمک مثال)، اندازه تغییرات (و تعداد فایل‌های تغییریافته) تاریخچه اعمال‌ها[۹] (تعداد زیادتر اعمال‌ها زمان بررسی را بالا و احتمال پذیرش آن را پایین می‌آورد) بود.

در [19] پیشنهاد شد که تغییرات اعمال‌شده در کد اغلب شامل یک کلاس برجسته[۱۰] است که اکثر تغییرات در این کلاس اعمال شده است و ایجاد تغییرات در این کلاس باعث ایجاد تغییرات در کلاس‌های وابسته به این کلاس می‌شود؛ بنابراین اگر به بازنگر در تعیین کلاس برجسته کمک شود، تغییرات اعمال‌شده را بهتر درک کرده و در نتیجه نظرات مفیدتری تولید می‌کند. در این مقاله با استفاده از الگوریتم جنگل تصادفی کلاس برجسته استخراج شد. در [5] چالش‌های بررسی کد از نظر بازنگرها و توسعه‌دهندگان در پروژه‌های مایکروسافت مورد بررسی قرار گرفت. در این مقاله پس از مصاحبه‌ای که با بازنگرها و توسعه‌دهندگان انجام شد، سعی شد اصولی برای از بین بردن این چالش‌ها تدوین شود که بازنگرها و توسعه‌دهندگان ضمن رعایت آن‌ها بتوانند بر این چالش‌ها فائق آیند.

در [23] به بررسی عوامل مؤثر در تصمیم‌گیری بازنگرها برای پذیرش یا عدم پذیرش انجام بررسی تغییرات کد پرداخته شده است. دلیل اصلی این بررسی این بود که طبق مطالعات انجام‌شده مشارکت کم بازنگرها در امر بررسی کد باعث کیفیت کم کد می‌شود. در این مقاله عوامل مؤثر بر بررسی کد در سه دسته فاکتورهای انسانی مرتبط با بار کاری و تعاملات اجتماعی بازنگر، تجربه بازنگر و خصوصیات مربوط به تغییرات مورد بررسی قرار گرفت. نویسندگان مقاله به این نتیجه رسیدند که حدود ۱۶ تا ۶۶ درصد تغییرات حداقل یک بازنگر دعوت‌شده دارند که در امر بررسی، همکاری نداشته است. بررسی انجام‌شده نشان داد که دعوت تعداد زیادی برنامه‌نویس به عنوان بازنگر احتمال اینکه تغییرات از طرف بازنگرها پاسخ دریافت نکند را افزایش می‌دهد. برای بررسی تأثیر فاکتورهای انسانی در تصمیم بازنگرها برای مشارکت در بررسی کد دو مدل با استفاده از رگرسیون لجستیک ساخته شد که مدل اول با در نظر گرفتن هر سه دسته فاکتور و مدل دوم بدون در نظر گرفتن فاکتورهای انسانی ساخته شد با بررسی انجام‌شده مدل اول کارایی بهتری داشت و این موضوع نشان‌دهنده تأثیر فاکتورهای انسانی است.

بر اساس [16] احساساتی که در نظرات بررسی کد وجود دارند نیز در مفید یا نامفید بودن نظر تأثیرگذار است. تنها ابزاری که به استخراج احساسات نظرات بررسی کد پرداخته است ابزار SentiCR [20] است. این ابزار روی ۲۰۰۰ نظر بررسی کد موجود در ۵۰ پروژه که در بستر گریت ایجاد شده‌اند، آموزش دیده شده است. این ابزار احساسات را به دو دسته مثبت و خنثی و یا منفی تفکیک کرده است. در این ابزار پیش‌پردازش‌هایی صورت گرفته است که باعث می‌شود توانایی بهتری در تشخیص احساس نظرات کد داشته باشد.

تنها روشی که تاکنون برای پیش‌بینی نوع نظرات ارائه شده است RevHelper [16] است. در این روش تنها از ۱۴۸۲ نظر روی چهار پروژه متن‌باز پایتون استفاده شده است که با توجه به حجم کم مجموعه داده، قابلیت تعمیم این روش به چالش کشیده می‌شود. در همین راستا در این پژوهش سعی شد تا نظرات مربوط به پروژه‌های متن‌باز پایتون از تعداد بیشتری پروژه استخراج و یک مدل پیش‌بینی‌کننده برای تعیین نوع نظرات پیاده‌سازی شود. بستری که از آن برای استخراج نظرات پرداخته شد وب‌گاه گیت‌هاب است.

این پژوهش دربردارنده نوآوری‌های زیر در زمینه بررسی کد همکار است:

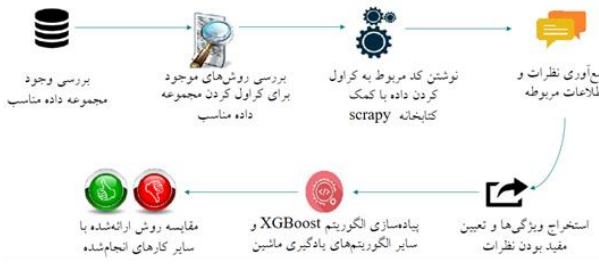
بررسی ویژگی‌های متفاوت در زمینه تجربه بازنگر و ویژگی‌های متنی نظر در زمینه ایجاد نظرات مفید

