دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

استفاده از داده‌های تجربی برای پشتیبانی از انتخاب فناوری در تصمیم‌گیری معماری نرم‌افزار

پایان‌نامه کارشناسی ارشد فناوری اطلاعات
گرایش معماری سازمانی

نگارش

سیدمحمدمسعود صدرنژاد

استاد راهنما

دکتر صادق علی‌اکبری

پاییز ۱۴۰۱

دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

پایان‌نامه کارشناسی ارشد فناوری اطلاعات - گرایش معماری سازمانی

تحت عنوان:

استفاده از داده‌های تجربی برای پشتیبانی از انتخاب فناوری در تصمیم‌گیری معماری نرم‌افزار

در تاریخ ۱۴۰۱/۰۷/۲۷ پایان‌نامه دانشجو، سیدمحمدمسعود صدرنژاد، توسط کمیته تخصصی داوران مورد بررسی و تصویب نهایی قرار گرفت.

۱- استاد راهنما اول:	دکتر صادق علی‌اکبری	امضا
۲- استاد داور (داخلی):	دکتر حسن حقیقی	امضا
۳- استاد داور (خارجی):	دکتر حسین رحمانی	امضا
۴- نماینده تحصیلات تکمیلی:	دکتر فریدون شمس علیئی	امضا

با سپاس و قدردانی از

استاد گرامی، جناب آقای دکتر صادق علی‌اکبری که به ثمر رسیدن این پایان‌نامه، بدون زحمات پیوسته و راهنمایی‌های خیرخواهانه و بی‌دریغشان محال بود.

برادر عزیزم مهدی صدرنژاد، به خاطر مشورت‌ها و منابع ابری که برای انجام پردازش‌های مورد نیاز در این پایان‌نامه به من اهدا کرد.

به نام خدا

نام و نام خانوادگی: سیدمحمدمسعود صدرنژاد

عنوان پایان‌نامه: استفاده از داده‌های تجربی برای پشتیبانی از انتخاب فناوری در تصمیم‌گیری معماری نرم‌افزار

استاد راهنما: دکتر صادق علی‌اکبری

اینجانب سیدمحمدمسعود صدرنژاد تهیه‌کننده پایان‌نامه کارشناسی ارشد حاضر، خود را ملزم به حفظ امانت‌داری و قدردانی از زحمات سایر محققین و نویسندگان بنابر قانون Copyright می‌دانم. بدین وسیله اعلام می‌نمایم که مسئولیت کلیه مطالب درج شده با اینجانب می‌باشد و در صورت استفاده از اشکال، جداول و مطالب سایر منابع، بلافاصله مرجع آن ذکر شده و سایر مطالب از کار تحقیقاتی اینجانب استخراج گشته‌است و امانت‌داری را به صورت کامل رعایت نموده‌ام. در صورتی که خلاف این مطلب ثابت شود، مسئولیت کلیه عواقب قانونی با شخص اینجانب می‌باشد.

نام و نام خانوادگی: سیدمحمدمسعود صدرنژاد

تاریخ و امضا:

تقدیم به

دلسوزترین و فداکارترین همراهان زندگیم؛

پدرم که من را همواره درس متانت و بزرگواری آموخت و مادرم که مهرش در قلبم جاودان است.

چکیده

استفاده از خرد جمعی[1] معماران نرم‌افزار از طریق کاویدن مخازن پروژه‌های نرم‌افزار[2] های معتبر آزاد/متن‌باز[3]، یکی از حوزه‌های پژوهشی است که در سال‌های اخیر با استقبال بیشتری روبرو شده‌است. با توجه به رویش روزافزون فناوری‌های در دسترس، انتخاب فناوری‌های مناسب از میان آن‌ها به یکی از چالش‌های پیش‌روی معماران نرم‌افزار تبدیل شده‌است. سامانهٔ توصیه‌گر راهکاری مناسب برای پشتیبانی از معمار در انتخاب فناوری است. پژوهش‌های صورت‌گرفته در این حوزه از طیف وسیعی از روش‌های یادگیری ماشین برای ساخت این سامانه‌ها استفاده کردند ولی تأثیر کیفیت و اندازهٔ پروژه‌های استفاده شده مورد مطالعه قرار نگرفته‌است. همچنین دقت پیشنهادهایشان با توجه به پیشرفت‌های اخیر در الگوریتم‌های یادگیری عمیق چندان مناسب نیست و به مشکل شروع سرد مبتلا هستند. این پژوهش پس از استخراج دادهٔ کیفیت پروژه‌ها، با توسعهٔ دو سامانهٔ توصیه‌گر مبتنی بر یادگیری عمیق به نام DeepLibAide و مبتنی بر محتوا به نام ContentLibAide و توصیه‌گر تلفیقی با هدف حل مشکل شروع سرد، به مقایسهٔ نتایج اجرای آن‌ها روی نمونه‌های مختلف از کیفیت پروژه‌ها می‌پردازد. مقایسهٔ نتایج DeepLibAide نشان می‌دهد که به واسطهٔ استفاده از تابع فعال‌ساز خطی، حذف تبدیل ویژگی‌ها و جمع زدن بردارهای جاساز لایه‌ها، نتایج ارزیابی مدل طراحی شده در فراسنج[4] های مختلف، نسبت به روش پایه در معیارهای دقت، بازیابی و سود انباشتهٔ کاهش‌یافتهٔ نرمال‌شده و ضرر آزمون بهبود یافته‌است. دقت و بازیابی آموزش مدل توسط دادگان بزرگ‌تر، دادگان حاوی فقط پروژه‌های باکیفیت‌تر و انتخاب تصادفی پروژه‌ها مقایسه شده‌است. DeepLibAide نسبت به روش پایه به‌طور میانگین هفت درصد در معیار بازیابی بهبود یافته. معماری خط‌لولهٔ طراحی‌شده از ابتدای استخراج یا به‌روزرسانی دادگان ورودی و ساخت نمونه‌ها تا مرحلهٔ آموزش مدل‌ها با فراسنج‌های مختلف و ذخیره‌سازی پیشنهادها و ارزیابی و مقایسهٔ آن‌ها با روش پایه در نمودارهای مختلف طوری طراحی شده‌است که کل برنامه به‌طور خودکار و با حداقل نیاز به دستکاری اجرا شود.

واژگان کلیدی: انتخاب فناوری، سامانهٔ توصیه‌گر[5]، یادگیری عمیق، کتابخانهٔ نرم‌افزاری[6]، استخراج دادگان

---

[1] wisdom of the crowd
[2] mining software repositories
[3] free/open source software
[4] parameter
[5] recommender system
[6] software library

فهرست مطالب

فهرست شکل‌ها

فهرست جداول

فصل ۱

مقدمه

طراحان نرم‌افزار، ضمن فرایند طراحی معماری، شماری تصمیم‌های اساسی می‌گیرند که تأثیرات ژرفی روی نرم‌افزار و فرایند ایجاد آن می‌گذارد به همین دلیل تفکر دربارهٔ فرایند طراحی معماری از دیدگاه تصمیم‌گیری و نه از دیدگاه یک فعالیت مفید است. اتخاذ تصمیم‌های معماری مناسب، برای موفقیت پروژه‌های نرم‌افزاری حیاتی است؛ بنابراین معماری نرم‌افزار نیازمند یک فرایند مطمئن و دقیق برای انتخاب از میان گزینه‌های معماری است و باید تضمین کند که تصمیم‌ها باعث کمینه کردن مخاطرات و بیشینه کردن کیفیت می‌شوند. این امر منجربه نیاز به مطالعهٔ روش‌های تصمیم‌گیری در حوزهٔ معماری نرم‌افزار شده‌است.

پژوهشگران این حوزه تأکید می‌کنند که تصمیم‌گیری دربارهٔ معماری نرم‌افزار، فرایندی پیچیده‌است و معمولاً چگونگی تصمیم‌گیری معمار واضح نیست. گاهی حتی خود فرد تصمیم‌گیر هم نمی‌تواند مشخص کند که چگونه تصمیم به‌خصوصی را گرفته‌است. این پژوهش در قالب یک سامانهٔ توصیه‌گر[1] با پیشنهاد فناوری، در اتخاذ تصمیم‌های معماری نرم‌افزار در حوزهٔ انتخاب فناوری به معمار نرم‌افزار یاری می‌رساند.

۱.۱ تعریف مسئله

تصمیم‌گیری توسط معمار، نیازمند دانش قبلی دربارهٔ زمینه، هدف تصمیم‌گیری یا معیارهای ارزیابی، انتخاب‌های بدیل[2] و وضعیت هر گزینه در ارتباط با هر یک از معیارها است و خروجی این فرایند، خود تصمیم یا فهرستی مرتب‌شده از انتخاب‌های ممکن است. طراحی روش برای یاری رساندن به معمار نرم‌افزار با در نظر گرفتن این مسیر، نیازمند تأمین دانش مورد نیاز دربارهٔ هدف، انتخاب‌های بدیل و وضعیت هر انتخاب در قبال هر معیار است. در بیشتر مواقع استفاده از روش‌های استنتاج براساس دانش خبره در زمینهٔ تصمیم‌گیری معماری نرم‌افزار به‌تنهایی کافی نیست زیرا قسمت دشوار فعالیت معمار اخذ دانش مورد نیاز دربارهٔ ورودی‌های تصمیم (هدف، انتخاب‌های بدیل و وضعیت هر انتخاب در قبال هر معیار) و ثبت آن است یا تفاوت خروجی تصمیم‌گیری یک معمار خبره با یک معمار تازه‌کار در میزان درستی دانش آن‌ها دربارهٔ همین سه مورد است. دانش سطح بالای مورد نیاز برای استفاده از روش‌های استنتاج براساس دانش خبره، دانش ضمنی[3] معماران خبره است که مخصوصاً با محبوبیت بیشتر روشگان چابک، معمولاً مستند نشده و به‌طور صریح بیان نمی‌شود.

[1] recommender system
[2] alternatives
[3] implicit

این درحالی است که خود تصمیم‌های معماری در سوابق یک پروژهٔ نرم‌افزاری موجود است؛ مثلاً با بررسی کدمنبع یک پروژه می‌توان دریافت که از چه فناوری‌هایی برای ارتباط با پایگاه‌داده یا برای آزمون واحد از چه فناوری‌هایی استفاده شده. در واقع دانش سطح پایین دربارهٔ معماری که خود تصمیم‌های اتخاذشده است با بررسی کدمنبع قابل استخراج است ولی دانش سطح بالای استفاده‌شده مانند هدف یا معیارها و انتخاب‌های بدیل[۱] در هر تصمیم‌گیری و پایه‌واساس[۲] اتخاذ یک تصمیم مشخص، ثبت نمی‌شوند.

۲.۱ اهمیت و ضرورت موضوع

عملیات معماری کمّی‌سازی شده نیست و معماری نرم‌افزار دانشی است که اغلب نزد معماران نرم‌افزار است یا اگر کسی بخواهد معمار نرم‌افزار شود باید به معماران نرم‌افزار پیشین مراجعه کند. برای همین استفاده از کمّی‌سازی[۳]، اندازه‌گیری[۴] و به‌طور کلی روش‌های مهندسی برای تمام بخش‌های عملیات معماری یک زمینهٔ پژوهشی شایستهٔ توجه است. کمّی‌سازی و قابل محاسبه و اندازه‌گیری کردن معماری نرم‌افزار به تعبیری معماری نرم‌افزار را از یک پیشه[۵] به سمت استفاده از روش‌های مهندسی، محاسباتی و تکرارپذیر هدایت می‌کند.

با مطالعهٔ پژوهش‌های پیشین و بررسی خلأهای موجود در حوزهٔ تصمیم‌گیری معماری نرم‌افزار، مشخص می‌شود که اهمیت انتخاب فناوری در این است که (الف) فناوری‌ها در قیاس با سایر تصمیمات معماری مثل سبک‌ها یا الگوهای معماری با سرعت بیشتری در حال رشد و افزایش هستند. با رشد روزافزون فناوری‌ها، انتخاب‌های معماران نیز روبه‌افزایش است بنابراین انتظار می‌رود، بتوان مدلی که تصمیم‌گیری برمبنای آن انجام می‌شود را با فناوری‌های جدید به‌طور خودکار کامل کرد و از سوی دیگر روش پیشنهادی برای تصمیم‌گیری باید از مقیاس‌پذیری بالایی برخوردار باشد [۱].

علاوه بر این (ب) خصوصیات فناوری‌ها هم، در طی زمان در حال ظهور و تغییر است و مدلی که ارائه می‌شود بهتر است وابسته به خصوصیات کنونی فناوری‌ها نباشد. چون هم معیارها و هم فناوری‌ها عوض می‌شوند؛ و مدلی برای تصمیم‌گیری مطلوب است که مستقل از این دو عمل کند [۱]. برای همین ارائهٔ یک مدل که بتواند به‌طور خودکار با فناوری‌های جدید تکمیل شود و وابسته به جزئیات خصوصیات هر فناوری نباشد امری ضروری است.

[1] alternatives
[2] rationale
[3] quantification
[4] measurement
[5] craftmanship

برای این منظور استفاده از روش‌های خودکار برای استخراج فناوری‌ها و وابسته کردن تصمیم‌گیری به نحوهٔ استفاده از آن‌ها به جای ویژگی‌های خود فناوری‌ها، باعث افزایش انعطاف دربرابر رشد و تغییرات فناوری‌ها در طی زمان شود.

از سوی دیگر (ج) هرچند که پژوهش‌های اندکی روی تصمیم‌گیری[1] دربارهٔ انتخاب فناوری توسط معماران انجام شده‌است ولی موضوع انتخاب فناوری، بیشتر در پژوهش‌های اخیر مورد توجه قرار گرفته‌است. همچنین از آن‌جایی که انتخاب‌های فناوری در خود پروژه‌ها موجود است می‌توان با روش‌های کاملاً تجربی دستاوردها را ارزیابی کرد و الزاماً نیازی به روش‌هایی مانند پرسشنامه و مطالعهٔ میدانی و مصاحبه و ... برای ارزیابی یافته‌ها نیست.

۳.۱ اهداف پایان‌نامه

به‌طور نظری هر سامانهٔ نرم‌افزاری، دارای معماری است حتی اگر آن معماری مستند نشده باشد یا طراحان آن در دسترس نباشند اما در عمل استخراج همهٔ دانش معماری درخصوص پایه‌واساس تصمیم‌ها با فقط دراختیار داشتن خود تصمیم‌های انتخاب فناوری ممکن نیست. با این حال با در اختیار داشتن انتخاب‌های فناوری، که از پروژه‌ها قابل استخراج هستند، می‌توان از معمار برای تکمیل و بهبود آن‌ها پشتیبانی کرد.

یکی از ملاک‌های انتخاب فناوری، سازگاری فناوری با پشتهٔ فناوری موجود است. برای این منظور می‌توان فناوری‌هایی که در پروژه‌های معتبر با هم به کار می‌روند را مطالعه کرد یا شبکهٔ وابستگی‌های آن‌ها و نسخه‌های به‌کار رفتهٔ آن‌ها را شناسایی و مورد مطالعه قرارداد. این روش همهٔ پایه‌واساس تصمیم‌ها را مشخص نمی‌کند ولی تا همین‌جا می‌شود پیشنهادهای خوبی به معمار برای استفاده از فناوری‌ها از روی مدل ساخته‌شده از روی پروژه‌های پیشین داد؛ مثلاً معمار می‌تواند مجموعه‌ای از انتخاب‌های فناوری فعلی را (فارغ از پایه و اساس انتخاب‌هایش) به سامانهٔ تصمیم‌یار ارائه دهد و سامانه مثلاً به معمار پیشنهاد کند که «در کنار این فناوری، فناوری‌های دیگری را هم استفاده کن چون پروژه‌های با کیفیت بالا در گیت‌هاب که از فناوری‌هایی مشابه شما استفاده کرده‌اند، آن‌ها را نیز مورد استفاده قرار دادند» یا می‌توان بررسی کرد که مثلاً در پروژه‌های با تعداد ستارهٔ بالا در کنار یک چارچوب وب مانند اسپرینگ[2] چه برنامه‌ای برای نگاشت رابطه به شی عمدتاً مورد استفاده قرار گرفته‌است. سپس معمار می‌تواند با استفاده از داده‌هایی که دریافت کرده انتخاب‌های فعلی فناوری را بهبود دهد. برای این منظور مفروضاتی برای

[1] decision-making
[2] Spring

کوچک‌تر کردن دامنهٔ مسئله در نظر گرفته شده‌است. این مدل از روی وابستگی‌های موجود به فناوری‌ها در پروژه‌های جاوای با قدمت و اعتبار کافی در گیت‌هاب ساخته شده‌است. در واقع، از این‌رو وسعت کار این پژوهش کوچک‌تر شده‌است که (الف) فناوری مورد انتخاب به کتابخانه‌های سطح بالا و (ب) دادهٔ آموزشی به کدهای منتشر شده از پروژه‌های جاوا و کتابخانه‌های مخزن مِیوِن محدود شده‌است.

یکی از نقاط ضعف روش بیان‌شده برای پیشنهاد فناوری به معمار بر اساس دادهٔ فناوری‌های به‌کار رفته در پروژه‌های پیشین، شروع سرد است. مسئلهٔ شروع سرد در این حوزه به این شکل صورت‌بندی می‌شود که اگر پروژه‌ای که می‌خواهیم برای آن کتابخانه‌هایی را پیشنهاد کنیم در ابتدای فرایند ایجادش باشد یا در مراحلی باشد که هنوز به بلوغ کافی نرسیده، دادهٔ کافی دربارهٔ فناوری‌های مورد استفاده در آن پروژه وجود ندارد و بدون داشتن دادهٔ کافی در مورد یک پروژه، نمی‌توان فناوری‌هایی برای استفاده در آن پیشنهاد کرد. برای حل این مسئله از منابع دادهٔ ثانویه برای بررسی شباهت محتوای پروژهٔ جدید با پروژه‌های پیشین استفاده شده‌است و به این ترتیب حتی درصورت نداشتن دادهٔ کافی دربارهٔ انتخاب‌های فناوری موجود در یک پروژه، می‌توان پیشنهادهایی به معمار ارائه داد.