استخراج مجموعه داده مناسب با حجم زیاد داده نسبت به مجموعه داده موجود به منظور تعیین مفید یا نامفید بودن نظرات

پیاده‌سازی یک مدل پیش‌بینی‌کننده با عملکرد بهتر نسبت به تنها مدل پیش‌بینی‌کننده موجود

۲- روش پیشنهادی

در این بخش به بررسی راهکار پیشنهادی می‌پردازیم. ابتدا با توجه به عدم وجود مجموعه داده مناسب برای ساخت مدل پیش‌بینی‌کننده به جمع‌آوری مجموعه داده مناسب و استخراج ویژگی‌های مناسب پرداخته شد. سپس به پیاده‌سازی الگوریتم XGBoost و انواع شبکه‌های عصبی برای پیش‌بینی نظرات مفید پرداخته شد با توجه به بررسی‌های صورت‌گرفته الگوریتم XGBoost نسبت به بقیه روش‌ها و RevHelper توانایی بیشتری در پیش‌بینی نظرات داشت. روش پیشنهادی در شکل (۱) نشان داده شده است.



شکل (۱): روش پیشنهادی در یک نگاه

لازم به ذکر است در این پژوهش منظور از نظر مفید مانند تعریف بیان-شده در [16,3]، نظری است که منجر به تغییر در کد در همسایگی ۱۰ خطی آن بخش از کدی که مورد بررسی قرار گرفته است، شود.

۲-۱- جمع‌آوری مجموعه داده

با توجه به اینکه برای پیاده‌سازی مدل پیش‌بینی‌کننده به ویژگی‌های مناسب و مجموعه داده‌ای که بتوان ویژگی‌های مورد نظر را از آن استخراج کرد نیاز است. در ابتدا با بررسی کارهای انجام‌شده در زمینه بررسی کد به انتخاب ویژگی‌های مناسب و بررسی وجود مجموعه داده مناسب برای پیاده‌سازی مدل پیش‌بینی‌کننده پرداخته شد. ویژگی‌های مورد نظر شامل عوامل بیان‌شده در

[16] که در جدول (۱) ذکر شدند و احساس نظرات هستند. عوامل ذکرشده در بقیه منابع به دلیل تکراری بودن و یا عدم دسترسی به عوامل بیان‌شده هنگام ایجاد نظر در نظر گرفته نشدند. برای مثال وضعیت موضوع یکی از ویژگی‌هایی است که تا اتمام فرایند بررسی کد مشخص نمی‌شود.

برای استخراج احساس موجود در نظر از ابزار sentiCR [20] استفاده شد تا احساس نظرات با دقت بالایی شناسایی شود.

جدول (۱): ویژگی‌های ارائه شده در [16]

ویژگی	نوع ویژگی	توضیح ویژگی	تأثیر
خوانایی	متنی	قابل فهم بودن متن نظر	ندارد
خوانایی (حذف عناصر کد از نظرات)	متنی	قابل فهم بودن نظر بدون در نظر گرفتن عناصر کد	ندارد
نسبت پرسش	متنی	نسبت جملات پرسشی به کل جملات نظر بررسی	ندارد
نسبت عناصر کد	متنی	نسبت عناصر کد به کل کلمات نظر بررسی	در نظرات مفید بیشتر
نسبت کلمات ایست	متنی	نسبت کلمات ایست به کل کلمات نظر بررسی	در نظرات مفید کمتر
نسبت کلمات ایست و کلیدی	متنی	نسبت کلمات ایست و کلیدی به کل کلمات نظر	در نظرات مفید کمتر
شباهت معنایی	متنی	شباهت کسینوسی نظر و خط کد تغییر کرده	در نظرات مفید بیشتر
دوباره بررسی-کرده[11]	تجربه بازنگر	بازنگر در فایل مورد بررسی، قبلاً تغییری نوشته؟	باعث ایجاد نظرات مفید
تعداد اعمال‌های نوشته‌شده[12]	تجربه بازنگر	تعداد کل تغییراتی که بازنگر اعمال کرده	باعث ایجاد نظرات مفید
شباهت با کتابخانه-های خارجی	تجربه بازنگر	میزان آشنایی بازنگر با کتابخانه‌های فایل	ندارد
تعداد درخواست‌های کشیدن بررسی-شده[13]	تجربه بازنگر	تعداد کل درخواست‌هایی که بازنگر، بررسی کرده	ندارد
تعداد اعمال‌های بررسی‌شده[14]	تجربه بازنگر	تعداد کل تغییراتی که بازنگر، بررسی کرده	باعث ایجاد نظرات مفید
بررسی اعمال‌های فایل[15]	تجربه بازنگر	تعداد فایل‌های این درخواست که بازنگر قبلاً نظر داده	باعث ایجاد نظرات مفید
نویسنده اعمال‌های فایل[16]	تجربه بازنگر	تعداد فایل‌های این درخواست که بازنگر قبلاً روی آن‌ها تغییری داده	باعث ایجاد نظرات مفید

سپس با در نظر گرفتن این ویژگی‌ها به بررسی وجود مجموعه داده‌ای که شامل این ویژگی‌ها باشد یا بتوان این ویژگی‌ها را از آن استخراج کرد، پرداخته شد.

با بررسی‌های انجام‌شده دو مجموعه داده برای بررسی کد یافته شد [16,13]. مجموعه داده ارائه‌شده در [13] تقریباً شامل پنج میلیون و پانصد هزار نظر بررسی کد است. اما این مجموعه داده شامل ویژگی‌های مربوط به درخواست‌های کشیدن نیست، بنابراین برخی ویژگی‌ها شامل تعداد درخواست-های کشیدن بررسی‌شده، بررسی اعمال‌های فایل، نویسنده اعمال‌های فایل قابل استخراج نبودند بنابراین این مجموعه داده کنار گذاشته شد. از مجموعه داده ارائه‌شده در [16] تمام ویژگی‌های در نظر گرفته‌شده قابل استخراج است ولی متأسفانه این مجموعه داده تنها شامل حدود ۱۴۸۲ نظر بررسی کد هست و چون در این مقاله نیاز به بررسی عملکرد شبکه‌های عصبی بود و چون شبکه‌های عصبی روی مجموعه داده کوچک به نتایج خوبی نمی‌رسند و به مجموعه داده بزرگی نیاز دارند و علاوه بر این با این مجموعه داده کوچک احتمال تعمیم نتایج زیر سؤال می‌رود، این مجموعه داده نیز قابل استفاده نبود. بنابراین در نهایت به استخراج مجموعه داده مناسب پرداخته شد.

برای استخراج نظرات با کمک کتابخانه scrapy پایتون به استخراج داده‌های مناسب از مخازن کد وب‌گاه گیت‌هاب پرداخته شد. مخازن کد انتخاب‌شده، پروژه‌های متن‌بازی هستند که به زبان پایتون نوشته شده‌اند؛ بنابراین نظرات استخراج‌شده، نظراتی هستند که روی فایل‌های پایتون گذاشته شده‌اند. برای اینکه نتایج بررسی قابل تعمیم باشند و محدود به یک پروژه خاص نباشند، از نظرات یازده پروژه متن‌باز استفاده شد. جدول (۲) تعداد نظرات استخراج‌شده از هر پروژه را نشان می‌دهد.

جدول (۲): نظرات استخراج‌شده از وب‌گاه‌ها

شماره	نام پروژه	تعداد نظرات استخراج‌شده
۱	[1] ansible	۷۱۸۸
۲	[8] django	۹۴۲۴
۳	[9] flask	۳۱۷
۴	[12] keras	۳۰۴۴
۵	[15] pandas	۱۷۶۷۱
۶	[18] pytorch	۱۱۹۵۴
۷	[21] sentry	۵۱۴۳
۸	[22] tensorflow	۲۹۱۷
۹	[25] Tornado	۲۹۱
۱۰	[28] youtube-dl	۳۳۵۴
۱۱	[29] Zulip	۴۹۶۷

در نهایت ۶۶۲۷۰ نظر و ویژگی‌های در نظرگرفته‌شده جمع‌آوری شد. از داده‌های استخراج‌شده حدود ۳۸ درصد نظرات مفید و بقیه نظرات غیر مفید بودند.

از بین ویژگی‌های استخراج‌شده از مجموعه داده بین تعدادی از آن‌ها وابستگی مستقیم وجود داشت برای مثال بین ویژگی دوباره اعمال کرده و تعداد اعمال‌های نوشته‌شده رابطه مستقیم وجود داشت؛ بنابراین مهم‌ترین ویژگی‌ها با کمک تابع feature_importances_ روی مدل حاصل از الگوریتم XGBoost [26] که با ۵۰۳۲۳ داده آموزشی، آموزش داده شده بود، استخراج شد. ویژگی‌های بیان‌شده در جدول (۳) به عنوان ویژگی‌های مدل پیش‌بینی کننده در نظر گرفته شدند.

جدول (۳): ویژگی‌های در نظر گرفته‌شده برای پیاده‌سازی مدل پیش‌بینی‌کننده

نوع ویژگی	ویژگی‌های مورد استفاده
متنی	شباهت معنایی- نسبت کلمات ایست
تجربه بازنگر	تعداد درخواست‌های کشیدن بررسی‌شده-تعداد اعمال‌های بررسی شده-بررسی اعمال‌های فایل-نویسنده اعمال‌های فایل- تعداد اعمال-های نوشته‌شده

۲-۲- پیاده‌سازی روش‌های دسته‌بندی

با توجه به ویژگی‌های مجموعه داده (تعداد نمونه‌های آموزشی زیاد، تعداد ویژگی‌ها کم و ویژگی‌ها از نوع ویژگی‌های عددی[17] یا طبقه‌ای[18])، دسته-

بندهای مناسب برای پیش‌بینی نظرات، الگوریتم XGBoost و شبکه‌های عصبی هستند (شرط اول مجموعه داده برای استفاده از این روش به تنهایی کافی است). در این بخش به پیاده‌سازی الگوریتم XGBoost و شبکه‌های عصبی و انتخاب دسته‌بند مناسب می‌پردازیم.

در این بخش برای اطمینان از انتخاب دسته‌بند مناسب از بخشی از مجموعه داده شامل ۵۰۳۲۳ داده آموزشی برای مقایسه و بهبود انواع روش‌ها استفاده شد. برای آموزش مدل‌ها از ۷۵ درصد این بخش از مجموعه داده و برای آزمون مدل‌ها از ۲۵ درصد این بخش از مجموعه داده استفاده شد. دلیل عدم استفاده از تمام مجموعه داده جلوگیری از اینکه روش انتخابی نهایی نسبت به مجموعه داده دچار بیش‌برازش[۱۹] شود، بود.