سوال‌های اصلی

۱. با مطالعهٔ فناوری‌های استفاده‌شده در معماری پروژه‌های پیشین، چگونه می‌توان انتخاب‌های فناوری انجام‌گرفته در پروژه‌های آینده را بهبود داد؟

۲. با توجه به این‌که بسیاری از پروژه‌ها در ابتدای مسیر ایجادشان هستند و هنوز به بلوغ زیادی نرسیدند و دچار مشکل شروع سرد[1] هستند، یا به عبارت دیگر هنوز دادهٔ کافی دربارهٔ انتخاب‌های فناوری موجود در آن‌ها وجود ندارد، چگونه بدون وجود دادهٔ کافی دربارهٔ انتخاب‌های پیشین در یک پروژه، می‌توان پیشنهادهای شخصی‌سازی‌شده برای چنین پروژه‌هایی ارائه داد؟ چگونه می‌توان داده‌های جانبی دربارهٔ پروژه‌ها استخراج نمود و از این داده برای ساخت یک توصیه‌گر تلفیقی، استفاده کرد؟

۳. کیفیت و تعداد پروژه‌های مورد مطالعه برای پیشنهاد فناوری چه تأثیری روی کیفیت پیشنهادهای تولیدشده توسط مدل دارد؟

فرضیه‌ها

[1] cold start

۱. انتخاب‌های فناوری در پروژه‌های نرم‌افزاری به عنوان یک نمونه از تصمیم‌های معماری با بررسی داده‌های تجربی حاصل از کدمنبع قابل شناسایی و استخراج است.

۲. اگر معماران پروژه‌ها در مورد برخی از انتخاب‌های فناوری‌هایشان با هم توافق داشته باشند، می‌توان پیش‌بینی کرد که در مورد بقیهٔ فناوری‌ها هم باید توافق داشته باشند.

۳. با داشتن موارد استفاده از فناوری‌ها در پروژه‌های گذشته، می‌توان به معمار گزارش‌های سودمندی برای تصمیم‌گیری دربارهٔ انتخاب‌های جدید فناوری برای ایجاد یا تکامل نرم‌افزار ارائه کرد.

۴. استفاده از یک مدل یادگیری عمیق سبک می‌تواند باعث افزایش قابل توجه کارایی مدل در ارائهٔ پیشنهادها شود و می‌توان از اطلاعات جانبی برای حل مسئلهٔ شروع سرد در یک سامانهٔ توصیه‌گر تلفیقی استفاده کرد.

در این پژوهش با استفاده از یک مدل گراف پیچشی سبک و یک توصیه‌گر مبتنی بر محتوا تلاش می‌شود تا از داده‌های تجربی مرتبط با تصمیم‌های اتخاذشدهٔ پیشین در تصمیم‌گیری معماری نرم‌افزار استفاده شود. این رویکرد توجه ویژه‌ای به اهمیت نقش عامل انسانی و دانش معمار به عنوان پیشه‌ور در تصمیم‌گیری نهایی در مورد انتخاب‌های فناوری دارد.

۴.۱ ساختار پایان‌نامه

این پایان‌نامه شامل شش فصل است. فصل دوم دربرگیرندهٔ ادبیات موضوع و تعاریف اولیهٔ مرتبط با مسئلهٔ پایان‌نامه و روش‌های مورد استفاده در این حوزه است. در فصل سوم کارهای مرتبطی که در زمینهٔ انتخاب فناوری انجام شده به تفصیل بیان می‌گردد. در فصل چهارم روش پیشنهادی این پژوهش برای حل مسئلهٔ فوق مورد بررسی قرار گرفته‌است. در فصل پنجم نتایج ارزیابی و بهبودهایی که در این پایان‌نامه به دست آمده ارائه می‌گردد. در این فصل، فراسنج‌های مؤثر در اجرای الگوریتم مورد بررسی قرار می‌گیرند. سپس نگاهی به معیارهای ارزیابی به‌کار رفته در پژوهش‌های پایه می‌شود. در انتها با در نظر گرفتن معیارها و فراسنج‌های بیان‌شده، به مقایسهٔ روش پیشنهادی با پژوهش پایه پرداخته شده‌است. فصل ششم به نتیجه‌گیری و پیش‌نهادهایی برای کارهای آتی خواهد پرداخت.

فصل ۲

ادبیات و مفاهیم اولیه

این فصل دربرگیرندهٔ ادبیات موضوع و تعاریف اولیهٔ مرتبط با این پایان‌نامه است.

۱.۲ معماری نرم‌افزار

طراحی نرم‌افزار هم به معنای «فرایند تعریف معماری، مولفه‌ها، واسط‌ها و سایر مشخصات یک سامانه یا مولفه» و هم «نتیجهٔ فرایند مذکور» است. اگر به طراحی به عنوان یک فرایند نگاه شود، طراحی یکی از فعالیت‌هایی خواهد بود که در چرخهٔ عمر ایجاد نرم‌افزار[1] مورد توجه قرار می‌گیرد. طراحی به عنوان یک فعالیت به معنای تحلیل نیازمندی‌ها برای تولید توصیفی از ساختار داخلی برنامه است که اساس ساخت نرم‌افزار[2] است.

تعریف ۱.۲ (معماری نرم‌افزار). *معماری نرم‌افزار مجموعه ساختارهای مورد نیاز برای استدلال در مورد سامانه، شامل عناصر نرم‌افزاری، روابط بین آن‌ها و خصوصیات هر دو است و در فازهای اولیهٔ فرایند ایجاد نرم‌افزار با هدف رفع نیازهای خاص معماری[3] و تسهیل اهداف حرفه طراحی می‌شود.*

در نسخهٔ سوم کتاب پیکرهٔ دانش مهندسی نرم‌افزار[4]، معماری نرم‌افزار بخشی از فرایند طراحی نرم‌افزار و طراحی نرم‌افزار بخشی از چرخهٔ عمر نرم‌افزار است. جایگاه معماری نرم‌افزار در درخت موضوعی مهندسی نرم‌افزار در شکل ۱.۲ نمایش داده شده‌است. هدف از طراحی نرم‌افزار پر کردن خلأ میان نیازمندی‌ها تا ساخت نرم‌افزار است [۲]. گره‌هایی از این درخت که به مباحث مطرح شده در این پژوهش مرتبط هستند با خط توپُر مشخص شده‌اند.

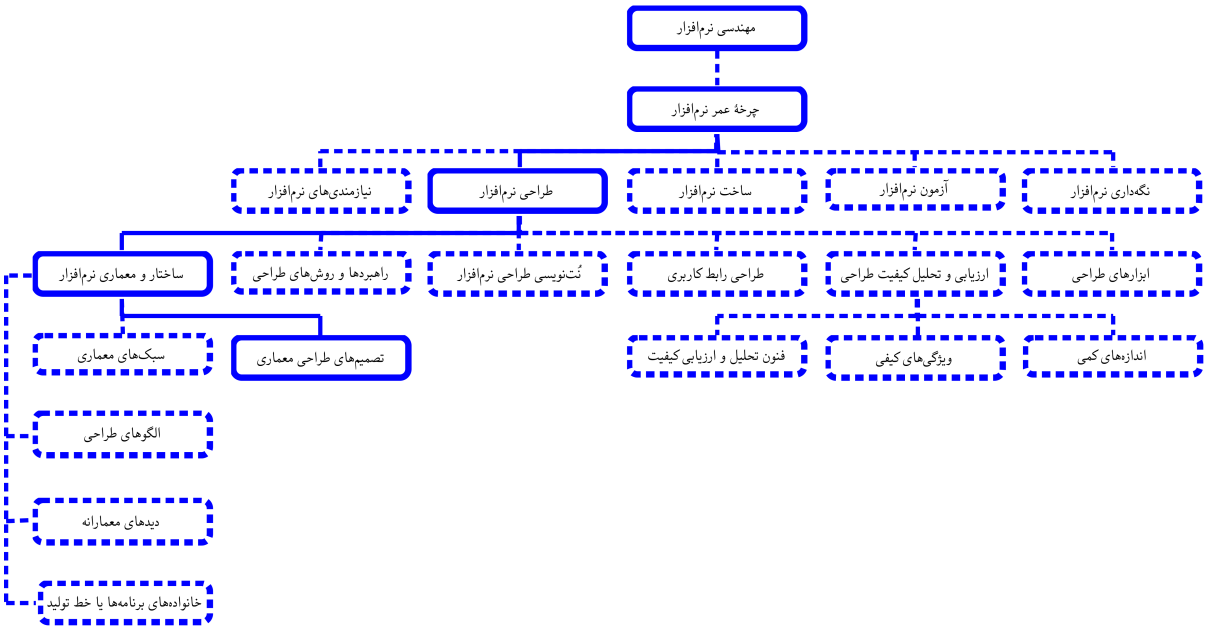
مطابق شکل ۱.۲، ساختار و معماری نرم‌افزار[5]، ذیل عنوان طراحی نرم‌افزار[6] مورد بررسی قرار می‌گیرد و ذیل موضوع ساختار و معماری نرم‌افزار به مفاهیم ساختارها و دیدگاه‌های معماری[7]، سبک‌های معماری، الگوهای طراحی، تصمیم‌های طراحی معماری، خانواده‌های برنامه‌ها و چارچوب‌ها[8] پرداخته شده‌است [۲].

۱.۱.۲ انواع تصمیم‌های معماری نرم‌افزار

همان‌طور که در بخش قبل بیان شد، از بین مباحث مطرح ذیل عنوان ساختار و معماری نرم‌افزار، مفهوم تصمیم‌های طراحی معماری بیش از همه به هدف این پژوهش نزدیک است.

---

[1] software development life cycle
[2] software construction
[3] architecturally significant requirements
[4] Software Engineering Body of Knowledge (SWEBOK)
[5] software structure and architecture
[6] software design
[7] architectural viewpoints
[8] families of programs and frameworks



شکل ۱.۲: جانمایی معماری نرم‌افزار در درخت موضوعی مهندسی نرم‌افزار [۲]

تعریف ۲.۲ (تصمیمات معماری نرم‌افزار). *مجموعه‌ای از تصمیم‌های طراحی تأثیرگذار روی کیفیت کلی عملکرد سامانه.*

بر اساس طبقه‌بندی مطرح شده در [۳] تصمیم‌های معماری را می‌توان به تصمیم‌های معماری مفهومی و فناوری طبقه‌بندی کرد. تصمیم‌های مفهومی مانند انتخاب الگوها یا راهکنش‌های معماری، مستقل از پیاده‌سازی، روی پیکربندی معماری تأثیر می‌گذارند ولی هدف از تصمیم‌های فناوری، انتخاب چارچوب‌ها، زبان‌های برنامه‌نویسی، پایگاه‌های داده و ... است که برای پیاده‌سازی معماری مورد نظر باید مورد استفاده قرار گیرند و تصمیم‌های مفهومی را محقق می‌کنند. در انتخاب بین فناوری‌ها باید ویژگی‌های فناوری‌ها و نیازمندی‌های خاص معماری مورد توجه قرار گیرد. بر اساس یک مطالعهٔ انجام‌شده نزدیک به ۲۵ درصد از تصمیمات معماری در پروژه‌ها مرتبط با انتخاب‌های فناوری بودند [۳].

۲.۱.۲ اجزای تصمیم معماری نرم‌افزار

در مدل‌های پیشنهادشده برای مدیریت دانش مرتبط با تصمیم‌های معماری، هر تصمیم حداقل از پنج جزء تشکیل شده‌است که در ادامه مثالی از هر کدام برای یک تصمیم انتخاب فناوری نوشته شده‌است:

۱. **زمینهٔ تصمیم:** انتخاب فناوری

۲. **هدف تصمیم‌گیری:** انطباق فناوری برای نگاشت رابطه به شی[1] که با پشتهٔ فناوری کنونی شامل چارچوب وب اسپرینگ و ... سازگاری بالایی داشته باشد. استفاده از نگاشت رابطه به شی یک تصمیم معماری مرتبط با «مدل داده» است که با یک «تصمیم انتخاب فناوری» محقق می‌شود. درواقع این‌که کدام تصمیم‌ها از بین شش تصمیم اصلی دیگر معمار قرار است با انتخاب فناوری محقق شود در هدف تصمیم‌گیری جای می‌گیرد.

۳. **مجموعهٔ انتخاب‌های بدیل برای محقق شدن هدف تصمیم‌گیری:** فناوری‌های هایبرنت[2] و ...

۴. **وضعیت هر گزینه در قبال هر معیار:** فناوری هایبرنت پشتیبانی مناسبی از چارچوب وب اسپرینگ دارد.

۵. **خود تصمیم یا انتخاب:** از فناوری هایبرنت در پروژه استفاده شده‌است.

موارد یک تا چهار پایه‌واساس[3] تصمیم‌گیری و مورد پنج خروجی یا خود تصمیم است. برای نمونه برخی از موارد تأثیرگذار برای انتخاب از بین فناوری‌هایی که شش تصمیم دیگر معماری را محقق می‌کنند شامل موارد روبرو است: (الف) میزان پشتیبانی موجود برای هر فناوری مانند مستندات و ...، (ب) اثرات جانبی انتخاب یک فناوری مانند الزام به استفاده از یک مدل تعاملی خاص یا محدودیت‌های مرتبط با مصرف منابع و (ج) میزان سازگاری یک فناوری با پشتهٔ فناوری درحال استفاده. برای مثال این‌که «آیا این فناوری می‌تواند در بالای پشتهٔ فناوری کنونی اجرا شود؟ آیا می‌تواند با پشتهٔ فعلی ارتباط برقرار کند؟ تا چه‌اندازه امکان پایش و مدیریت این فناوری وجود دارد؟»

۲.۲ روش‌های پشتیبانی از تصمیم‌گیری

۱.۲.۲ روش‌های استنتاج براساس دانش خبره

ابزارهای مدیریت دانش، سامانه‌های خبره و سامانه‌های پشتیبان تصمیم مبتنی بر هستان‌شناسی، نمونه‌هایی از روش‌های مورد استفاده در روش‌های تصمیم‌گیری استنتاج براساس دانش خبره هستند. در این روش‌ها موضوع اصلی نحوهٔ رسیدن از دانش سطح بالا که توسط یک شخص خبره تأمین شده به دانش سطح پایین یا تصمیمات نهایی است؛

[1] Object Relation Mapping
[2] Hibernate
[3] rationale

مثلاً در روش‌های تصمیم‌گیری چند معیاره با یک تحلیل بده‌بستان می‌توان نقطهٔ بهینه برای تصمیم‌گیری را مشخص کرد. برای این منظور از یک ماتریس تصمیم استفاده می‌شود که نتایج تصمیم را برای مجموعه‌ای از گزینه‌ها و معیارهای ارزیابی بیان می‌کند. مسائل تصمیم‌گیری پیچیده معمولاً از تعدادی تصمیم‌گیرنده تشکیل می‌شود که به آن‌ها گروه‌های ذینفع می‌گویند. معیارهای مورد استفاده نیز باید درصورت کیفی بودن به کمی تبدیل شده، بی‌مقیاس شوند و سپس وزن نسبی معیارها مشخص شود. یکی از رایج‌ترین الگوریتم‌های تصمیم‌گیری چندمعیاره، فرایند تحلیل سلسله‌مراتبی است.

۲.۲.۲ سامانه‌های توصیه‌گر مبتنی بر الگوریتم‌های یادگیری سنتی

سامانه‌های توصیه‌گر در مواقعی استفاده می‌شوند که چندین قلم وجود دارد و از بین آن‌ها باید یک یا تعدادی را انتخاب کرد ولی برای انجام این انتخاب، نمی‌توان دربارهٔ تمام آن اقلام (به خاطر تعداد زیادشان) اطلاعات کافی به دست آورد. سامانهٔ توصیه‌گر در چنین شرایطی فهرستی مرتب شده از اقلام را به کاربر پیشنهاد کند یا می‌تواند نظر کاربر دربارهٔ هر قلم را پیش‌بینی کند. یک نوع سادهٔ سامانه‌های توصیه‌گر پیشنهادهای غیرشخصی[1] تولید می‌کنند، مثلاً هر قلمی که از محبوبیت بالایی برخوردار باشد را به همه پیشنهاد می‌کنند و به کاربری که به وی اقلامی را پیشنهاد می‌کنند توجه نمی‌کنند. یکی از اشکالات پیشنهادهای تولیدی توسط سامانه‌های غیرشخصی، بدیهی بودن آن‌ها است. دستهٔ دیگر پیشنهادهای نیمه‌شخصی ارائه می‌کنند و دستهٔ دیگر پیشنهادهای کاملاً شخصی برای کاربر تولید می‌کنند و این سامانه‌ها با این‌که از دقت بالاتری برخوردارند؛ ولی از آن‌جایی که برای ارائه پیشنهاد نیازمند داده‌هایی دربارهٔ کاربر جدید هستند، مشکل شروع سرد دارند.

۱.۲.۲.۲ یادگیری قانون انجمنی

یادگیری قانون انجمنی[2] مجموعه‌ای از قانون[3] استخراج و تولید می‌کند که هر قانون بیان‌گر اقلامی است که معمولاً با هم استفاده می‌شوند. از این پایگاه قوانین می‌توان برای انجام استنتاج‌های بعدی استفاده کرد. برای ارزیابی یک قانون در این روش از معیار اعتماد[4] استفاده می‌شود به این‌صورت که تعداد دفعات آمدن اقلام سمت چپ و راست یک قانون بر تعداد دفعات آمدن اقلام سمت چپ قانون تقسیم می‌شود. معیار دیگر برای ارزیابی این روش

[1] non-personalized recommendations
[2] association rule learning
[3] rule
[4] confidence

معیار لیفت[1] است که برای شناسایی قانون‌های مشتبه[2] مورد استفاده قرار می‌گیرد. اگر نسبت تعداد پروژه‌هایی که از قلم سمت راست یک قانون استفاده کردند بیشتر از تعداد پروژه‌هایی باشد که از اقلام داخل قانون با هم استفاده کردند لیفت کمتر یا مساوی یک است و اگر لیفت یک قانون زیر یک باشد قانون اشتباهی استخراج شده‌است.

۲.۲.۲.۲ پالایش همکارانه

پالایش همکارانه[3] روشی رایج برای ساخت سامانه‌های توصیه‌گر است. ورودی این سامانه‌ها، تعدادی کاربر، تعدادی قلم[4] و سابقهٔ بازخورد کاربران نسبت به اقلام است؛ مثلاً در مسئلهٔ مورد بررسی در این پژوهش، کاربران همان پروژه‌ها و اقلام هم کتابخانه‌ها هستند. بازخوردهای کاربرها نسبت به قلم‌ها به دو دستهٔ بازخورد صریح[5] و بازخورد ضمنی[6] تقسیم می‌شوند. در مواردی که داده به شکل بازخورد صریح ارائه می‌شود کاربر بازخورد را به‌طور واضح نسبت به آن قلم اعلام کرده‌است مثلاً دربارهٔ آن قلم نظری نوشته یا آن را پسندیده‌است. ولی در بیشتر مواقع، کاربر بازخورد خود را صراحتاً اعلام نکرده ولی از سابقهٔ فعالیت‌های کاربر می‌شود اطلاعاتی دربارهٔ بازخورد کاربر نسبت به اقلام به دست آورد؛ مثلاً استفادهٔ یک فناوری در یک پروژه یک بازخورد ضمنی است.