الگوریتم XGBoost برای ساخت و بهبود درخت‌ها از پارامترهای مختلفی استفاده می‌کند که این پارامترها می‌توانند مقادیر مختلفی بگیرند. برای اینکه این الگوریتم بتواند به خوبی نتایج را پیش‌بینی کند این پارامترها باید به طور مناسبی تنظیم شوند بنابراین ابتدا پارامترهای مربوط به این الگوریتم تنظیم شدند. مقادیر تنظیم شده برای پارامترها در جدول (۴) گزارش شده است.

جدول (٤): پارامترهای تنظیم‌شده برای الگوریتم XGBoost

پارامتر	مقدار تنظیم‌شده
max_depth	۱۰
min_child_weigh	۱
Gamma	۰/۷
Eta	۰/۰۵
Subsample	۰/۵۵
colsample_bytree	۰/۹۵
Alpha	۰/۰۰۱
Eta	۰/۰۵
n_estimators	۱۰۰۰

برای مقایسه انواع روش‌ها علاوه بر معیار دقت، معیارهای صحت، فراخوانی و امتیاز اف-وان مورد استفاده قرار گرفته است. معیار دقت زمانی که مجموعه داده نامتوازنی در اختیار داشته باشیم، به تنهایی معیار مناسبی برای انتخاب روش دسته‌بند مناسب نیست به همین دلیل از معیارهای دیگر نیز استفاده شد. برای مثال اگر یک دسته‌بند در تمامی مواقع، نوع نظر را غیر مفید تلقی کند با توجه به اینکه ۶۲ درصد نظرات این مجموعه داده غیرمفید هستند به دقت ۶۲ درصد می‌رسد در حالی‌که می‌دانیم این یک مدل پیش‌بینی مناسب برای پیش‌بینی نوع نظرات نخواهد بود؛ بنابراین معیارهای دیگر نیز در نظر گرفته شدند. معیار صحت بیان می‌کند که از بین نظراتی که به عنوان نظر مفید (/غیر مفید) توسط مدل پیش‌بینی شدند چه نسبتی از آن‌ها واقعاً نظر مفید (/غیر مفید) هستند. معیار فراخوانی بیان می‌کند از بین نظراتی که مفید (/غیر مفید) هستند چه نسبتی از این نظرات توسط مدل پیش‌بینی‌کننده مفید (/غیر مفید) در نظر گرفته شده‌اند. لازم به ذکر است مقادیر ذکرشده در جداول مختلف بر اساس درصد هستند.

با توجه به اینکه هدف بهبود عملکرد پیش‌بینی‌کننده نسبت به RevHelper [16] است، ابتدا الگوریتم به‌کاررفته در این روش که جنگل تصادفی بود، پیاده‌سازی شد تا نتایج حاصل از پیاده‌سازی روش این مقاله روی مجموعه داده در نظرگرفته‌شده، مشخص شود. نتایج حاصل از پیاده‌سازی انواع روش‌ها در جدول (۵) قابل مشاهده است. از آنجایی که در ساخت مدل XGBoost و مدل جنگل تصادفی تعدادی درخت به طور تصادفی ساخته می‌شوند پس نتیجه این الگوریتم‌ها به ترتیب انتخاب ویژگی‌ها و ساخت درخت وابسته است؛ بنابراین برای اطمینان از نتایج، این الگوریتم‌ها ده بار اجرا شدند و نتایج بیان‌شده در جداول میانگین نتایج ده بار اجرای این الگوریتم-هاست. از آنجایی که می‌توان شبکه‌های عصبی را با انواع معماری‌ها پیاده‌سازی کرد در بخش ۲-۳- به توضیح انواع معماری شبکه‌های عصبی استفاده‌شده و نتایج حاصل از پیاده‌سازی این شبکه‌ها پرداخته شده است. همان‌طور که بعداً بحث خواهد شد، در این پژوهش شبکه‌های عصبی نسبت به روش جنگل تصادفی به نتایج خوبی دست نیافتند.

همان‌گونه که از جدول (۵) مشخص است، با توجه به پیاده‌سازی این روش‌ها مشاهده می‌شود که دسته‌بند XGBoost بهترین عملکرد را در زمینه پیش‌بینی هر دو نوع نظرات داراست. بقیه دسته‌بندها با توجه به اینکه بیشتر نظرات موجود در مجموعه داده غیر مفید هستند، ممکن است از لحاظ برخی معیارها در پیش‌بینی نظرات غیر مفید بهتر عمل کرده باشند، اما چون هدف یافتن روشی است که هر دو نظر را به خوبی پیش‌بینی کند این روش‌ها، انتخاب مناسبی برای پیاده‌سازی مدل پیش‌بینی‌کننده نهایی نیستند؛ بنابراین XGBoost برای استفاده در پیش‌بینی‌کننده نهایی مورد استفاده قرار می-گیرد.

جدول (۵): نتایج روی بخشی از مجموعه داده برای اطمینان از انتخاب مدل پیش‌بینی‌کننده نهایی

الف) نظرات مفید

مدل پیش‌بینی‌کننده	نظرات مفید		
	F1-score	recall	precision
RevHelper (جنگل تصادفی)	۶۵/۳۶	۷۱/۴۵	۶۰/۲۳
XGBoost	۶۶/۹	۷۳/۴	۶۱/۴۶
شبکه کاملاً متصل	۴۵/۹۹	۳۹/۰۷	۵۵/۸۹
شبکه GRU	۰	۰	۰
شبکه LSTM	۴۹	۷۸	۳۶
شبکه ترکیبی	۴۶/۴۹	۵۵/۸۸	۳۹/۸

ب) نظرات غیر مفید و تمام نظرات

مدل پیش‌بینی‌کننده	نظرات غیر مفید			تمام نظرات
	F1-score	recall	precision	accuracy
RevHelper (جنگل تصادفی)	۸۱	۷۷	۸۵	۷۵/۲۷
XGBoost	۸۲	۷۸	۸۶	۷۶/۴۵
شبکه کاملاً متصل	۷۴	۸۱	۶۸	۶۴/۴۶
شبکه GRU	۷۶	۶۲	۱	۶۱/۶۲
شبکه LSTM	۵۸/۲۳	۴۴/۸۲	۸۳/۱	۵۴/۲۵
شبکه ترکیبی	۷۴	۶۸	۸۰	۶۵/۴۷

۲-۳- انواع معماری شبکه‌های عصبی

برای پیاده‌سازی شبکه عصبی می‌توان معماری‌های مختلفی را در نظر گرفت. در این بخش به بررسی شبکه‌های پیاده‌شده در این پژوهش پرداخته شده است (برای هر نوع شبکه، شبکه‌ای که بهترین نتایج را تولید کرده است، توضیح داده شده است). لازم به ذکر است برای پیاده‌سازی شبکه‌های عصبی

از کتابخانه کراس [12] در زبان پایتون استفاده شده است. در ابتدا با کمک ویژگی‌های متنی و تجربه نویسنده و به وسیله شبکه‌های کاملاً متصل[۲۰]، عملکرد این نوع شبکه‌ها مورد ارزیابی قرار گرفت. نتایج نشان‌دهنده این بود که این شبکه‌ها به خوبی قادر به تشخیص نوع نظرات نیستند؛ بنابراین سعی شد تا با پیاده‌سازی شبکه‌های GRU [۲۱] و LSTM [۲۲] تنها با کمک متن نظرات این نوع شبکه‌ها مورد ارزیابی قرار گیرند و در نهایت با ترکیب شبکه‌های کاملاً متصل با LSTM یا GRU، عملکرد این نوع شبکه‌ها نیز مورد ارزیابی قرار گرفت. نتایج حاصل از پیاده‌سازی این روش‌ها در جدول ۵ آمده است.

۲-۳-۱- شبکه عصبی کاملاً متصل

آنچه در این نوع شبکه اهمیت دارد؛ تعداد مناسب لایه‌ها، واحدهای هر لایه و تابع فعال‌ساز هر لایه است از این‌رو معماری‌های متفاوت با عملکردهای متفاوتی ایجاد شد. برای جلوگیری از بیش‌برازش سعی شد تا معماری شبکه‌ها تا حد امکان ساده نگه‌داشته شود یعنی تعداد لایه‌ها و واحدهای هر لایه کم در نظر گرفته شود. علاوه بر این نیز از تنظیم‌کننده وزن‌ها[۲۳] و لایه dropout نیز استفاده شد.

در نهایت با آزمون چندین نوع شبکه کاملاً متصل با انواع معماری‌ها، معماری به شرح زیر به بهترین نتایج دست یافت:

لایه اول: ۱۶ واحد با تنظیم‌کننده وزن L2 و مقدار ۰/۰۰۱
لایه دوم: لایه dropout با میزان ۰/۲
لایه سوم: ۳۲ واحد با تنظیم‌کننده وزن L2 با مقدار ۰/۰۰۱
لایه چهارم: لایه dropout با میزان ۰/۲
لایه پنجم: ۱۰ واحد با تنظیم‌کننده وزن L2 با مقدار ۰/۰۰۱
لایه ششم: لایه dropout با میزان ۰/۲
لایه هفتم: ۴ واحد با تنظیم‌کننده وزن L2 و مقدار ۰/۰۰۱
لایه هشتم: یک واحد با تنظیم‌کننده وزن L2 با مقدار ۰/۰۰۱ برای محاسبه خروجی

۲-۳-۲- شبکه‌های عصبی GRU

در این بخش عملکرد شبکه عصبی GRU با استفاده از متن نظرات مورد ارزیابی قرار می‌گیرد. در این مدل برای تبدیل متن نظرات به بردارهای عددی از روش رمزگذاری وان–هات استفاده شد.

برای بررسی عملکرد این نوع شبکه، معماری‌های متفاوتی در نظر گرفته شد معماری که در این نوع شبکه به بهترین نتایج دست یافت، معماری به شرح زیر بود:

لایه اول: از نوع GRU و با ۱۶ واحد
لایه دوم: یک واحد با تنظیم‌کننده وزن L2 با مقدار ۰/۰۰۱ به منظور محاسبه خروجی

با توجه به پیاده‌سازی معماری‌های مختلف GRU و بهترین معماری GRU، GRU در پیش‌بینی نظرات مفید، خوب عمل نمی‌کند و تمامی نظرات را غیرمفید پیش‌بینی می‌کند یعنی نسبت به مجموعه داده کاملاً دچار بیش‌برازش شده است.