۳.۲.۲.۲ مبتنی بر محتوا

درسامانهٔ توصیه‌گر مبتنی‌بر محتوا[7] برخلاف سامانه‌های پالایش همکارانه که از دادهٔ بازخورد کاربران به اقلام برای محاسبهٔ شباهت استفاده می‌کرد، به محتوای قلم یا به محتوای کاربر توجه می‌شود. به عبارت دیگر اگر محتوای یک قلم برای یک کاربر مناسب تشخیص داده شود آن را می‌توان به کاربر پیشنهاد کرد. این محتوا در هر مسئله برحسب تعریف مسئله می‌تواند تعریف شود مثلاً می‌توان از محتوای کد یک کتابخانه یا برچسب‌های یک پروژه برای پیشنهاد کردن آن کتابخانه به آن پروژه استفاده کرد. یکی از روش‌های رایج مبتنی بر محتوا به این‌صورت عمل می‌کند که میزان شباهت محتوای تمام اقلام با هم را محاسبه می‌کند و به کاربر جدید اقلامی که بیشترین شباهت با اقلام مورد استفاده‌اش را دارند را پیشنهاد می‌کند. در نوع دیگری از روش‌های مبتنی برمحتوا می‌توان از محتوای پروژه‌ها و محاسبهٔ میزان شباهت آن‌ها با هم استفاده کرد.

---

[1] lift
[2] misleading
[3] collaborative filtering
[4] item
[5] explicit feedback
[6] implicit feedback
[7] content-based recommender system

۳.۲.۲ سامانه‌های توصیه‌گر مبتنی بر یادگیری عمیق

استفاده از یادگیری عمیق تأثیر قابل ملاحظه‌ای روی افزایش دقت سامانه‌های توصیه‌گر داشته‌است به این علت که امکانی را فراهم می‌کند تا علاوه بر بازخوردهای مستقیم کاربران به اقلام، در لایه‌های بالاتر از رابطهٔ غیرمستقیم میان آنان نیز استفاده شود. به عبارت دیگر استفاده از شبکه‌های عصبی گراف این امکان را فراهم می‌کند که هر دوی اطلاعات مرتبهٔ پایین و اطلاعات مرتبهٔ بالا[1] که از گراف‌ها قابل استخراج است، در جریان ارائهٔ پیشنهادها مورد استفاده قرار گیرد [۴].

۱.۳.۲.۲ گراف شبکه پیچشی

گراف شبکه پیچشی[2] الگوریتمی برای انتقال بردار ویژگی‌های ورودی[3] و هموار کردن تأثیر بردار ویژگی‌ها در یک گراف است. طرز کار این الگوریتم به این‌صورت است که در هر لایه از شبکهٔ عصبی، تمام گره‌های گراف بردار ویژگی همسایگانشان را دریافت می‌کنند، این بردارها را با استفاده از یک تابع تجمیع مانند میانگین، تجمیع می‌کنند و سپس آن گره با یک بردار جدید در گراف بازنمایی می‌شود و در لایهٔ بعدی شبکهٔ عصبی همین مراحل تکرار می‌شوند با این تفاوت که ورودی هر گره، بردارهای به‌روز شدهٔ همسایگانش از لایهٔ قبل است. درواقع تفاوت این الگوریتم با یک شبکهٔ عصبی ساده در این است که یک مرحلهٔ پیش‌پردازش قبل از اجرای هر لایه وجود دارد که در آن، هر گره بردارهای همسایگانش را با هم تجمیع می‌کند.

۲.۳.۲.۲ پالایش همکارانه گراف عصبی

الگوریتم پالایش همکارانه گراف عصبی[4] یک الگوریتم بر پایهٔ گراف شبکهٔ پیچشی است که هدف آن ساخت یک سامانهٔ توصیه‌گر پالایش همکارانه است [۴].

در این روش در گام اول به کاربر و قلم یک بردار جاساز از شناسه‌ها تخصیص داده می‌شود. سپس همان‌طور که در فرمول ۱.۲ مشاهده می‌شود، از تعاملات میان کاربران و اقلام استفاده می‌شود تا بردارهای جاساز در لایه‌های بالاتر در گراف منتشر شود. سپس برای به‌دست آوردن بردار جاساز آخرین لایه، بردارهای جاساز لایه‌های مختلف به هم می‌چسبد.

[1] high-order
[2] Graph Convolutional Network
[3] input feature vector
[4] Neural Graph Collaborative Filtering

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2\left(\mathbf{e}_i^{(k)}\odot\mathbf{e}_u^{(k)}\right)\right)\right) \quad (۱.۲)$$

در رابطهٔ ۱.۲ مقدار $e_u^{(k+1)}$ بردار جاساز یک کاربر u در لایهٔ $k+1$ و $e_u^{(k)}$ بردار جاساز همان کاربر در لایهٔ قبل است. درواقع این رابطه نشان می‌دهد بردار جاساز یک کاربر در هر لایه را چگونه می‌توان از بردار جاساز آن در لایهٔ قبل به دست آورد. σ یک تابع فعال‌ساز غیرخطی مانند سیگموید[1] و همچنین $\mathbf{W}_1$ و $\mathbf{W}_1$ ماتریس‌های وزن تبدیل ویژگی هستند. همچنین $\mathcal{N}_u$ و $\mathcal{N}_i$ به ترتیب برابر مجموعهٔ اقلامی که با کاربر u تعامل داشتند و مجموعهٔ کاربرانی که با قلم i تعامل داشتند است. رابطهٔ ۱.۲ عیناً برای رسیدن به بردار جاساز تمام اقلام در لایه‌های بالاتر هم قابل استفاده است به این شکل که فقط جای u و i در رابطهٔ بالا باید با هم تغییر کند.

۳.۳.۲.۲ شبکهٔ پیچشی گراف سبک

الگوریتم پالایش همکارانه گراف عصبی کاملاً از گراف شبکه پیچشی اقتباس شده‌است و برای همین تمام جزئیات آن را بدون تغییر چندانی استفاده کرده‌است درحالی‌که بسیاری از این جزئیات برای مسئلهٔ پالایش همکارانه در سامانه‌های توصیه‌گر نیاز نیست و باعث پیچیدگی اضافه و افت عملکرد الگوریتم می‌شود. برای همین پژوهش‌های بسیاری پس از آن تلاش کردند تا با حذف یا تغییر بخشی از این جزئیات آن را بهبود بخشند. یکی از پرارجاع‌ترین کارهای انجام‌شده در این راستا شبکهٔ پیچشی گراف سبک[2] است که یک الگوریتم بر پایهٔ پالایش همکارانه گراف عصبی است که در حوزهٔ پالایش همکارانه برای ساخت سامانه‌های توصیه‌گر مورد استفاده قرار می‌گیرد [۵]. این الگوریتم مشابه پالایش همکارانهٔ گراف عصبی است با این تفاوت مهم که بردارهای جاساز لایه‌های بالاتر از رابطهٔ ۲.۲ به دست می‌آید.

$$\mathbf{e}_u^{(k+1)} = \sum_{i\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}\mathbf{e}_i^{(k)} \quad (۲.۲)$$

از آن‌جایی که این الگوریتم به‌طور گسترده در این پژوهش مورد استفاده قرار گرفته، جزئیات و روابط مربوط به آن در فصل روش پیشنهادی به تفصیل بررسی و در این فصل به معرفی اجمالی آن بسنده شده‌است.

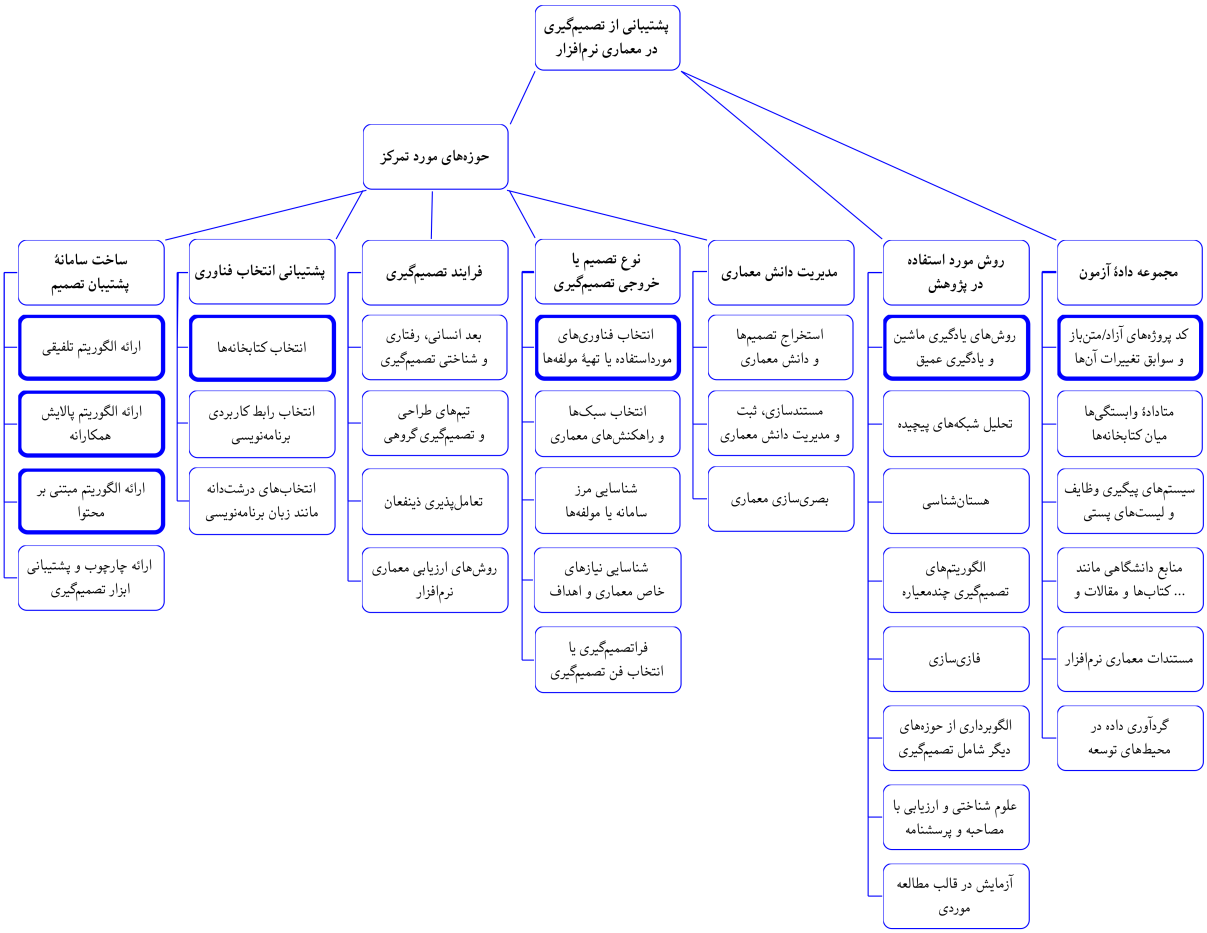
[1] sigmoid
[2] Light Graph Convolution Network Filtering

فصل ۳

کارهای پیشین

پژوهش در زمینهٔ مهندسی نرم‌افزار ترکیبی از پژوهش داده‌محور[1] و پژوهش در علوم اجتماعی[2] است. از یک سو برخی جنبه‌های توسعهٔ نرم‌افزار مانند افزایش صحت یا بهینه بودن الگوریتم‌ها، قابلیت اثبات ریاضی دارند از سوی دیگر ساخت نرم‌افزار یک فعالیت اجتماعی است که اعتبارسنجی و قطعیت در آن ممکن نیست. برای همین دو الگواره[3] را می‌توان برای پژوهش در زمینهٔ مهندسی نرم‌افزار متصور شد. الگوارهٔ اول از چشم‌انداز علوم رفتاری[4] است و الگوارهٔ دوم از چشم‌انداز علوم طراحی[5] است که در این الگواره، پژوهش به عنوان یک فعالیت حل مسئله دیده می‌شود. به‌طور خاص در پژوهش‌های حوزهٔ تصمیم‌گیری معماری نرم‌افزار نیز هر دو الگواره قابل مشاهده است.

۱.۳ طبقه‌بندی کارهای پیشین



شکل ۱.۳: درخت موضوعی پژوهش‌ها در زمینهٔ پشتیبانی از تصمیم‌گیری در معماری نرم‌افزار

حوزه‌های مورد تمرکز در پژوهش‌های پیشین تصمیم‌گیری در معماری نرم‌افزار، مطابق شکل ۱.۳ به چند دستهٔ

[1] data-driven
[2] social science
[3] paradigm
[4] behavioral science
[5] design science

زیر تقسیم می‌شوند:

- **ساخت سامانهٔ پشتیبان تصمیم:** این دسته از پژوهش‌ها به جنبه‌های فنی ابزارهای یاری‌رسان تصمیم‌گیری معماری نرم‌افزار پرداخته‌اند. پژوهش‌های [۶]، [۷]، [۸]، [۹] و [۱۰] دربارهٔ مسائل مرتبط با ساخت این ابزارها هستند و یک چارچوب برای این ابزارها پیشنهاد کردند یا یک نسخهٔ پیاده‌سازی‌شده از ابزار پیشنهادی خود را معرفی کرده‌اند. پژوهش [۱۱] مربوط به نحوهٔ استفاده از این سامانه در خط تولید محصولات نرم‌افزاری[1] است. همچنین یک دسته از پژوهش‌ها یکی از ابزارهای پیشنهاد شده را در شرایط واقعی آزمایش یا مورد مطالعهٔ موردی[2] قرار داده‌اند.

- **فن یا الگوریتم تصمیم‌گیری:** پژوهش‌های بسیاری دربارهٔ نحوهٔ به‌کارگرفتن الگوریتم‌های تصمیم‌گیری در زمینهٔ معماری نرم‌افزار انجام گرفته‌است. برخی از پژوهش‌ها به ارزیابی و بررسی نقاط قوت و ضعف هریک از این فنون پرداخته‌اند. برخی نیز یکی از این فنون را در یک مطالعهٔ موردی با داده‌های واقعی مورد استفاده قرار داده‌اند و درواقع یک مثال از به‌کارگیری یک فن تصمیم‌گیری ارائه کردند. هریک از فنون تصمیم‌گیری شاخصه‌های به‌خصوصی دارد که شناخت آن به انتخاب فن تصمیم‌گیری متناسب با خصوصیات مسئله کمک می‌کند. برخی از این شاخصه‌ها، ورودی‌ها و خروجی‌های فنون تصمیم‌گیری، میزان وابستگی و نقش انسان در آن‌ها، میزان حساسیت خروجی تصمیم‌گیری به ورودی آن، درجهٔ دشواری به‌کارگیری فن و امکان رتبه‌بندی انتخاب‌ها است.

- **فرایند تصمیم‌گیری:** پژوهش‌های [۱۲]، [۱۳]، [۱۴]، [۱۵] و [۱۶] به بررسی فرایند تصمیم‌گیری به معنای گام‌هایی که معمار باید برای رسیدن به تصمیمات معماری طی کند، پرداختند. روش‌های ارزیابی معماری مانند ATAM گام‌های مشخصی را برای ارزیابی معماری با استفاده از سناریو تعریف کردند. برخی از پژوهش‌ها به رفتار تصمیم‌گیری[3] توجه دارند نه استفاده از یک الگوریتم ریاضی برای رسیدن از یک ورودی به یک خروجی مشخص و نقش انسان در این پژوهش‌ها بسیار پررنگ‌تر است. پژوهش [۱۵]، [۱۲] و [۱۷] مستقیماً روی ابعاد انسانی تصمیم‌گیری تمرکز کردند و از علوم رفتاری و علوم شناختی و حتی روان‌شناسی

[1] software product line
[2] case study
[3] decision-making behaviour

نیز بهره گرفته‌اند. سوگیری‌های شناختی[1] معمار یکی از مسائل مورد توجه در این پژوهش‌هاست. در فرایند طراحی معماری معمولاً ذینفعان متعددی درگیر می‌شوند برای همین تعامل‌پذیری مناسب این افراد برای رسیدن به یک تصمیم مشخص نیز یکی از مسائل مورد توجه در این حوزه است. همچنین گاهی تصمیم‌گیری توسط یک تیم یا گروهی از افراد انجام می‌شود و خود این مسائلی ایجاد می‌کند. برخی از پژوهش‌های مربوط به تیم‌های طراحی مربوط به به‌کارگیری فنون سامانه‌های تصمیم‌گیری گروهی است که به همین خاطر این مورد می‌تواند ذیل فنون نیز قرار گیرد.

- **نوع تصمیم یا خروجی تصمیم‌گیری:** پژوهش‌های پیشین را می‌توان براساس تعریفی که از خود تصمیم‌های معماری دارند نیز دسته‌بندی کرد. پژوهش‌های [۱۸]، [۱۹] و [۲۰] تمرکزشان روی انتخاب راهکنش‌ها و سبک‌های معماری مورد استفاده در طراحی نرم‌افزار است. یکی دیگر از مسائل قابل توجه در این زمینه نحوهٔ تبدیل نیازمندی‌های خاص معماری به اهدافی مشخص برای طراحی معماری است که بتوان در جریان طراحی معماری، ارضا شدن آن اهداف را مورد بررسی قرار داد. به عبارت دیگر سازوکار مشخص کردن هدف در تصمیم‌گیری یک حوزهٔ مورد بررسی در پژوهش‌های پیشین است. در پژوهش‌های [۲۱] و [۱]، [۲۲]، [۲۳] و [۲۴]، [۲۵]، [۲۶] و [۲۷] تصمیم‌گیری دربارهٔ انتخاب فناوری‌های مورد استفاده مانند زبان‌های برنامه‌نویسی، چارچوب‌ها، زیرساخت‌های ابری، کتابخانه‌ها و ... و در پژوهش‌های [۲۸]، [۲۹] و [۳۰] روندهای تغییر کتابخانه‌ها در پروژه‌ها مورد بررسی قرار گرفته‌است. شناسایی مرز میان مولفه‌ها نیز یکی از تصمیماتی است که عموماً به عهدهٔ معمار است.

- **مدیریت دانش[2] معماری:** پژوهش‌های [۳۱]، [۳۲] و [۳]، [۳۳] و [۸] به جای توجه به ابزارهایی که مستقیماً به خود مرحلهٔ تصمیم‌گیری مرتبط هستند به روش‌ها و ابزارهایی پرداختند که در استخراج، مستندسازی و ثبت شدن تصمیم‌های معماری، توجیه پشت هر تصمیم و مجموعهٔ گزینه‌های جایگزین مورد استفاده قرار گیرند. یکی از مسائل مورد توجه در این پژوهش‌ها بصری‌سازی معماری نرم‌افزار است.