۲-۳-۳- شبکه‌های عصبی LSTM

در این بخش عملکرد شبکه‌های LSTM با استفاده از متن نظرات مورد ارزیابی قرار می‌گیرد. بهترین معماری که مورد بررسی قرار گرفت، معماری با مشخصات زیر بود:

لایه اول: لایه embedding با ۱۵ واحد
لایه دوم: لایه LSTM با ۳۲ واحد
لایه سوم: لایه dense با یک واحد به منظور محاسبه خروجی

همان‌گونه که از نتایج پیاده‌سازی شبکه‌های عصبی کاملاً متصل، LSTM و GRU مشخص است، این شبکه‌ها توانایی جنگل تصادفی در تشخیص هر دو نوع نظر را نداشتند؛ بنابراین در ادامه با تلفیق این شبکه‌ها سعی شد تا قدرت شبکه‌ها افزایش یابد.

۲-۳-۴- ترکیب انواع مختلف شبکه‌های عصبی

در این بخش با ترکیب شبکه‌های کاملاً متصل با LSTM یا GRU شبکه‌های جدیدی ایجاد. اگرچه در پیاده‌سازی انواع معماری‌های شبکه GRU نتایج خوبی به دست نیامد، اما باز در ترکیب شبکه‌ها این شبکه نیز مورد بررسی قرار گرفت. برای ساخت این شبکه‌ها سعی شد تا از نتایج به‌دست‌آمده در مراحل قبل نیز استفاده شود و در تلفیق شبکه‌ها برای انتخاب هر نوع شبکه، شبکه‌ای انتخاب شده است تا با توجه به بررسی‌های قبل هم نتایج خوبی در پیش‌بینی هر دو نظر داشته باشد و هم معماری ساده‌ای داشته باشد تا زمان آموزش و پارامترهایی که باید آموزش دیده شوند، کاهش یابد.

در ترکیب شبکه‌ها ابتدا در شبکه‌های LSTM با تبدیل متن نظرات به بردار با کمک روش word2vec [7] شبکه‌های ترکیبی ساخته شد. سپس با روش doc2vec [6] یک مدل doc2vec آموزش داده شد که متن نظرات را به بردارهایی به ابعاد ۳۰۰ تبدیل می‌کرد و از این بردارها به عنوان ورودی LSTM (یا GRU) استفاده شد. در نهایت با ترکیب شبکه‌های LSTM یا GRU با شبکه‌های کاملاً متصل (ورودی شبکه‌های کاملاً متصل ویژگی‌های استخراج‌شده از مجموعه داده بود) معماری‌های ترکیبی جدیدی ساخته شد. بهترین معماری که مورد بررسی قرار گرفت، معماری به شرح زیر بود:

بخش LSTM با ورودی به ابعاد ۳۰۰(استفاده از روش doc2vec برای ایجاد بردار):

لایه اول: لایه LSTM با ۳۲ واحد

بخش کاملاً متصل با ورودی به تعداد ویژگی‌های در نظر گرفته‌شده (ویژگی‌های مبتنی بر متن نظر و تجربه نویسنده):

لایه اول: لایه dense با ۱۶ واحد
لایه دوم: لایه dense با ۱۰ واحد
لایه سوم: لایه dense با ۴ واحد

بخش ترکیب‌کننده:

لایه اول: لایه dense با ۲ واحد
لایه دوم: لایه dense با ۱ واحد برای محاسبه خروجی

با توجه به نتایج حاصل از پیاده‌سازی‌های صورت‌گرفته حتی با ترکیب انواع شبکه‌های عصبی که به منظور قدرتمند کردن شبکه‌ها انجام شد؛ هیچ‌کدام از این شبکه‌ها توانایی جنگل تصادفی در تشخیص هر دو نوع نظر را نداشتند.

۳- نتیجه‌گیری

در این بخش به مقایسه نتایج حاصل از پیاده‌سازی مدل پیش‌بینی‌کننده نهایی که با کمک الگوریتم XGBoost آموزش دیده است با سایر کارهای انجام-شده در این زمینه می‌پردازیم. همان‌طور که قبلاً هم اشاره شد تنها کار انجام-شده در زمینه پیش‌بینی مفید بودن نظرات بررسی کد محدود به RevHelper [16] است، بنابراین در این بخش به مقایسه روش پیشنهادی و RevHelper می‌پردازیم. روش به‌کاررفته در RevHelper استفاده از الگوریتم جنگل تصادفی بود، این روش با ویژگی‌هایی که در همان مقاله برای ساخت مدل پیش‌بینی‌کننده استفاده شده بود، پیاده‌سازی شد. ویژگی‌های مورد استفاده در الگوریتم XGBoost همان ویژگی‌های بیان‌شده در جدول (۳) هستند.