روش‌های انجام پژوهش‌ها و داده‌های آزمون

از نظر روش‌های مورد استفاده در انجام پژوهش‌ها نیز می‌توان پژوهش‌ها را به صورت زیر دسته‌بندی کرد:

[1] cognitive bias
[2] knowledge management

- **هستان‌شناسی:** پژوهش [۳۴]، [۳۵] و [۳۶] یک هستان‌شناسی معماری نرم‌افزار با هدف تسهیل درک مشترک از ساختارهای موجود میان اطلاعات این دامنه و استفادهٔ مجدد از دانش معماری ارائه کردند.

- **الگوریتم‌های تصمیم‌گیری چندمعیاره:** بخشی از پژوهش‌های انجام شده در زمینهٔ تصمیم‌گیری معماری نرم‌افزار معطوف به استفاده از روش‌های استنتاج[1] براساس دانش خبره[2] مثل روش‌های تصمیم‌گیری چندمعیاره هستند. در این روش‌ها می‌توان معیارهای تصمیم‌گیری را از نیازمندی‌های خاص معماری به ویژگی‌های کیفی یا نیازمندی‌های غیروظیفه‌ای و انتخاب‌های تصمیم‌گیری را به فناوری‌ها یا سبک‌های معماری تخفیف داد زیرا در بیشتر مواقع مبنای تصمیم‌های معماری، بده‌بستان‌ها[3] بین ویژگی‌های کیفی[4] هماورد است.

- **فازی‌سازی:** در منطق فازی به جای تخصیص مقادیر درست و غلط به متغیرها از هر عدد حقیقی در بازهٔ صفر تا یک می‌توان استفاده کرد. به این ترتیب یک گزاره می‌تواند تقریباً حقیقت داشته باشد یا به عبارت دیگر مقدار درستی یک گزاره می‌تواند در بازهٔ صفر تا یک باشد. فازی‌سازی به عنوان یک راه‌حل برای تصمیم‌گیری در محیط‌های دارای ابهام و عدم قطعیت که امکان تمیز دادن مفاهیم از هم وجود ندارد مورد استفاده قرار می‌گیرد. نقطهٔ مقابل روش‌های فازی، روش‌های خشک[5] است. تصمیم‌گیری معماری نرم‌افزار به خاطر تفاسیر متعددی که تصمیم‌گیرندگان انسانی از یک مفهوم مشترک دارند، یک مسئلهٔ غیرقطعی و نادقیق است [۳۷] [۳۸].

- **الگوبرداری از حوزه‌های دیگر شامل تصمیم‌گیری:** برای پژوهش در زمینهٔ تصمیم‌گیری معماری نرم‌افزار می‌توان به حوزه‌های دیگری که شامل تصمیم‌گیری می‌شوند نیز مراجعه کرد؛ مثلاً نظریه‌های اقتصاد کلاسیک فرض می‌کنند که مصرف‌کننده برمبنای منطق و بهینه‌سازی تصمیم‌گیری می‌کند؛ ولی بعدها محققین متوجه شدند که موارد دیگری مانند منطق محدود و تأثیرات شناختی می‌توانند روی تصمیم‌گیری تأثیر بگذارند [۳۹].

از نظر مجموعه دادهٔ آزمون مورد استفاده، بیشتر پژوهش‌های این حوزه به جستجو در متن منابع دانشگاهی مانند کتاب‌ها، مقالات و ... در پایگاه‌های اینترنتی پرداختند. داده‌های به دست‌آمده از مصاحبه با معماران و پرسشنامه در ردهٔ بعد برای مجموعه دادهٔ آزمون قرار می‌گیرد. استفاده از مصاحبه نسبت به پرسشنامه در مقالات جدیدتر

[1] reasoning
[2] expert knowledge
[3] trade-offs
[4] quality attribute
[5] crisp

محبوبیت بیشتری پیدا کرده‌است [۳۹]. برخی مستندات معماری نرم‌افزار در شرکت‌ها و سازمان‌ها را مورد بررسی قرار دادند مثلاً تصمیم‌های معماری را از روی متن آن‌ها استخراج کردند. برخی از پژوهش‌ها هم مجموعه دادهٔ آزمون را از محیط‌های توسعه گردآوری کردند. در این موارد استفاده از آزمایش نسبت به مطالعهٔ موردی در سال‌های اخیر محبوبیت بیشتری پیدا کرده‌است [۳۹].

۲.۳ پژوهش‌های پایه برای سامانهٔ توصیه‌گر کتابخانه

در بیشتر پژوهش‌های پایه انجام‌شده در زمینهٔ پیشنهاد کتابخانه از دو ابزار پم‌واکر[۱] یا از دادگان اندروید آرسنال[۲] استفاده شده‌است.

پژوهش لیب‌رک اولین مقاله‌ای است که دقیقاً به مسئلهٔ پیشنهاد کتابخانه پرداخته‌است. این مقاله کدها و دادگان ورودی یا خروجی خود را منتشر نکرده‌است ولی بر اساس گزارش‌های داخل متن از زبان برنامه‌نویسی سی‌شارپ برای پیاده‌سازی روش خود استفاده کرده‌است. معیار پشتیبانی[۳] یا درصدی از پروژه‌ها که از کتابخانه‌های یک کاربر جدید را هم استفاده کردند محاسبه می‌کند. سپس مشخص می‌کند چه پروژه‌هایی از دقیقاً همان کتابخانه‌های کاربر جدید استفاده کردند و چه پروژه‌هایی با دقت مشخصی از کتابخانه‌های کاربر جدید استفاده کردند سپس معیار اعتماد محاسبه می‌شود و تعدادی قانون استخراج می‌شود. این قوانین مشخص می‌کند که این کتابخانه‌ها زیاد رخ داده که با هم در یک پروژه بیایند. در مرحله بعد قوانین انجمنی از قوانین به دست‌آمده در مرحلهٔ قبل استخراج می‌شوند. هر کدام از این قوانین مشخص می‌کنند که اگر از چند کتابخانهٔ خاص استفاده کردید، احتمال زیادی دارد که به کتابخانهٔ دیگری هم نیاز داشته باشید.

پژوهش بعدی که به حل این مسئله پرداخته، لیب‌کاپ است. لیب‌کاپ کد خود را منتشر نکرده ولی یک ابزار برای بصری‌سازی[۴] دادگان و نتایج خود ایجاد کرده که آن را در یک صفحهٔ گیت‌هاب[۵] معرفی کرده‌است. پس از لیب‌کاپ، لیب‌فایندر با روشی متفاوت به این مسئله پرداخته‌است.

پژوهش بعدی کراس‌رک بوده‌است که از پالایش همکارانه استفاده کرده‌است. کراس‌رک با زبان برنامه‌نویسی

[1] https://github.com/raux/PomWalker
[2] https://github.com/malibdata/MALib-Dataset
[3] support
[4] visualization
[5] https://saiedmoh.github.io/LibCUP

جاوا توسعه یافته و کد خود را در گیت‌هاب[1] منتشر کرده‌است. کراس‌رک ابتدا تشابه همهٔ پروژه‌ها با پروژهٔ مورد نظر را با معیار شباهت کسینوسی[2] محاسبه می‌کند. سپس از این اطلاعات برای پیدا کردن پروژه‌ها با بیشترین میزان ارتباط با پروژهٔ مورد نظر یا همسایه‌های آن استفاده می‌کند و با استفاده ازپالایش همکارانه[3] بر اساس کتابخانه‌هایی که پروژه‌های همسایه استفاده کردند برای ارائهٔ فهرست سی عدد پیشنهاد استفاده می‌کند. نوآوری اصلی این پژوهش این است که برای هر کتابخانه مقدار فراوانی اصطلاح - معکوس فراوانی متن[4] را محاسبه کرده‌است که این باعث افزایش نوظهوری[5] پیشنهادها شده‌است؛ به عبارت دیگر کتابخانه‌ای که در کل بیشتر مورد استفاده قرار گرفته وزن کمتری خواهد داشت.

پژوهش بعدی، لیب‌سیک است. لیب‌سیک کد خود را با زبان‌برنامه‌نویسی متلب[6] نوشته و در گیت‌هاب[7] منتشر کرده‌است. پژوهش بعدی اندرولیب است که هیچ کدی منتشر نکرده‌است و بر خلاف روش‌های پایهٔ خود از دادگان پروژه‌های اندروید استفاده کرده‌است. معیارهای ارزیابی مورد توجه در پژوهش اندرولیب دقت و بازیابی هستند.

جی‌رک آخرین پژوهشی است که به این مسئله پرداخته‌است و هفت ماه پیش از نوشتن این پایان‌نامه منتشر شده‌است و برای اولین بار از یک الگوریتم مبتنی بر یادگیری عمیق برای تولید فهرست پیشنهادها استفاده کرده‌است و به همین واسطه، نسبت به تمام کارهای قبلی کارایی بسیار بالاتری دارد و به عنوان پژوهش پایه در این پایان‌نامه مرجع تمام مقایسه‌ها قرار گرفته‌است. جی‌رک برای پیاده‌سازی از زبان برنامه‌نویسی پایتون و کتابخانهٔ پایتورچ[8] استفاده می‌کند و کدش را در گیت‌هاب[9] منتشر کرده‌است.

۳.۳ جمع‌بندی کارهای پیشین

خلاصه‌ای از الگوریتم مورد استفاده، دادگان خام مورد استفاده، محدودیت‌های اعمال شده و اندازهٔ نمونهٔ مستخرج برای ارزیابی روش به ترتیب زمانی برای جمع‌بندی پژوهش‌های پیشین که به حل این مسئله پرداختند در جدول ۱.۳ مشاهده می‌شود.

[1] https://github.com/crossminer/CrossRec
[2] cosine similarity
[3] Collaborative Filtering
[4] tf-idf
[5] novelty
[6] MatLab
[7] https://github.com/libseek/LibSeek
[8] PyTorch
[9] https://github.com/fio1982/GRec

جدول ۱.۳: مقایسهٔ پژوهش‌های پایه

نام روش	سال انتشار	دادگان خام مورد استفاده، محدودیت‌های اعمال شده و اندازهٔ نمونهٔ مستخرج برای ارزیابی روش	الگوریتم مورد استفاده
LibRec [۴۰]	۲۰۱۳	پانصد عدد پروژهٔ گیت‌هاب با حداقل ده کتابخانه و ده هزار خط کد که انشعاب بقیه نباشد و وجود فایل pom در پروژه	ترکیب یادگیری قوانین انجمنی با پالایش همکارانه
LibCup [۲۲]	۲۰۱۸	۶۶۳۸ تا کتابخانه مِیوِن از بین ۴۰۹۳۶ تا کتابخانه و بررسی استفادهٔ آن‌ها در ۳۸۰۰۰ تا پروژهٔ گیت‌هاب. کتابخانه‌ها حداقل پنجاه تا شناسه برحسب نام متد، صفت و کلاس داشته باشند و پروژه‌ها حداقل هزار کامیت و ده هزار خط کد داشته باشند و انشعاب بقیه نباشد و فایل pom در پروژه وجود داشته باشد	یک روش خوشه‌بندی سلسله مراتبی[1] به نام خوشه‌بندی فاصله‌ای مبتنی‌بر چگالی برنامه‌ها با آلودگی[2] معروف به DBSCAN
LibFinder [۲۳]	۲۰۱۸	۶۰۸۳ تا کتابخانه مِیوِن در ۳۲۷۶۰ پروژهٔ گیت‌هاب	الگوریتم ژنتیک مرتب‌سازی بدون‌سلطه[3]

ادامه در صفحهٔ بعد

[1] hierarchical clustering

[2] Density-based spatial clustering of applications with noise

[3] non-dominated sorting genetic algorithm

جدول ۱.۳ — *ادامه از صفحهٔ قبل*

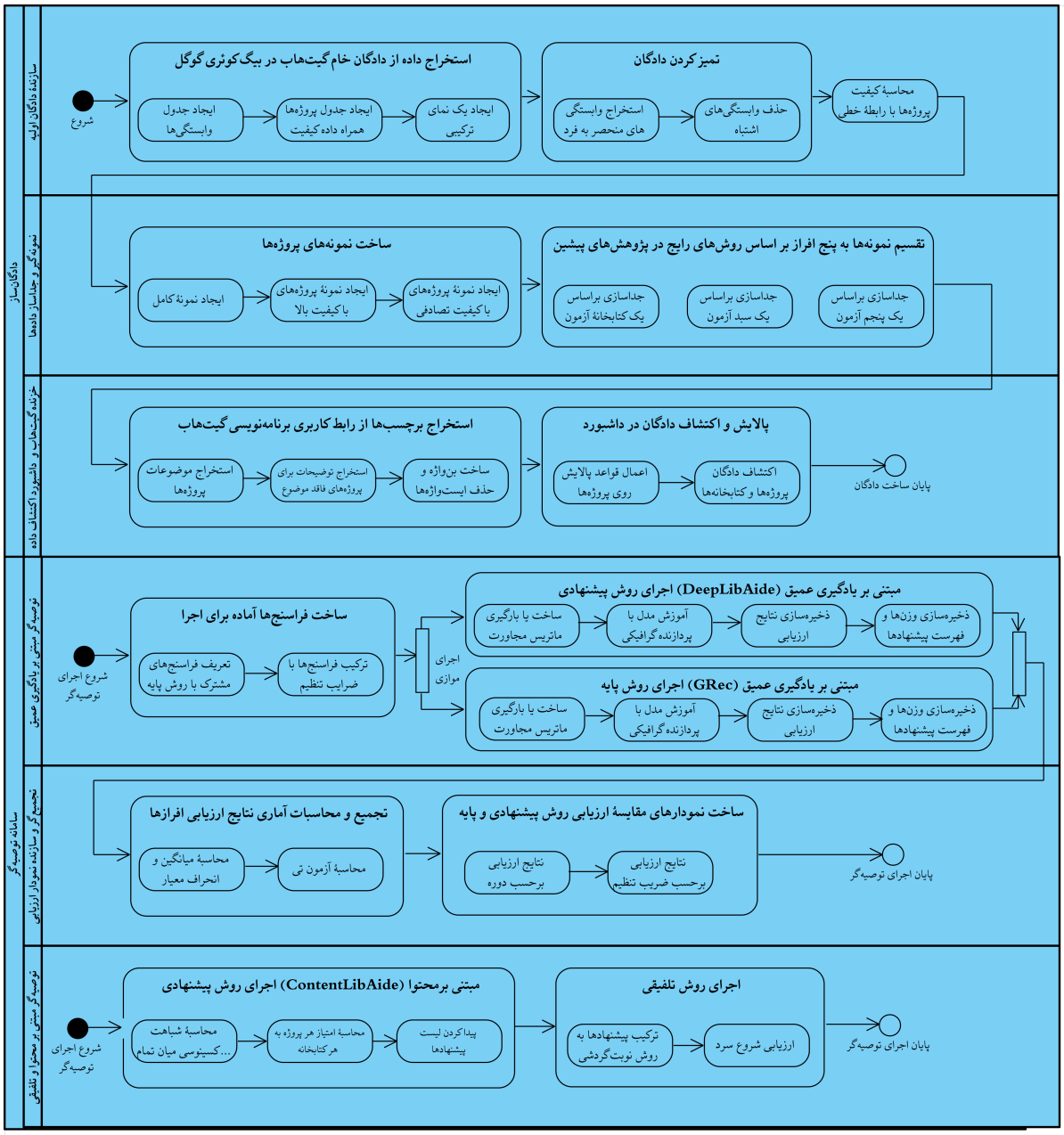
نام روش	سال انتشار	دادگان خام مورد استفاده، محدودیت‌های اعمال شده و اندازهٔ نمونهٔ مستخرج برای ارزیابی روش	الگوریتم مورد استفاده
CrossRec [۲۴]	۲۰۲۰	۱۲۰۰ تا پروژه تصادفی مستخرج از طریق رابط برنامه‌نویسی کاربردی گیت‌هاب	پالایش همکارانه مبتنی بر کاربر- کاربر با درنظر گرفتن شباهت کسینوسی و محاسبهٔ فراوانی اصطلاح - معکوس فراوانی متن
LibSeek [۴۱]	۲۰۲۰	دادگان اندروید آرسنال[1] حاوی پروژه‌های سیستم‌عامل اندروید	پالایش همکارانه
AndroLib [۴۲]	۲۰۲۰	دادگان ۲۷۵۲ تا برنامهٔ اندروید مستخرج از اندروید آرسنال و امتیاز ۹۷۹ تا کتابخانهٔ اندروید در گوگل پلی[2]	الگوریتم ژنتیک[3] توصیه مبتنی‌بر جستجو چندهدفه[4]
GRec [۴۳]	۲۰۲۱	۳۱۴۳۲ تا برنامهٔ اندروید و ۷۵۲ کتابخانه و ۵۳۷۰۱۱ مورد استفاده از کتابخانه در پروژه‌ها	پالایش همکارانه گراف عصبی search-based

[1] Android Arsenal
[2] Google Play
[3] genetic algorithm
[4] multi-objective recommendation

فصل ۴

روش پیشنهادی

در شکل ۱.۴ شمای کلی روش پیشنهادی در قالب نمودار فعالیت نمایش داده شده‌است. همان‌طور که در شکل مشخص‌شده روال کلی کار شامل دو بخش دادگان‌ساز و سامانهٔ توصیه‌گر است. دادگان‌ساز ورودی‌های سامانهٔ توصیه‌گر را تأمین می‌کند.



شکل ۱.۴: شمای کلی روش پیشنهادی در قالب نمودار فعالیت

ساخت دادگان به‌طور کلی شامل چند مرحلهٔ ساخت دادگان اولیه، ساخت نمونه و تقسیم نمونه‌ها به افرازها بر اساس سه روش رایج در پژوهش‌های پیشین، استخراج برچسب‌ها از رابط کاربری برنامه‌نویسی گیت‌هاب و پالایش

و اکتشاف دادگان در داشبورد به شکل بصری می‌شود. بخش سامانهٔ توصیه‌گر نیز ابتدا فراسنج‌های آماده برای اجرا را تولید می‌کند و سپس به‌طور موازی روش پیشنهادی مبتنی بر یادگیری عمیق که DeepLibAide نام‌گذاری شده و روش پایه مبتنی بر یادگیری عمیق که نام آن GRec است را به‌طور همزمان اجرا می‌کند. در مرحلهٔ بعد نتایج به دست‌آمده را تجمیع و یک سری محاسبات آماری بر اساس نتایج ارزیابی افرازها انجام می‌دهد. سپس نمودارهای مقایسهٔ ارزیابی بین روش پیشنهادی و پایه را رسم می‌کند. همچنین یک روش مبتنی بر محتوا به نام ContentLibAide نیز پیشنهاد شده‌است که در نهایت نتایج هر دو روش پیشنهادی در یک توصیه‌گر تلفیقی با هم ترکیب می‌شوند. در ادامه هر کدام از مراحل انجام کار در روش پیشنهادی به تفصیل مورد بررسی قرار می‌گیرد.

۱.۴ جمع‌آوری داده

در این بخش کارهای انجام شده برای رسیدن به داده‌های نهایی نمونه برای ارزیابی سامانه مورد استفاده قرار گرفته‌است. یکی از دستاوردهای این پژوهش ایجاد یک خط لوله است که تمام مراحل ابتدا تا انتهای ساخت دادگان برای آموزش مدل را زیر ده دقیقه اجرا می‌کند و می‌شود بدون هیچ دستکاری با فقط اجرای آن دادگان را بر اساس آخرین نسخهٔ داده‌های گیت‌هاب به‌روزرسانی کرد.

۱.۱.۴ دادگان‌های مورداستفاده

در بخش ۲.۱.۴ از دادهٔ کامل کامیت‌ها و کد محتوای کامیت‌ها برای بیش از سه میلیون پروژه سایت گیت‌هاب با نرخ به‌روزرسانی هفته‌ای یک‌بار[1] استفاده شده‌است. این دادگان توسط خود گیت‌هاب پشتیبانی و به‌روزرسانی می‌شود. در بخش ۳.۱.۴ از دادگان تمام رویدادها[2] در سایت گیت‌هاب با نرخ به‌روزرسانی روزانه[3] برای استخراج داده‌های مرتبط با کیفیت پروژه‌ها استفاده شده‌است. همچنین رابط برنامه‌نویسی کاربردی[4] سایت گیت‌هاب برای دریافت داده‌های موضوعات و متن توضیحات هر پروژه مورد استفاده قرار گرفته‌است.

[1] https://console.cloud.google.com/marketplace/product/github/github-repos

[2] event

[3] https://www.gharchive.org/

[4] API

۲.۱.۴ استخراج فناوری‌های مورد استفاده

در مخازن مِیوِن فایل‌های XML حاوی تمام داده‌های مورد نیاز برای ساختن[1] پروژه‌های جاوا در فایل‌هایی با نامی شبیه pom.xml ذخیره می‌شوند. برای استخراج محتوای فایل‌های pom.xml در پروژه‌های زبان جاوا، پرسمان نوشته شده در ضمیمهٔ ۴.۲.۶ تهیه و روی دادگان گیت‌هاب اجرا شد. در این پرسمان آخرین نسخهٔ محتوای فایل‌ها در دادگان درواقع وضعیت فایل‌ها در کامیتی است که HEAD به آن اشاره دارد. با استفاده از عبارات منظم هر فایلی که مسیر[2] آن به pom.xml ختم شود پیدا شد. سپس بررسی شد که محتوای فایل دودویی[3] نباشد. نام پروژه و مسیر فایل و محتوای آن برای پردازش‌های بعدی در یک جدول جدید ذخیره شد. در جدول خروجی این پرسمان هر پروژه یک تعداد فایل pom دارد و داخل هر pom مسیر هر فایل و محتوای آن در قالب آرایه‌ای از ساختارها مشخص شده‌است. همچنین در پرسمان بیان‌شده اطلاعات تمام وابستگی‌های موجود در پروژه‌ها شامل گروه[4]، مصنوع[5]، نسخه[6] و محدوده[7] استخراج شده‌است. نکات قابل توجه برای استخراج کتابخانه‌های هر پروژه به شرح زیر است:

- هر وابستگی داخل یک برچسب <dependency> نوشته می‌شود. همچنین موارد استثنا هم در برچسب‌های <dependency> نوشته می‌شوند که جزو وابستگی‌های پروژه نیستند. برای همین از داخل فایل همهٔ موارد <exclusions> حذف شده‌است.
- داخل هر فایل pom.xml یک تعداد وابستگی وجود دارد و هر وابستگی گروه، مصنوع، نسخه و محدوده دارد. (آرایه‌ای از ساختارها)
- تعداد فایل‌های pom.xml در هر پروژه، تعداد کل وابستگی‌های هر پروژه، تعداد وابستگی‌ها در هر فایل pom.xml به‌صورت مرتب‌شده از زیاد به کم در جدول خروجی مشخص شده‌است.
- در بسیاری از موارد نسخهٔ یک وابستگی به جای قرار گرفتن در برچسب <version> در یک برچسب دیگر قرار گرفته و نام آن برچسب در برچسب <version> نوشته شده‌است. در پرسمان بیان‌شده مقدار این نسخه‌ها با استفاده از عبارات منظم پیدا شده و در جدول خروجی قرار گرفته‌است.

[1] build
[2] path
[3] binary file
[4] group
[5] artifact
[6] version
[7] scope

- پروژه‌هایی که کمتر از پنج وابستگی دارند برای پردازش‌های بعدی مناسب نیستند و از دادگان حذف شدند.

در این گام تعداد فایل pom.xml در هر پروژه، تعداد فناوری‌های استفاده شده در هر فایل pom.xml و تعداد کل فناوری‌های یکتای استفاده شده در هر پروژه نیز استخراج شده‌است.

۱.۲.۱.۴ حذف موارد کتابخانه‌های اشتباه با تطبیق الگو

بر اساس قراردادهای[1] پروژه‌ها مِیوِن برای نوشتن متن شناسهٔ مصنوع و شناسهٔ سازمان[2]، الگوی عبارت منظم زیر ایجاد شد:

```
^[a-zA-Z][a-zA-Z._0-9-]+:[a-zA-Z._0-9-]+$
```

سپس با استفاده از الگوی بالا همهٔ کتابخانه‌های اشتباه استخراج‌شده از گیت‌هاب از دادگان حذف شدند. آخرین پژوهش پایه انجام شده در این حوزه که از داده‌های پروژه‌ها مِیوِن برای حل این مسئله استفاده کرده بود مقالهٔ کراس‌رک بود و این مقاله و کارهای پیش از آن هیچ‌کدام به وجود داده‌های اشتباه در کتابخانه‌ها نپرداختند. نمونه‌هایی از برخی از مواردی که بیشترین تعداد استفاده را داشتند و با این روش از دادگان حذف شدند در ادامه نوشته شده‌است:

```
org.scalatest:scalatest_${scala.binary.version}
${project.groupId}:naked-objects-fixture
${groupId}:${artifactId}-api
```

۳.۱.۴ استخراج دادهٔ کیفیت و اندازهٔ پروژه‌ها

در گام قبل نام پروژه‌ها و مسیر و فهرستی از کتابخانه‌های به‌کار رفته در پروژه‌ها و نسخه‌های هریک از آن‌ها به دست آمد. در این گام با داشتن اطلاعاتی دربارهٔ کیفیت پروژه‌هایی که در جدول وابستگی‌ها، نام آن‌ها وجود دارد می‌توان مدل را با توجه به داده‌های پروژه‌های معتبر یا دارای اندازهٔ بیشتر به‌طور جداگانه آموزش داد.

۱.۳.۱.۴ فرایند استخراج دادهٔ کیفیت و اندازهٔ پروژه‌ها

در این گام دادهٔ کیفیت و اندازهٔ همهٔ پروژه‌ها استخراج شده‌است. این اطلاعات شامل تعداد افرادی که روی هر پروژه کار کردند، تعداد کامیت‌ها و فاصلهٔ زمانی اولین و آخرین کامیت، تعداد افراد و کامیت‌های یک‌سال اخیر

[1] conventions

[2] https://maven.apache.org/guides/mini/guide-naming-conventions.html

که بیشتر به اندازهٔ پروژه اشاره دارند و دادهٔ تعداد انشعاب‌ها و تعداد ستاره‌های اعطا شده توسط کاربران به پروژه‌ها که بیشتر بیانگر کیفیت و محبوبیت آن‌ها هستند است. تولید این دادگان حاصل تلفیق چندین دادگان بسیار بزرگ در بیگ‌کوئری است که برخی از آن‌ها به‌طور رسمی توسط خود گیت‌هاب منتشر و به‌صورت دوره‌ای به روزرسانی می‌شوند است که با اجرای فقط یک پرسمان بزرگ نوشته‌شده در ضمیمهٔ ۴.۲.۶ به دست می‌آید.

در پرسمان ۴.۲.۶ برای به دست آوردن تعداد ستاره‌ها تعداد رویدادهای WatchEvent و برای تعداد انشعاب تعداد رویدادهای ForkEvent رخ‌داده توسط کاربران یکتا برای هر پروژه شمارش می‌شود. همچنین زبان اول هر پروژه زبانی است که بیشترین حجم بایت[1] از محتوای پروژه به آن تخصیص یافته‌است. در مواردی گیت‌هاب هیچ زبانی را نتوانسته تشخیص دهد که در این صورت مقدار پوچ[2] درج شده‌است. همچنین باید توجه کرد که یک کامیت ممکن است در چند پروژهٔ مختلف باشد مثلاً در حالتی‌که پروژه‌ها از روی یکدیگر انشعاب شده‌باشند یا ممکن است نام یک پروژه در طی زمان تغییر کرده باشد. در نهایت دادگان حاوی ۱۴۳,۸۸۰ پروژهٔ گیت‌هاب با دادهٔ هفتهٔ اخیر گیت‌هاب شد.

۲.۳.۱.۴ حذف پروژه‌های بی‌فایده برای مراحل بعد

در ادامه فهرستی از مخاطرات مربوط به استفاده از داده‌های عمومی گیت‌هاب برای انجام پژوهش‌ها مشخص شده‌است [۴۴]. وجود این مخاطرات انگیزهٔ اصلی از استفاده از دادهٔ کیفیت و اندازهٔ پروژه‌ها بوده‌است.

۱. بیشتر پروژه‌های داخل گیت‌هاب فعال نیستند و تمام کامیت‌هایشان مربوط به مثلاً یک هفته بعد از ساخته شدنشان است و بعد یک هفته رها شده‌اند.

۲. بیشتر پروژه‌ها فقط یک نفر توسعه‌دهنده دارند فقط و علیرغم این‌که گیت‌هاب به عنوان یک شبکهٔ اجتماعی[3] طراحی شده کسی خیلی از این قابلیت‌های گیت‌هاب استفاده نمی‌کند.

۳. بسیاری از پروژه‌های گیت‌هاب درواقع اصلاً پروژهٔ گیت نیستند و به عنوان یک میزبان مجانی فایل از آن استفاده شده‌است یا این‌که سایت‌های ایستا روی آن میزبانی شده‌است. بسیاری از پروژه‌های گیت‌هاب تمرین‌های درسی و دانشگاه هستند نه پروژهٔ صنعتی و واقعی.

[1] byte
[2] null
[3] social network

۴. کامیت‌های بسیاری در گیت‌هاب توسط افرادی انجام گرفته که عضو گیت‌هاب نیستند. در این موارد به جای نام کاربری فرد کامیت‌کننده آدرس ایمیلش درج می‌شود و در این موارد معمولاً سامانهٔ پیگیری وظایف و ... ای که استفاده شده که جدا از گیت‌هاب است. مثال معروف آن پروژه‌های بنیاد موزیلا است که سامانهٔ پیگیری ایرادها[1] خودش را دارد و از گیت‌هاب پروژه‌های بنیاد موزیلا کاملاً جداست. برای همین نمی‌شود حکم کلی داد و فرض کرد که همهٔ پروژه‌ها از قابلیت‌های درخواست واکشی[2] و بازبینی کد[3] و سامانهٔ پیگیری وظایف[4] گیت‌هاب حتماً استفاده می‌کنند.

در نهایت فقط پروژه‌هایی که زبان اصلیشان جاوا بوده، بیشتر از هفت تا کتابخانه دارند، بیشتر از سی تا کامیت دارند و زیر ۷۵ تا فایل pom دارند نگه داشته شده‌است زیرا پروژه‌هایی که بیشتر از این فایل pom داشتند اغلب مخزن نگه‌داری فایل‌های pom بودند که به مسئلهٔ توصیه‌گر کمکی نمی‌کند.

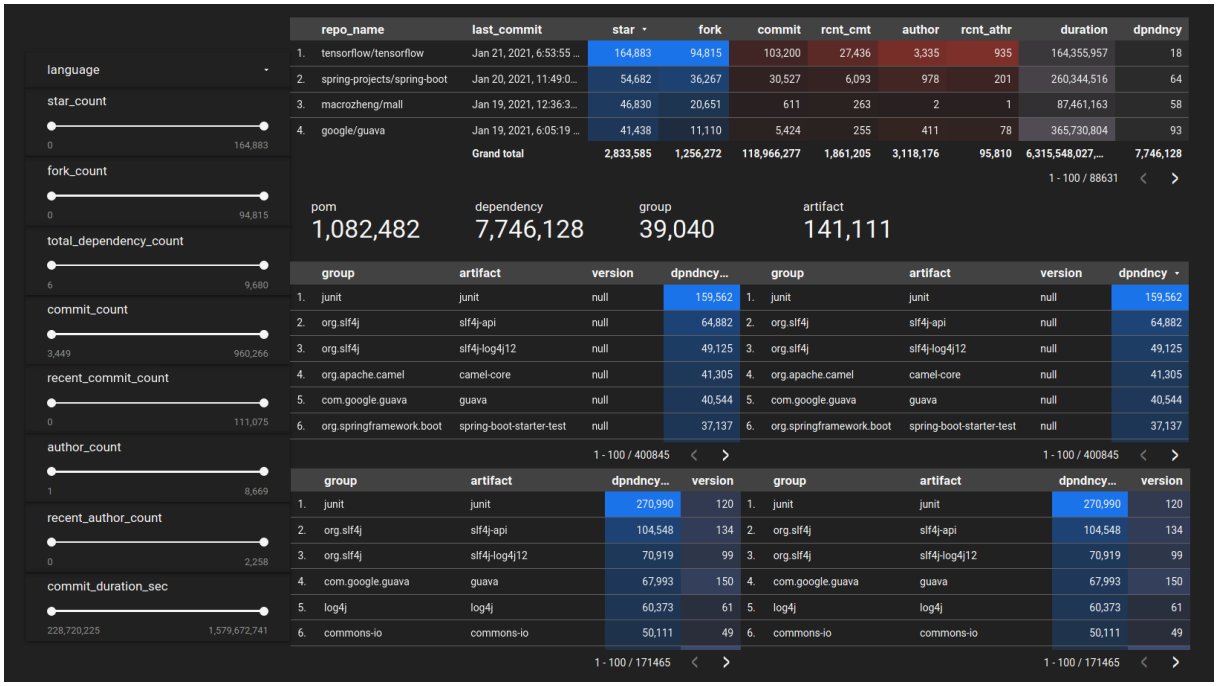
۴.۱.۴ استخراج دادهٔ برچسب‌ها برای پروژه‌ها

در این گام موضوعات برچسب‌خورده به پروژه‌ها از طریق رابط کاربری برنامه‌نویسی گیت‌هاب استخراج شد. برای پروژه‌هایی که هیچ موضوعی به آن‌ها برچسب نخورده بود متن توضیحات پروژه‌ها دریافت شد و با استفاده از فنون پردازش زبان طبیعی[5]، بن‌واژهٔ آن‌ها پیدا شد و ایست‌واژه‌ها از بین آن‌ها حذف شد. در نهایت برچسب‌های به دست‌آمده برای پروژه‌ها به پایگاه داده اضافه شد. هدف از استخراج این داده این است که در یک روش مبتنی بر محتوا[6] برای ارائهٔ پیشنهاد به کاربر در کنار روش پالایش همگانی مبتنی بر یادگیری عمیق مورد استفاده قرار گیرد.

۲.۴ ساخت داشبورد تعاملی برای اکتشاف داده‌ها

داشبورد شکل ۲.۴ طراحی شد تا تمام داده‌های تهیه شده در بخش ۳.۱.۴ و ۲.۱.۴ را بتوان در آن دید و براساس معیارهای مختلف دربارهٔ پروژه‌ها که در ستون سمت چپ صفحه قرار گرفته‌اند، آن‌ها را فیلتر کرد. همچنین برای ایجاد امکان مقایسه در حالت فیلتر شده با حالت بدون اعمال فیلترها جداول وابستگی‌ها در سطر دوم و سوم در دو حالت فیلترشده و بدون اعمال فیلتر در کنار یکدیگر قرار دارند. جدول سطر اول در داشبورد مربوط به پروژه‌ها و اطلاعات آن‌ها است. در صورت تقه‌زدن روی هریک از پروژه‌ها اطلاعات جداول سطر دو و سطر سه فقط برای همان

[1] bug tracker system
[2] pull request
[3] code review
[4] issue tracking system
[5] natural language processing
[6] content based



شکل ۲.۴: داشبورد تحلیل و مقایسهٔ فناوری‌های به‌کار رفته

پروژه به نمایش در می‌آید. جداول سطر دو تعداد وابستگی‌ها برای هر نسخه از هر فناوری در پروژه‌ها و جداول سطر سه تعداد کل وابستگی‌ها و تعداد ورژن‌های به‌کار رفته از هر فناوری را نمایش می‌دهد. درواقع با دقت به جداول سطر دو و سه می‌توان دریافت که (الف) هر فناوری چند بار استفاده شده‌است؟ (ب) هر نسخه از هر فناوری چند بار استفاده شده‌است؟ (ج) چند نسخه از هر فناوری در پروژه‌های مختلف استفاده شده‌است؟

همچنین در داشبورد شکل ۲.۴ مشخص شده‌است که تعداد کل پروژه‌های موجود در گیت‌هاب که حداقل پنج وابستگی دارند ۸۸۶۳۱ تا است. هریک از این پروژه‌ها حداقل یک فایل pom.xml دارند. همچنین در کل ۱۰۸۲۴۸۲ تا فایل pom.xml و در این فایل‌ها در کل تعداد ۷۷۴۶۱۲۸ تا وابستگی وجود دارد.

۳.۴ ارائهٔ سامانهٔ توصیه‌گر بهبود یافته

در این فصل روش پیشنهادی توصیه‌گر DeepLibAide مبتنی بر یادگیری عمیق و ContentLibAide مبتنی‌بر محتوا و روش مورد استفاده برای تلفیق فهرست پیشنهادهای این دو سامانه بیان خواهد شد.