برای مقایسه این دو روش از دو مجموعه داده مجزا استفاده شد:

۱) مجموعه داده ارائه شد در [16]: از ۱۱۱۶ نظر و ویژگی‌های مربوط به آن برای آموزش مدل‌ها و از ۳۶۶ نظر برای آزمون مدل‌ها استفاده شد. این مجموعه داده نامتوازن بوده و حدود ۳۰ درصد از نظرات را نظرات غیر مفید تشکیل می‌دهند؛ بنابراین در این مجموعه داده پیش‌بینی نظرات غیر مفید از درجه اهمیت بالاتری برخوردار است. نتایج حاصل از پیاده‌سازی روش پیشنهادی و RevHelper روی این مجموعه داده در جدول (۶) قابل مشاهده است.

۲) مجموعه داده جمع‌آوری‌شده در این پژوهش: داده‌های آموزشی شامل ۵۰۳۲۳ نظر و ویژگی‌های مربوط به این نظرات بودند که برای مقایسه انواع روش‌های دسته‌بندی به کار گرفته شدند. داده-های آزمون بخشی از مجموعه داده شامل ۱۵۹۴۷ نظر و ویژگی‌های مربوط به این نظرات بودند که در هیچ‌یک از مراحل قبل مورد استفاده قرار نگرفته بودند. نتایج حاصل از پیاده‌سازی روش پیشنهادی و RevHelper روی این مجموعه داده در جدول(۷) قابل مشاهده است. همان‌گونه که قبلاً هم ذکر شد، مجموعه داده جمع‌آوری‌شده نامتوازن است و حدود ۳۸ درصد از نظرات مفید هستند بنابراین در این مجموعه داده پیش‌بینی درست نظرات مفید از اهمیت بالاتری برخوردار است.

همان‌طور که اشاره شد برای ساخت مدل نهایی از الگوریتم XGBoost استفاده شد. از آنجایی که در ساخت این مدل و مدل جنگل تصادفی تعدادی درخت به طور تصادفی ساخته می‌شوند پس نتیجه این الگوریتم‌ها به ترتیب انتخاب ویژگی‌ها و ساخت درخت وابسته است؛ بنابراین برای اطمینان از نتایج، این الگوریتم‌ها ده بار اجرا شدند و نتایج بیان‌شده در جداول میانگین نتایج ده بار اجرای این الگوریتم‌هاست.

همان‌گونه که از نتایج گزارش‌شده مشخص است با در نظر گرفتن مجموعه داده ارائه شد در [16] روش پیشنهادی با در نظر گرفتن تمام معیارها هم در زمینه نظرات مفید و هم در زمینه نظرات غیر مفید در مقایسه با RevHelper [16] بهتر عمل می‌کند. با در نظر گرفتن نظرات مفید، روش پیشنهادی در تمامی معیارها بین یک تا دو درصد بهتر عمل کرده است و با در نظر گرفتن نظرات غیر مفید، روش پیشنهادی در تمامی معیارها بین سه تا پنج درصد بهبود داشته است. علاوه بر این با در نظر گرفتن معیار دقت حدود دو درصد بهبود داشته است.

با در نظر گرفتن مجموعه داده جمع‌آوری‌شده در این پژوهش، روش پیشنهادی با در نظر گرفتن تمامی معیارها نظرات مفید را بین یک تا سه درصد بهتر پیش‌بینی می‌کند و علاوه بر این بر اساس معیار دقت حدود نیم درصد عملکرد بهتری نسبت به روش ارائه‌شده در [16] دارد. در زمینه پیش-بینی نظرات غیرمفید تنها بر اساس معیار بازخوانی ضعیف‌تر عمل کرده است. در این مجموعه داده چون روش پیشنهادی در زمینه پیش‌بینی نظرات مفید که کمتر هستند و با در نظر گرفتن تمام معیارها بهتر عمل کرده است و تنها بر اساس معیار بازخوانی در پیش‌بینی نظرات غیر مفید ضعیف‌تر عمل کرده است، می‌توان گفت روش ارائه شد بهتر عمل کرده است.

در مجموع با توجه به نتایج حاصل از پیاده‌سازی مدل روی دو مجموعه داده کاملاً مجزا روش پیشنهادی در پیش‌بینی نوع نظرات با توجه به معیارهای صحت، فراخوانی و امتیاز اف–وان حدود سه درصد و با توجه به معیار دقت حدود یک درصد بهبود داشته است.

جدول (٦): نتایج روی مجموعه داده مقاله [16]

نوع نظرات	معیار	پیش‌بینی‌کننده	
		RevHelper (جنگل تصادفی)	XGBoost
نظرات مفید	precision	۷۳/۰۸	۷۵/۰۸
	recall	۶۶/۸۳	۶۷/۶۹
	F1-score	۶۹/۸۲	۷۱/۱۹
نظرات غیرمفید	precision	۳۰/۲	۳۳/۷
	recall	۳۷	۴۲/۵
	F1-score	۳۳/۲	۳۷/۷
تمام نظرات	accuracy	۵۸/۴۵	۶۰/۶۱

جدول(۷): نتایج روی کل مجموعه داده جمع‌آوری‌شده

نوع نظرات	معیار	پیش‌بینی‌کننده	
		RevHelper (جنگل تصادفی)	XGBoost
نظرات مفید	precision	۴۵/۱۶	۴۶/۸۲
	recall	۱۳/۲	۱۵/۹۳
	F1-score	۲۰/۴۳	۲۳/۷۶
نظرات غیرمفید	precision	۶۲	۶۲
	recall	۸۹/۸	۸۸/۳
	F1-score	۷۳	۷۳
تمام نظرات	accuracy	۵۹/۸۶	۶۰/۱۱

مراجع

[1] *"ansible"* https://github.com/ansible/ansible (accessed February-2019).