اگر مجموعهٔ کل کتابخانه‌های موجود به تعداد m که می‌شود برای یک پروژهٔ فرضی p پیشنهاد کرد را L و تعداد کل پیشنهادهایی که سامانه برای آن پروژه تولید می‌کند را k بنامیم، خروجی این دو سامانهٔ توصیه‌گر دو

فهرست مرتب‌شده از اعضای مجموعهٔ L است که در رابطهٔ نوشته‌شده در فرمول ۱.۴ با نام‌های $O_{DeepLibAide}$ و $O_{ContentLibAide}$ مشخص شدند. همچنین $(R_{ij})_{1\leq i\leq n,1\leq j\leq m}$ بیان‌گر استفاده یا عدم استفاده از کتابخانهٔ l_j در پروژهٔ p_i در دادگان ورودی است.

$$P = \{p_1, p_2, \cdots, p_n\}$$

$$L = \{l_1, l_2, \cdots, l_m\}$$

$$R_{ij} = \begin{cases} 1 & \text{استفاده} \\ NULL & \text{عدم استفاده} \end{cases} \tag{۱.۴}$$

$$O(k, L; p) = Hybrid(O_{DeepLibAide}(k, L; p), O_{ContentLibAide}(k, L; p))$$

۱.۳.۴ ترکیب خطی داده‌های کیفیت پروژه‌ها

یکی از موضوعات مورد بررسی در بخش‌های بعدی تاثیر کیفیت پروژه‌های موجود در دادگان ورودی بر توصیه‌های خروجی سامانه است. برای این منظور به یک عدد نهایی به عنوان مجموع امتیاز کیفیت هر پروژه نیاز است و در ادامه بر اساس این عدد، پروژه‌ها براساس محبوبیت و کیفیت آن‌ها دسته‌بندی شده‌است و می‌توان نتایج ارزیابی سامانهٔ توصیه‌گر را برای هر کدام را با دیگری مقایسه کرد.

$$q(p, j) = \frac{c_{pj}}{max\left[(c_{ij})_{1\leq i\leq P}\right]} \tag{۲.۴}$$

اطلاعات استخراج‌شده دربارهٔ هر پروژه شامل تعداد ستاره، انشعاب، نویسنده، کامیت، نویسنده‌ها و کامیت‌های یک سال اخیر و مدت زمان عمر پروژه، مقیاس و واحد متفاوتی دارند مثلا برای ستاره‌های اعطا شده به یک پروژه، تعداد آن‌ها شمارش می‌شود ولی برای مدت زمان عمر یک پروژه تعداد ثانیه‌های بازهٔ زمانی بین اولین و آخرین کامیت در آن پروژه بررسی می‌شود و معمولا تعداد کامیت‌ها از تعداد نویسنده‌های یک پروژه بسیار بیشتر است. در رابطهٔ ۲.۴ هر کدام از این فقره‌های اطلاعات به‌صورت c_{ij} به عنوان مقدار یک ویژگی کیفی از یک پروژه در یک ماتریسِ پروژه

- کیفیت بازنمایی شده‌است و P تعداد کل پروژه‌ها است و حاصل این رابطه یک مقدار نرمال‌شده برای وضعیت یک پروژه در خصوص یکی از معیارهای کیفی بیان‌شده است.

$$\mathbf{S}_p = \frac{\sum_{j\in 1}^{Q} w_j \times q(p,j)}{\sum_{j\in 1}^{Q} w_j} \tag{۳.۴}$$

رابطهٔ ۳.۴ یک میانگین وزنی از تمام اطلاعات به دست آمده دربارهٔ یک پروژه است و حاصل آن یک عدد امتیاز در بازهٔ صفر تا یک برای هر پروژه است. در این رابطه w_j ضرایب مورد نظر برای هر کدام از ویژگی‌های کیفی است که به عنوان میزان اهمیت هر کدام از ویژگی‌های کیفی می‌تواند توسط کاربر مشخص شود و به این ترتیب کاربر مثلاً می‌تواند درخواست کند که «بر اساس دادهٔ پروژه‌های باکیفیت که از کتابخانه‌هایی مشابه من استفاده کرده‌اند به من کتابخانه‌هایی را پیشنهاد کن و معیار محاسبهٔ کیفیت هم برای من این‌طور است که قدمت پروژه و تعداد ستاره‌ها دو برابر سایر موارد اهمیت دارند.»

در این مرحله از دادگان گیت‌هاب، سه تا نمونه برای ارزیابی برنامهٔ نوشته‌شده در این پژوهش و برنامهٔ روش پایه استخراج شده‌است. این نمونه‌ها «کامل» و «با کیفیت» و «با کیفیت متوسط» نام‌گذاری شده‌اند. در نمونهٔ کامل ۳۱۴۰۰ تا پروژه وجود دارد که با تعداد پروژه‌های جی‌رک یا بیس‌لاین فعلی دقیقاً مساوی است. در نمونهٔ «با کیفیت» و «با کیفیت متوسط» هم ۶۲۸۰ تا پروژه وجود دارد که اولی درواقع بیست درصد پروژه‌هایی است که بیشترین تعداد ستاره را دارند و دومی هم به همان تعداد پروژه با کیفیت تصادفی انتخاب شده‌است. به این ترتیب می‌شود بررسی کرد که آموزش دادن مدل با پروژه‌های با کیفیت‌تر چه تأثیری روی نتایج ارزیابی خواهد داشت.

۲.۳.۴ استفاده از روش‌های رایج اعتبارسنجی متقابل

در این پژوهش برای اطمینان از سوگیری نداشتن نتایج نسبت به بخشی از دادگان، از سه روش رایج اعتبارسنجی متقابل[1] در ادبیات حوزهٔ سامانه‌های توصیه‌گر شامل روش «یک قلم آخر را بیرون بگذار[2]» و روش «یک سبد را بیرون بگذار[3]» و روش «جداسازی تقسیم زمانی مبتنی‌بر کاربر[4]» استفاده شده‌است [۴۵]. در هر کدام از این روش‌ها، دادگان به پنج افراز[5] تقسیم می‌شود که به این ترتیب جمعاً ۱۵ افراز وجود دارد. روش تقسیم دادگان در هر کدام

[1] cross validation
[2] leave-one-last-item-out
[3] leave-one-basket-out
[4] temporal-split-user-based
[5] fold

از پنج افراز به روش مبتنی بر کاربر است به این‌صورت که بخشی از کتابخانه‌های مورد استفاده توسط هر پروژه، به عنوان دادهٔ آزمون و سایر کتابخانه‌های مورد استفاده در پروژه‌ها به عنوان دادهٔ آموزش برای آموزش دادن مدل استفاده می‌شود. در هر کدام از پنج افراز بخشی از داده‌های هر پروژه که به عنوان دادهٔ آزمون استفاده می‌شود با سایر افرازها متفاوت است. در مواردی که تعداد کتابخانه‌های یک پروژه کم است و برای ساخت پنج افراز کافی نیست، دادهٔ آزمون در افرازها می‌تواند با یکدیگر همپوشانی داشته باشد ولی در مواردی که به تعداد کتابخانه‌های یک پروژه در دادگان ورودی کافی است دادهٔ آزمون در افرازها همپوشانی ندارند.

روش پایه و پیشنهادشده در این پژوهش، در هر کدام از این ۱۵ افراز برای مقایسهٔ نتایج ارزیابی، باید اجرا می‌شدند که به این ترتیب نیاز به سی بار اجرای برنامه است و این سی بار اجرا باید روی هر یک از دادگان‌های با کیفیت بالا و با کیفیت متوسط و دادگان کامل که در بخش ۱.۳.۴ معرفی شد، اجرا شود که به این ترتیب برای اطمینان در نهایت به نَود بار اجرای برنامهٔ توصیه‌گر نیاز بوده‌است. برنامه برای هر کدام از این نود افراز به‌طور جداگانه اجرا می‌شود و میانگین و انحراف معیار نتایج ارزیابی بین افرازها محاسبه می‌شود.

۱.۲.۳.۴ روش یک قلم آخر را بیرون بگذار

این روش برای جداسازی دادگان به این‌صورت است که در هر افراز برای هر پروژه فقط یک کتابخانهٔ آزمون وجود دارد که از آن کتابخانه استفاده کرده‌است به عبارت دیگر یک پیشنهاد درست برای هر پروژه وجود دارد که در ارزیابی سامانه انتظار می‌رود که آن را پیشنهاد کند و سایر کتابخانه‌های استفاده‌شده در آن پروژه برای آموزش استفاده می‌شوند. این روش جداسازی داده‌ها در پژوهش پایه جی‌رک استفاده شده‌است و نسبت به جداسازی براساس درصد برای ارزیابی سامانه‌های توصیه‌گر روش رایج‌تری است. مزیت این روش این است که نتایج ارزیابی آن هم برای هر افراز خیلی اختلاف زیاد ندارد و بیشتر دادگان برای آموزش باقی می‌ماند ولی اشکال این روش این است که برای این‌که تمام دادگان موجود یک‌بار به عنوان دادهٔ آزمون مورد استفاده قرار گیرد باید دادگان را به افرازهای بسیار زیادی تقسیم کرد که امکان اجرای آن ممکن نیست.

۲.۲.۳.۴ روش یک سبد را بیرون بگذار

این روش برای جداسازی دادگان به این‌صورت است که در هر افراز برای هر پروژه یک سبد از کتابخانه‌های آزمون وجود دارد که از آن کتابخانه‌ها استفاده کرده‌است به عبارت دیگر یک سبد از پیشنهادهای درست برای هر

پروژه وجود دارد که در ارزیابی سامانه انتظار می‌رود که آن را پیشنهاد کند و سایر کتابخانه‌های استفاده‌شده در آن پروژه برای آموزش استفاده می‌شوند. در این مسئله اندازهٔ هر سبد برابر پنج کتابخانه در نظر گرفته شده‌است. این روش جداسازی داده‌ها در پژوهش الگوریتم شبکهٔ پیچشی گراف سبک و پژوهش پایهٔ آن استفاده شده‌است و نسبت به جداسازی براساس درصد برای ارزیابی سامانه‌های توصیه‌گر روش رایج‌تری است. این روش نسبت به روش یک قلم آخر را بیرون بگذار تعداد افرازهای کمتری تولید می‌کند و برای همین اجرای آن سریع‌تر است.

۳.۲.۳.۴ روش جداسازی تقسیم زمانی مبتنی‌بر کاربر

این روش برای جداسازی دادگان به این‌صورت است که در هر افراز برای هر پروژه درصد مشخصی از کتابخانه‌های آن به عنوان دادهٔ آزمون وجود دارد که از آن کتابخانه‌ها استفاده کرده‌است به این ترتیب کتابخانه‌های دادهٔ آزمون برای هر پروژه با پروژهٔ دیگر می‌تواند متفاوت باشد که در ارزیابی سامانه انتظار می‌رود که آن‌ها را پیشنهاد کند و سایر کتابخانه‌های استفاده‌شده در آن پروژه برای آموزش استفاده می‌شوند. در این مسئله یک پنجم کتابخانه‌های هر پروژه به عنوان دادهٔ آزمون در نظر گرفته شده‌است. مزیت این روش این است که به تعداد افرازهای کمتری نیاز دارد و درنتیجه خیلی سریع‌تر اجرای آن تمام می‌شود و کل دادگان ورودی را هم پوشش می‌دهد یا به عبارت دیگر کل دادگان حتماً یک‌بار به عنوان دادهٔ آزمون مورد استفاده قرار می‌گیرد. اما اشکال این روش این است که نسبت به روش یک قلم را بیرون بگذار، کتابخانه‌های بیشتری را برای دادهٔ آزمون استفاده می‌کند و چون تعداد داده‌های آزمون برای هر پروژه با پروژهٔ دیگر اختلاف دارد، اعدادی که به عنوان نتایج ارزیابی برای هر پروژه محاسبه می‌شوند، انحراف معیار زیادی پیدا می‌کنند.

۳.۳.۴ تقسیم‌بندی داده به دادهٔ آزمون، آموزش و اعتبارسنجی

به جز اعتبارسنجی متقابل برای داده‌های آزمون و آموزش برای تفکیک دادهٔ اعتبارسنجی[1] از دادهٔ آموزش به این‌صورت عمل شده‌است که ماتریس مجاورت حاصل از تعامل میان پروژه‌ها و کتابخانه‌ها به صد بخش مساوی تقسیم شده‌است و هر بار یک بخش آن به عنوان دادهٔ اعتبارسنجی سامانه کنار گذاشته می‌شود و از ۹۹ بخش دیگر برای آموزش سامانه استفاده می‌شود. برای محاسبهٔ معیارهای ارزیابی، پیشنهادهای خروجی سامانه در نهایت با دادهٔ آزمون مقایسه می‌شود و همین مراحل برای چهار قسمت دیگر نیز تکرار و نتایج ارزیابی دوباره محاسبه می‌شود و

[1] validation data

میانگین گرفته می‌شود. به این ترتیب سعی شده که حتی‌الامکان کل دادگان موجود در هر نمونه حداقل یک‌بار به عنوان دادهٔ آموزش و حداقل یک‌بار به عنوان دادهٔ آزمون و اعتبارسنجی مورد استفاده قرار گرفته باشد.

۴.۳.۴ بازنمایی ماتریس مجاورت دادهٔ آزمون

به طور کلی دادگان ورودی مورد استفاده در الگوریتم‌های پالایش همکارانه شامل مجموعهٔ کاربران و اقلام و روابط میان آن‌هاست. در این مسئله کاربران به پروژه‌ها، اقلام به کتابخانه‌ها و روابط میان آن‌ها به استفاده از یک کتابخانه در یک پروژه نگاشت می‌شود که از یک گراف دو بخشی که راس‌ها در یک بخش آن شامل فقط پروژه‌ها و در بخش دیگر شامل فقط کتابخانه‌ها می‌شود و یال‌ها میان پروژه و کتابخانه بیانگر استفاده از کتابخانه در پروژه است، می‌توان برای نمایش آن استفاده کرد.

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix} \tag{۴.۴}$$

ماتریس مجاورت این گراف دو بخشی در رابطهٔ ۴.۴ بیان شده‌است. سطرها و ستون‌های این ماتریس با ترتیب یکسانی ابتدا پروژه‌ها و بعد کتابخانه‌ها را نمایندگی می‌کند یا به عبارت دیگر تعداد سطرها و ستون‌های ماتریس برابر مجموع تعداد پروژه‌ها و کتابخانه‌ها است و هر خانهٔ ماتریس وجود یا عدم وجود یال بین دو راس را با یک و صفر مشخص می‌کند. این ماتریس به عنوان ورودی الگوریتم شبکهٔ پیچشی گراف سبک برای آموزش استفاده می‌شود و به این ترتیب بردار جاساز برای یک پروژه یا یک کتابخانه در اولین لایهٔ شبکه، سطر مربوط به آن پروژه یا کتابخانه در ماتریس مجاورت رابطهٔ ۴.۴ است.

۵.۳.۴ آموزش مدل یادگیری عمیق

در این فاز یک شبکهٔ پیچشی گراف سبک آموزش داده شده‌است که DeepLibAide نام‌گذاری شده‌است. علت کارایی بالای روش مورد استفاده در این پژوهش، استفاده از شبکهٔ پیچشی گراف سبک است. این الگوریتم آخرین دستاورد موجود در حوزهٔ کاربرد یادگیری عمیق برای ساخت سامانه‌های توصیه‌گر است. شبکهٔ پیچشی گراف سبک به خاطر استفاده از تابع فعال‌سازی خطی[1] و جمع‌زدن بردارهای جاساز به دست آمده از لایه‌های مختلف با

[1] linear activation function

ضرایب مشخص مطابق با رابطهٔ ۵.۴ و حذف توابع وزن عملکرد بسیار بهتری نسبت به روش پایهٔ جی‌رک دارد و در صورت افزایش تعداد لایه‌ها کمتر دچار مشکل بیش‌همواری[1] می‌شود. الگوریتم پالایش همکارانه گراف عصبی به خاطر چسباندن بردارهای جاساز به دست‌آمده از لایه‌های مختلف برای رسیدن به بردار جاساز نهایی، دچار مشکل بیش‌همواری است ولی در الگوریتم مورد استفاده در این پژوهش بردارهای جاساز با ضریب یک تقسیم بر یک به علاوهٔ شمارهٔ لایه با هم جمع می‌شوند به این ترتیب علاوه بر استفاده از دادهٔ ارتباطات لایه‌های بالاتر میان پروژه‌ها و کتابخانه‌ها، تاثیر لایه‌های بالاتر نسبت به لایه‌های پایین‌تر یادگیری عمیق کاهش می‌یابد.

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)} \tag{۵.۴}$$

همان‌طور که از مقایسهٔ رابطه‌های ۱.۲ و ۲.۲ مشخص می‌شود، پیچیدگی‌های غیرضروری طراحی الگوریتم مورد استفاده در پژوهش پایه برای پالایش گروهی باعث کاهش عملکرد و کارایی آن شده‌است که با حذف آن‌ها عملکرد بهتری حاصل می‌شود. روش فعلی از دو جزء ضروری تشکیل شده‌است. این دو جزء گراف پیچشی سبک و ترکیب لایه‌ها است. تابع فعال‌سازی غیرخطی و تبدیل بردارهای ویژگی‌ها با توابع وزن کنار گذاشته شده‌است. این‌ها دو عملیات استاندارد در شبکهٔ پیچشی گراف هستند که باعث افزایش پیچیدگی مرحلهٔ آموزش مدل می‌شود.

الگوریتم مورد استفاده در توصیه‌گر DeepLibAide برای ارائهٔ فهرست پیشنهادها، به جای چسباندن بردارهای جاساز به‌دست‌آمده از لایه‌های مختلف، آن‌ها را با ضرایب مشخصی با هم جمع می‌کند و همچنین با استفاده از یک تابع فعال‌ساز خطی و حذف بردارهای وزن، باعث سادگی مدل و در نتیجه بهبود دقت و بازیابی و سَکه می‌شود.

۱.۵.۳.۴ تبیین روش پیشنهادی با مثال

به خاطر خطی و ساده‌تر بودن شبکهٔ پیچشی گراف سبک نسبت به الگوریتم‌های پیش از آن، روش کار آن را راحت‌تر می‌توان در قالب یک مثال بیان کرد. پروژهٔ ContainX/openstack4j از ۳۱ کتابخانه و پروژهٔ google/caliper از ۲۲ کتابخانه استفاده می‌کنند که سه تا از آن‌ها به نام‌های زیر با هم اشتراک دارند و در کل پنجاه کتابخانه دارند:

- com.google.guava:guava

[1] over-smoothing

- com.google.code.findbugs:jsr305
- junit:junit

اگر کل دادگان ورودی مسئله شامل همین دو پروژه و کل کتابخانه‌ها شامل همین پنجاه تا باشد ماتریس مجاورت ورودی مسئله حاول ۵۲ سطر و ستون خواهد بود و هر سطر آن بردار جاساز یک پروژه یا کتابخانه است.

برای پیشنهاد کتابخانه‌های بیشتر به هر کدام از این پروژه‌ها، فرض کنید از این دادگان به عنوان ورودی یک شبکهٔ پیچشی گراف سبک با دو لایه استفاده شود. در لایهٔ اول یادگیری عمیق به فقط همسایه‌های مستقیم هر راس توجه می‌شود که برای پروژه‌ها، همسایه‌ها در این لایه کتابخانه‌های مورد استفادهٔ آن‌ها و برای کتابخانه‌ها، پروژه‌هایی که از آن استفاده کردند، است. در لایهٔ دوم یادگیری عمیق به روابط مرتبهٔ دوم راس‌های گراف توجه می‌شود که برای هر کتابخانه، پروژه‌های مشترکی است که از آن استفاده کردند و برای هر پروژه کتابخانه‌هایی است که به طور مشترک در آن استفاده شدند. با بازنویسی رابطهٔ ۲.۲ از زاویهٔ دید یک پروژه و سپس جایگزین کردن بردار جاساز لایهٔ یک با رابطهٔ محاسبهٔ آن برحسب بردار لایهٔ صفر یا ورودی مسئله در نهایت ۶.۴ به دست می‌آید [۵].

$$\mathbf{e}_p^{(2)} = \sum_{i \in \mathcal{N}_p} \frac{1}{\sqrt{|\mathcal{N}_p|}\sqrt{|\mathcal{N}_l|}} \mathbf{e}_l^{(1)} = \sum_{i \in \mathcal{N}_p} \frac{1}{|\mathcal{N}_i|} \sum_{v \in \mathcal{N}_l} \frac{1}{\sqrt{|\mathcal{N}_p|}\sqrt{|\mathcal{N}_v|}} \mathbf{e}_v^{(0)} \qquad (۶.۴)$$

اگر در رابطهٔ بالا پروژهٔ p را ContainX/openstack4j در نظر بگیریم l کتابخانه‌های استفاده شده در آن و v پروژه‌ای که در لایهٔ دوم از کتابخانه‌های مشترکی با p استفاده کرده یعنی همان google/caliper خواهد شد. در نهایت بردار جاساز لایهٔ دوم که حاصل در نظر گرفتن روابط مرتبهٔ دوم بین یک پروژه با پروژهٔ دیگر است با بردار جاساز همان پروژه در لایهٔ قبل که حاصل در نظر گرفتن روابط مستقیم آن پروژه با همسایگانش یعنی کتابخانه‌ها است با وزن‌های مشخصی جمع می‌شود. وزن لایهٔ اول یک تقسیم بر دو و وزن لایهٔ دوم یک تقسیم بر سه است به این ترتیب تاثیر روابط مرتبه‌های بالاتر ضعیف‌تر می‌شود. در رابطهٔ ۶.۴ هر کدام از مقادیر N برابر تعداد همسایه‌های مستقیم آن راس در گراف است یعنی مثلا N_p با عدد ۳۱ جاگذای می‌شود. در نهایت پس از ترکیب لایه‌ها با هم، برای پیشبینی امتیاز یک پروژه به یک کتابخانه باید بردار جاساز نهایی آن پروژه را در بردار جاساز نهایی آن کتابخانه ضرب داخلی کرد.

۶.۳.۴ تلفیق با سامانهٔ توصیه‌گر مبتنی‌بر محتوا

در کنار روش پیشنهادشده مبتنی بر یادگیری عمیق که در بخش قبل معرفی شد، یک روش مبتنی بر محتوا به نام ContentLibAide نیز پیشنهاد شده‌است. روش کار این سامانه برای ارائهٔ فهرست پیشنهاد به یک پروژهٔ جدید به این‌صورت است که بر اساس برچسب‌های استخراج‌شده برای پروژه‌ها در بخش ۴.۱.۴ میزان شباهت کسینوسی تمام آن‌ها را با پروژهٔ جدید محاسبه می‌کند. سپس برای محاسبهٔ پیش‌بینی وضعیت یک پروژه در قبال یک کتابخانه عدد شباهت تمام پروژه‌هایی که آن کتابخانه را استفاده کردند را با هم جمع می‌کند. تعداد ده تا از کتابخانه‌هایی که مقدار پیش‌بینی‌شده برایشان از همه بیشتر باشد را به صورت مرتب‌شده در نهایت به پروژهٔ مورد نظر پیشنهاد می‌کند.

در نهایت فهرست پیشنهادهای به دست آمده از روش مبتنی بر یادگیری عمیق و روش مبتنی‌بر محتوا به‌صورت نوبت‌گردشی[1] با هم تلفیق[2] می‌شوند. تلفیق روش یادگیری عمیق که در واقع مبتنی بر پالایش همکارانه کار می‌کند با یک روش مبتنی برمحتوا باعث حل شدن مشکل شروع سرد می‌شود. هرچند که روش ContentLibAide از نظر دقت و بازیابی پیشنهادها عملکرد قابل رقابتی با روش DeepLibAide ندارد ولی تلفیق این دو روش از این نظر مفید است که اگر پروژهٔ جدیدی به سامانه مراجعه کند که قبلاً از کتابخانه‌ای استفاده نکرده یا از تعداد کتابخانه‌های کمی استفاده کرده روش مبتنی بر یادگیری عمیق امکان ارائهٔ پیشنهاد به وی را ندارد ولی با داشتن برچسب‌های آن پروژه می‌توان از روش مبتنی بر محتوا برای ارائهٔ پیشنهادها بهره جست.

۴.۴ جمع‌بندی روش پیشنهادی

در شکل ۳.۴ شمای کلی معماری و مولفه‌های تشکیل‌دهندهٔ سامانهٔ توصیه‌گر در قالب نمودار استقرار نمایش داده شده‌است. برحسب این نمودار بخش قابل توجه کدهای نوشته‌شده روی کولب گوگل اجرا می‌شوند و مولفه‌های آن به سرویس بیگ‌کوئری گوگل و دیتا استودیو گوگل نیز وابستگی دارند.

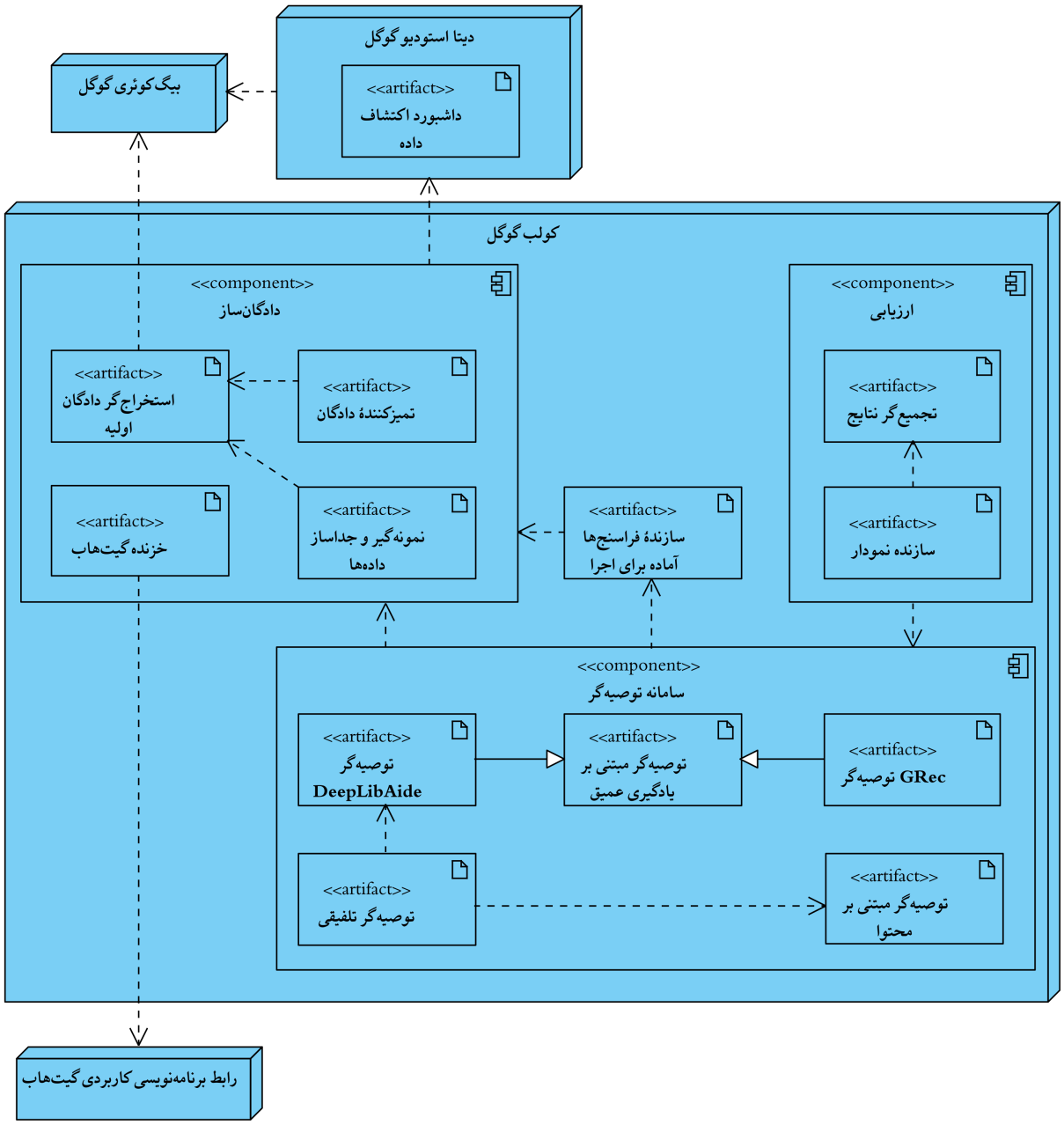
در این پژوهش با استفاده از تجمیع سه تا دادگان مختلف که با نرخ هفتگی به‌روزرسانی می‌شوند، یک دادگان از ۱۴۳۸۸۰ پروژهٔ گیت‌هاب[3] که از مدیر بسته‌های مِیوِن[4] استفاده می‌کنند و فهرست تمام کتابخانه‌های مورد استفاده در

[1] round-robin
[2] hybrid
[3] GitHub
[4] Maven

آن‌ها و فاصلهٔ بین اولین و آخرین کامیت، آمار تعداد کامیت، تعداد نویسنده، ستاره، انشعاب[1]، نویسندگان و کامیت‌ها در یک سال اخیر ساخته شده‌است. از ترکیب خطی هنجارشدهٔ آمارهای بیان‌شده، یک عدد به عنوان کیفیت هر پروژه به دست آمده‌است. یک داشبورد هم برای اکتشاف داده‌ها و اعمال فیلترها روی آن‌ها طراحی شده‌است. علاوه‌بر این، با استفاده از واسط برنامه‌نویسی کاربردی[2] گیت‌هاب تگ‌های پروژه‌ها از طریق موضوعات[3] برچسب‌خورده و درصورت عدم وجود موضوعات، برچسب‌ها از طریق پردازش متن توضیحات پروژه‌ها استخراج شدند.

سپس با دو سامانهٔ توصیه‌گر از داده‌های تجربی[4] پروژه‌های نرم‌افزاری زبان برنامه‌نویسی جاوا[5]، برای پیشنهاد فهرست مرتب شدهٔ کتابخانه‌ها[6] به عنوان تصمیم معماری نرم‌افزار[7] در زمینهٔ انتخاب فناوری استفاده شده‌است. سامانهٔ اول با استفاده از یادگیری عمیق[8] یک مدل شبکهٔ پیچشی گراف سبک[9] را آموزش می‌دهد و سامانهٔ دوم که نتایجش در یک توصیه‌گر تلفیقی[10] با روش اول ترکیب می‌شود، یک سامانهٔ توصیه‌گر مبتنی‌بر محتوا است. همچنین مدل‌های آموزش داده شده به‌طور جداگانه برای انتخاب کتابخانه توسط معمار نرم‌افزار در جریان ایجاد یا تکامل نرم‌افزار[11] می‌توانند مورد استفاده قرار گیرند.

[1] fork
[2] api
[3] topic
[4] empirical
[5] Java
[6] libraries
[7] software architecture
[8] deep learning
[9] Light Graph Convolutional Network
[10] hybrid recommender
[11] software evolution



شکل ۳.۴: شمای کلی معماری و مولفه‌های تشکیل‌دهندهٔ سامانهٔ توصیه‌گر در قالب نمودار استقرار

فصل ۵

ارزیابی

در این بخش ابتدا آمارهایی دربارهٔ سه دادگان اصلی استخراج شده بیان شده‌است. سپس معیارهای ارزیابی معرفی می‌شوند. توضیحاتی دربارهٔ پژوهش پایه اصلی ارائه می‌شود. سپس جزئیاتی از پیاده‌سازی مطرح شده‌است. در انتها نتایج ارزیابی در حالت‌های مختلف رسم و تفسیر شدند.

۱.۵ دادگان مورد استفاده

دادگان استخراج شده از داده‌های خام گیت‌هاب برای پروژه‌های میون و کتابخانه‌های آن‌ها حاوی ۱۴۳۸۸۰ پروژه است. از این تعداد برای ۱۷ درصد پروژه‌ها داده‌ای در خصوص کیفیت آن‌ها موجود نیست و این پروژه‌ها در مرحلهٔ بعد حذف شدند. پس از حذف پروژه‌های فاقد دادهٔ کیفیت و کتابخانه‌های اشتباه، این پروژه‌ها جمعاً از ۱۶۱۲۲۷ عدد کتابخانه استفاده می‌کنند.

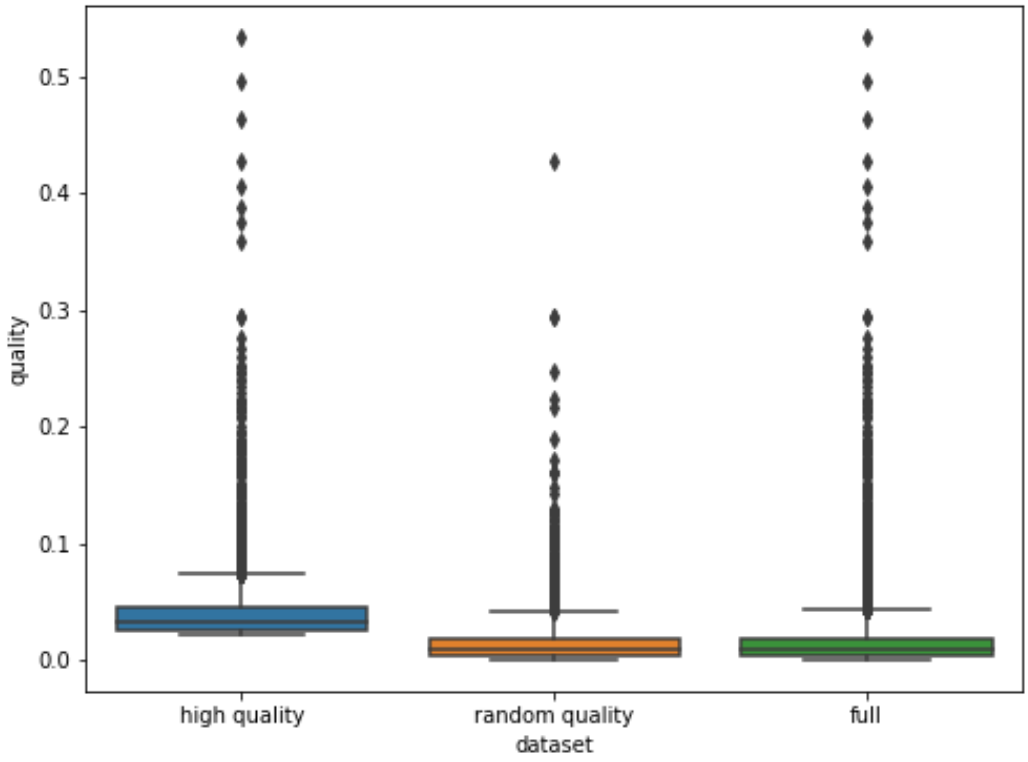
از روی این تعداد پروژه سه نمونه ساخته شده‌است. جزئیات مراحل ساخت نمونه‌ها در بخش ۲.۳.۴ بیان شده‌است. مطابق جدول ۱.۵ نمونهٔ اول که کامل‌ترین نمونه است حاوی ۳۱۴۰۰ پروژه است. نمونهٔ دوم حاوی بیست درصد پروژه‌های نمونهٔ اول است که بیشترین کیفیت را دارند بنابراین نمونهٔ دوم حاوی ۶۲۸۰ پروژه است و نمونهٔ سوم هم حاوی همان تعداد پروژه است با این تفاوت که پروژه‌های نمونهٔ سوم به‌صورت تصادفی انتخاب شدند.

جدول ۱.۵: آمارهایی دربارهٔ دادگان‌های آزمایشی

دادگان	تعداد پروژه‌ها	تعداد کتابخانه‌ها	تعداد استفاده از کتابخانه‌ها در پروژه‌ها	میزان تراکم
full	۳۱۴۰۰	۸۸۰۲۳	۱۰۳۴۴۳۳	۰/۰۰۰۳۷
high_quality	۶۲۸۰	۳۴۴۵۱	۳۵۵۵۸۷	۰/۰۰۱۶
random_quality	۶۲۸۰	۳۴۷۰۷	۲۰۳۸۷۴	۰/۰۰۰۹۳

همان‌طور که در جدول ۱.۵ مشخص است چگالی دادگان با کیفیت بالا از دو دادگان دیگر بیشتر است ولی هر سه دادگان ماهیتا تُنک هستند.

در شکل ۱.۵ توزیع کیفی پروژه‌ها در هر کدام از نمونه‌ها در قالب نمودار جعبه‌ای نمایش داده شده‌است. همان‌طور که از شکل پیداست بخش قابل توجهی از پروژه‌ها در سه دادگان موجود از امتیاز کیفی کمی برخوردارند ولی پروژه‌های



شکل ۱.۵: توزیع کیفی پروژه‌ها در هر کدام از نمونه‌ها در قالب نمودار جعبه‌ای

دادگان با کیفیت امتیازی بیشتری دارد.

۲.۵ معیارهای ارزیابی

معیارهایی که در این پژوهش برای ارزیابی نتایج استفاده شده‌است معیارهای کمی و قابل محاسبهٔ رایج در پژوهش‌ها برای ارزیابی کارایی سامانه‌های پشتیبان تصمیم هستند. از آن‌جایی که خروجی این پژوهش فهرستی از فناوری‌های پیشنهادی برای یک پروژه است، برای ارزیابی آن، دادهٔ موجود به دو قسمت دادهٔ آموزشی و دادهٔ آزمون تقسیم شده‌است و برای ارزیابی پیشنهادهای خروجی سامانه، با کتابخانه‌های داخل فایل آزمون مقایسه می‌شود.