[2] J. Cohen, E. Brown, B. DuRette, and S. Teleki, *"Best kept secrets of peer code review"*. Smart Bear Somerville, 2006.

[3] A. Bosu, M .Greiler, and C. Bird, *"Characteristics of useful code reviews: An empirical study at microsoft"*, in 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, 2015: IEEE, pp. 146-156.

[4] O. Kononenko, O. Baysal, and M. W. Godfrey, *"Code review quality: how developers see it"*, in 201 ۶IEEE/ACM 38th International Conference on Software Engineering (ICSE), 2016: IEEE, pp. 1028-1038 .

[5] L. MacLeod, M. Greiler, M.-A. Storey, C. Bird, and J . Czerwonka, *"Code reviewing in the trenches: Challenges and best practices"*, IEEE Software, vol. 35, no. 4, pp. 34-42, 2017.

[6] Q. Le and T. Mikolov, *"Distributed representations of sentences and documents"*, in International conference on machine learning, 2014, pp. 1188-1196.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *"Distributed representations of words and phrases and their compositionality"*, in Advances in neural information processing systems, 2013, pp. 3111-3119 .

[8] "*django"* https://github.com/django/django (accessed February-2019).

[9] *"flask"* https://github.com/pallets/flask (accessed February-2019).

[10] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M .W. Godfrey, *"Investigating code review quality: Do people and participation matter?"*, in 2015 IEEE international conference on software maintenance and evolution (ICSME), 2015: IEEE, pp. 111-120 .

[11] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, *"Investigating technical and non-technical factors influencing modern code review"*, Empirical Software Engineering, vol. 21, no. 3, pp. 932-959, 2016.

[12] *"keras"* https://github.com/keras-team/keras (accessed February-2019).

[13] X. Yang, R. G. Kula, N. Yoshida, and H. Iida, *"Mining the modern code review repositories: A dataset of people, process and product"*, in Proceedings of the 13th International Conference on Mining Software Repositories, 2016: ACM, pp. 460-463 .

[14] M. Beller, A. Bacchelli, A. Zaidman, and E. Juergens, *"Modern code reviews in open-source projects: Which problems do they fix?"*, in Proceedings of the 11th working conference on mining software repositories, 2014: ACM, pp. 202-211 .

[15] *"pandas"* https://github.com/pandas-dev/pandas (accessed February-2019).

[16] M. M. Rahman, C. K. Roy, and R. G. Kula, *"Predicting usefulness of code review comments using textual features and developer experience"*, in 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017: IEEE, pp. 215-226 .

[17] M. M. Lehman, *"Programs, life cycles, and laws of software evolution"*, Proceedings of the IEEE, vol. 68, no. 9, pp. 1060-1076, 1980.

[18] *"pytorch"* https://github.com/pytorch/pytorch (accessed February-2019).

[19] Y. Huang, N. Jia, X. Chen, K. Hong, and Z. Zheng, *"Salient-class location: Help developers understand code change in code review"*, in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018: ACM, pp. 770-774 .

[20] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, *"SentiCR: a customized sentiment analysis tool for code review interactions"*, in Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, 2017: IEEE Press, pp. 106-111 .

[21] *"sentry"* https://github.com/getsentry/sentry (accessed February-2019).

[22] *"tensorflow"* https://github.com/tensorflow/tensorflow (accessed February-2019).

[23] S. Ruangwan, P. Thongtanunam, A. Ihara, and K. Matsumoto, *"The impact of human factors on the participation decision of reviewers in modern code review"*, Empirical Software Engineering, vol. 24, no. 2, pp. 973-1016, 2019.

[24] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, *"The influence of non-technical factors on code review"*, in 2013 20th Working Conference on Reverse Engineering (WCRE), 2013: IEEE, pp. 122-131 .

[25] *"tornado"* https://github.com/tornadoweb/tornado (accessed February-2019).

[26] T. Chen and C. Guestrin, *"Xgboost: A scalable tree boosting system"*, in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016: ACM, pp. 785-794

[27] A. Ram, A. A. Sawant, M. Castelluccio, and A. Bacchelli, *"What makes a code change easier to review: an empirical investigation on code change reviewability"*, in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018: ACM, pp. 201-212 .

[28] *"youtube-d"* https://github.com/ytdl-org/youtube-dl (accessed February-2019).

[29] *"zulip"* https://github.com/zulip/zulip (accessed February-2019).

پانویس ها

[1] Peer Code Review
[2] Data Set
[3] Precision
[4] Recall
[5] F1 Score
[6] Accuracy
[7] Bugs
[8] Technical
[9] Commit
[10] Salient
[11] Reviewed Twice
[12] Total Authored Commit
[13] Reviewed Pull Request
[14] Total Reviewed Commits
[15] Reviewed Commits Files
[16] Authored Commits Files
[17] Numerical
[18] Categorical
[19] Overfit
[20] Fully Connected
[21] Gated Recurrent Unit
[22] LSTM
[23] Weight Regularizers