۱. **دقت**[1] در این‌جا دقت به معنای نسبت فناوری‌هایی که به درستی پیشنهادشده به کل فناوری‌های پیشنهادشده است. این معیار با فرمول ۱.۵ محاسبه می‌شود.

[1] precision

$$Precision = \frac{\text{Positive True}}{\text{Positive True} + \text{Positive False}} \quad (۱.۵)$$

۲. **بازیابی**[1] در این‌جا بازیابی به معنای فناوری‌هایی که به درستی پیشنهادشده به کل فناوری‌هایی که باید پیشنهاد می‌شد است. فرمول این معیار در زیر مشخص شده‌است. این معیار با فرمول ۲.۵ محاسبه می‌شود.

$$Recall = \frac{\text{Positive True}}{\text{Positive True} + \text{Negative False}} \quad (۲.۵)$$

۳. **سود انباشتهٔ کاهش‌یافته هنجارشده**[2] **(سَکه)** نشان می‌دهد تا چه اندازه پیشنهادهای درست در بالای فهرست پیشنهادها هستند. روش محاسبهٔ سَکه به این‌صورت است که برای هر پیشنهاد درست یک تقسیم بر لگاریتم شماره جایگاه آن پیشنهاد به علاوه یک و برای هر پیشنهاد غلط عدد صفر را جمع می‌کنیم و در نهایت حاصل جمع به دست‌آمده را تقسیم به مجموع لگاریتم‌های جایگاه‌ها به علاوه یک می‌کنیم تا به عددی در بازهٔ صفر و یک برسیم. این عدد هر چقدر به یک نزدیک‌تر باشد بهتر است چون به این معنی است که پیشنهادهای درست بیشتر در بالای فهرست پیشنهادها نمایش داده شده‌اند.

۳.۵ پیاده‌سازی

در پیاده‌سازی این پژوهش از ابزارهای زیر استفاده شده‌است:

- TensorFlow یک کتابخانهٔ زبان پایتون برای یادگیری ماشین و هوش مصنوعی است که بیشتر برای آموزش و آزمون شبکه‌های عصبی مورد استفاده قرار می‌گیرد.
- سایر ابزارها و کتابخانه‌های زیست‌بوم[3] زبان پایتون در زمینهٔ هوش مصنوعی مانند دفتر جوپیتر[4] و پانداس[5] و نامپای[6] و کتابخانه‌های پردازش زبان طبیعی مانند ان‌ال‌تی‌کی

[1] recall
[2] Normalized Discounted Cumulative Gain
[3] ecosystem
[4] jupyter notebook
[5] pandas
[6] numpy

- Google Colab Pro+ خدمات گوگل برای اجرای برنامه‌های با پردازش سنگین گرافیکی
- Google BigQuery خدمات گوگل برای اجرای پرسمان روی حجم دادهٔ بسیار بزرگ
- Google Data Studio خدمات گوگل برای کنکاش در داده‌ها و ساخت داشبوردهای هوش تجاری

۴.۵ فراسنج‌های مؤثر در اجرا

فراسنج‌های مؤثر در اجرای برنامه در ادامه توضیح داده شده‌اند و مقادیر در نظر گرفته‌شده برای مقایسهٔ نتایج ارزیابی روش پیشنهادی با روش پایه مشخص شده‌است.

- **بهینه‌ساز:** از بهینه‌ساز ادم[۱] استفاده شده‌است. همچنین با بهینه‌ساز گرادیان کاهشی[۲] مقدار بهینه برای فراسنج ضریب تنظیم به دست آمده‌است.
- **نحوه ساخت ماتریس مجاورت:** روش‌های مختلف بهنجارش هنگام ساخت ماتریس مجاورت وجود دارد. در این روش از یک ماتریس مجاورت هنجارشده استفاده شده‌است.
- **ضریب تنظیم:** ضریب تنظیم[۳] یکی از فراسنج‌های یادگیری عمیق است که برای حل مسئلهٔ بیش‌برازش[۴] استفاده می‌شود. ضریب تنظیم برای چهار حالت مختلف در روش پیشنهادی و روش پایه بررسی و در نمودارهای بعدی مقایسه شده‌است.
- **توقف زودهنگام[۵]:** روش پیشنهادی و روش پایه اگر در چند دورهٔ آخر هیچ بهبودی پیدا نکنند به توقف زودهنگام می‌خورند و اجرای برنامه را ادامه نمی‌دهند و برای همین در این موارد نمودار هر دو روش برای دوره‌های بالاتر از یک عددی رسم نشده‌است.
- **دوره[۶]:** ارزیابی روش پیشنهادی نشان می‌دهد که مدل پیشنهادشده حتی برای دوره[۷]های بسیار بالاتر از ۵۰ هم کارایی بهتری دارد ولی به خاطر منابع محدود پردازشی تا ۵۰ دوره نمودارها رسم شدند.

[1] Adam optimizer
[2] gradient descent
[3] Regularization Coefficent
[4] over-fitting
[5] early stopping
[6] epoch
[7] Epoch

- **تعداد پیشنهاد:** تمام ارزیابی‌ها برای از یک پیشنهاد تا ده پیشنهاد اول به‌صورت جداگانه انجام شده‌است. برای تمام پیشنهادها وضعیت روش پیشنهادی نسبت به روش پایه بهتر است.

- **اندازهٔ بردار جاساز:** اندازهٔ بردار جاساز[1] به‌طور یکسان در تمام لایه‌ها و مساوی اندازهٔ لایه گرفته‌شده‌است چون روش پیشنهادی از تبدیل ویژگی[2] استفاده نمی‌کند و در نهایت برای رسیدن به لیست جاسازی لایهٔ آخر، تمام بردارهای جاساز لایه‌های مختلف را با ضرایب مختلفی با هم جمع می‌کند. اندازهٔ بردار جاساز برابر ۳۲ در نظر گرفته شده‌است.

- **اندازهٔ لایه:** اندازهٔ لایه[3] در شرایط کاملاً مساوی با پژوهش پایه مقایسه شده‌است. اندازهٔ هر لایه ۳۲ است.

- **حذف تصادفی پیام:** حذف تصادفی پیام[4] در شرایط کاملاً مساوی با پژوهش پایه مقایسه شده‌است. با توجه به این‌که در مدل‌های خطی، حذف تصادفی پیام معادل ضریب تنظیم مرتبه اول L2 عمل می‌کند، و این‌که از ضریب تنظیم برای جلوگیری از بیش‌برازش[5] استفاده شده، این فراسنج برابر صفر در نظر گرفته شده‌است [۴۶] [۵].

- **نرخ یادگیری:** نرخ یادگیری[6] به خاطر استفاده از بهینه‌ساز adam به صورت خودکار تنظیم می‌شود ولی برای شروع مقدار آن مطابق مقدار رایج در آموزش مدل‌های یادگیری عمیق برابر ۰/۰۱ در نظر گرفته شده‌است.

- **اندازهٔ دسته:** اندازهٔ دسته[7] برابر تعداد نمونه‌ها یا پروژه‌هایی است که به عنوان ورودی به مدل داده می‌شود. تعداد کل نمونه‌ها تقسیم بر اندازهٔ دسته برابر تعداد دسته‌هایی است که در هر دوره به مدل داده می‌شود. هدف از تقسیم نمونه‌ها به دسته‌های کوچک‌تر مصرف حافظهٔ اصلی کمتر است. در این پژوهش اندازهٔ دسته‌ها برابر نصف تعداد نمونه‌ها در دادگان تصادفی و با کیفیت بالا در نظر گرفته شده‌است.

[1] embedding vector size
[2] feature transformation
[3] Layer Size
[4] Message Dropout
[5] over-fitting
[6] Learning Rate (lr)
[7] Batch Size

۵.۵ نتایج ارزیابی

در گزارش‌های موجود در این فصل میانگین نتایج حاصل از هر کدام از پنج افراز بیان شده‌است و نتایج برای هر کدام از پنج افراز به‌طور مجزا ذکر نشده‌است. همان‌طور که در بخش ۲.۳.۴ گفته شد از سه روش رایج در پژوهش‌های پیشین برای ارزیابی متقابل استفاده شده‌است. همان‌طور که جدول ۲.۵ نشان می‌دهد انحراف معیار نتایج ارزیابی پنج افراز برای هر سه روش ارزیابی متقابل بسیار جزئی است که نشان می‌دهد مقادیر میانگین نتایج ارزیابی میان افرازها برای مقایسهٔ دو روش قابل استناد است.

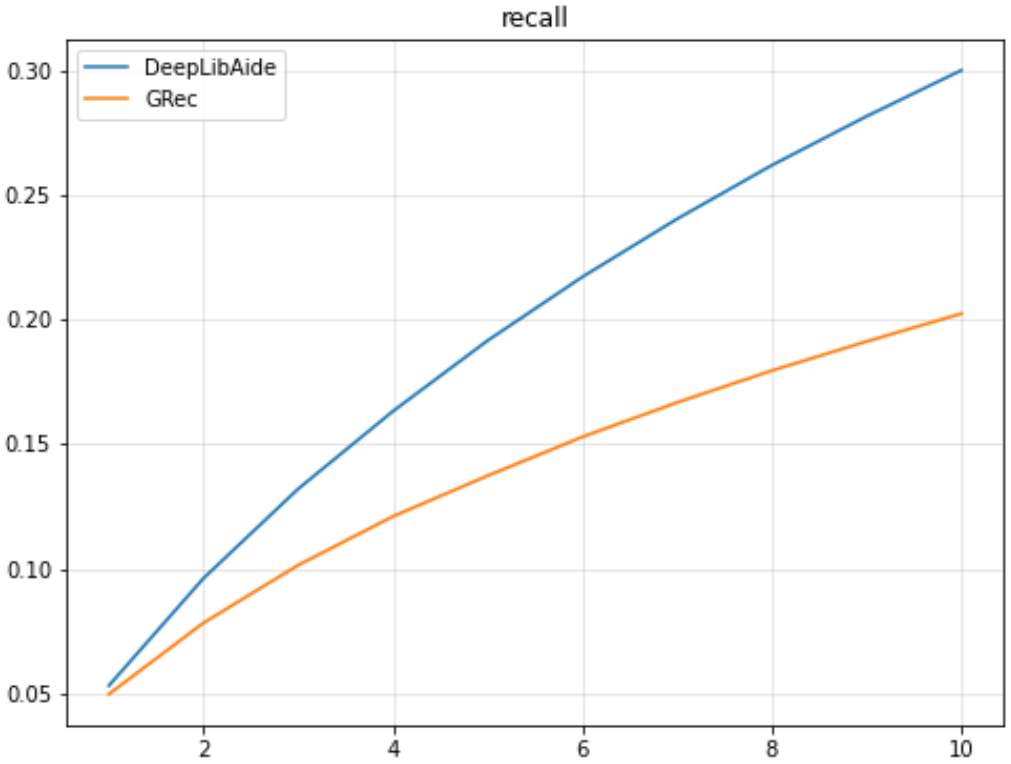
جدول ۲.۵: انحراف معیار نتایج ارزیابی در افرازها

دادگان	تعداد کتابخانه‌های داخل آزمون	سَکه ده پیشنهاد اول	دقت ده پیشنهاد اول	بازیابی ده پیشنهاد اول
high_quality	۱	۰/۰۰۰۰۱۲	۸/۶۵۸۴۹۴e-۰۶	۰/۰۰۰۰۰۲
high_quality	۵	۰/۰۰۰۰۱۲	۲/۵۲۲۵۸۰e-۰۵	۰/۰۰۰۰۰۹
high_quality	۲۰%	۰/۰۰۰۰۱۴	۳/۶۵۰۲۳۱e-۰۵	۰/۰۰۰۰۱۴
random_quality	۱	۰/۰۰۰۰۱۲	۳/۷۶۶۲۳۵e-۰۶	۰/۰۰۰۰۰۲
random_quality	۵	۰/۰۰۰۰۲۵	۹/۳۸۱۴۰۹e-۰۷	۰/۰۰۰۰۲۱
random_quality	۲۰%	۰/۰۰۰۰۶۳	۸/۶۴۴۲۰۹e-۰۵	۰/۰۰۰۰۱۵

روش جداسازی تقسیم زمانی مبتنی‌بر کاربر نسبت به دو روش دیگر که در جدول با اعداد ۱ و ۵ مشخص شده انحراف معیار بیشتری دارد که شرح دلایل آن در ۲.۳.۴ بیان شده‌است.

۱.۵.۵ مقایسهٔ میانگین دو روش به‌طور کلی

در شکل ۲.۵ میانگین میزان اختلاف بهبود بازیابی در روش پیشنهادی نسبت به روش پایه برحسب جایگاه پیشنهاد مورد بررسی قرار گرفته‌است. همان‌طور که از شکل پیداست در تعداد پیشنهادهای بیشتر اختلاف بازیابی بهبود بیشتری نسبت به روش پایه پیدا می‌کند.



شکل ۲.۵: مقایسهٔ میانگین میزان اختلاف بهبود بازیابی در روش پیشنهادی نسبت به روش پایه برحسب جایگاه پیشنهاد

روش پیشنهادی با روش پایه از نظر معیارهای دقت، بازیابی و سَکه مقایسه شدند. روش پیشنهادی در این پژوهش با اختلاف قابل توجهی عملکرد بهتری نسبت به روش پایه نشان داده‌است. همان‌طور که در جدول ۳.۵ مشخص است بازیابی، سَکه و دقت روش پیشنهادی روی هر سه دادگان کامل، با کیفیت بالا و با کیفیت تصادفی نسبت به روش پایه بهبود داشته‌است.

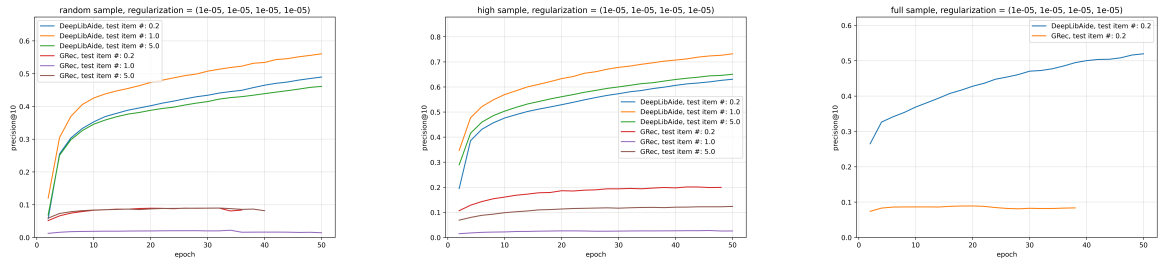
در جدول ۳.۵ روش پیشنهادی در این پژوهش که LibDeepAide نام دارد با روش پایه روی سه دادگان موجود

جدول ۳.۵: مقایسهٔ بازیابی، سَکه و دقت بین روش پیشنهادی و پایه

دادگان	نام روش	بازیابی پنج پیشنهاد اول	بازیابی ده پیشنهاد اول	سَکه پنج پیشنهاد اول	سَکه ده پیشنهاد اول	دقت پنج پیشنهاد اول	دقت ده پیشنهاد اول	دوره توقف زودهنگام
full	DeepLibAide	۰/۱۵۸۳	۰/۲۴۴۹	۰/۴۸۱۱	۰/۴۴۳۳	۰/۴۵۱۹	۰/۳۷۴۸	۱۲
high_quality	DeepLibAide	۰/۱۶۷۶	۰/۲۷۹۳	۰/۷۱۲۵	۰/۶۸۱۴	۰/۶۹۳۲	۰/۶۳۰۷	۵۰
random_quality	DeepLibAide	۰/۱۹۹۷	۰/۳۱۳۵	۰/۶۱۲۴	۰/۵۶۶۵	۰/۵۷۹۰	۰/۴۸۹۷	۵۰
full	GRec	۰/۱۳۲۹	۰/۱۹۲۳	۰/۲۶۰۸	۰/۳۲۴۳	۰/۱۰۸۲	۰/۰۸۳۸	۳۸
high_quality	GRec	۰/۱۵۴۸	۰/۲۳۳۳	۰/۴۰۴۰	۰/۴۹۷۹	۰/۲۴۴۶	۰/۱۹۹۸	۴۸
random_quality	GRec	۰/۱۳۲۳	۰/۱۹۲۷	۰/۲۶۱۵	۰/۳۲۳۶	۰/۱۰۷۷	۰/۰۸۳۷	۳۶

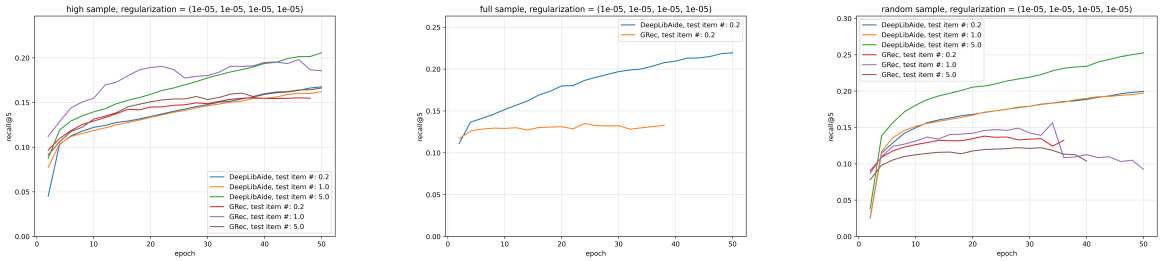
شامل دادگان کامل، دادگان باکیفیت و دادگان با کیفیت تصادفی از نظر بازیابی، سَکه و دقت مقایسه شده‌است. همچنین مشخص شده که هر کدام از الگوریتم‌ها هنگام اجرای روی هر یک از دادگان در کدام دوره به توقف زودهنگام برخورد کرده‌اند. به‌طور مثال الگوریتم DeepLibAide در معیار بازیابی در بازهٔ دو درصد تا حداکثر ده درصد هم برای پنج پیشنهاد اول و هم برای ده پیشنهاد اول نسبت به روش پایه بهبود داشته‌است. همچنین بازیابی روش DeepLibAide همانند روش پایه در تمام دادگان‌ها، برای ده پیشنهاد اول نسبت به پنج پیشنهاد اول بهبود نه تا یازده درصدی داشته‌است که این نتیجه با انتظارمان مطابقت دارد زیرا افزایش تعداد پیشنهادها باعث بهبود بازیابی می‌شود ولی روی دقت تأثیر عکس دارد.

در روش جی‌رک اختلاف نتایج ارزیابی برای دادگان کامل و دادگان تصادفی که اندازهٔ بسیار کوچک‌تری از دادگان کامل دارد تقریباً یکسان است ولی روش پیشنهادی برای دادگان تصادفی که اندازهٔ کوچک‌تری دارد با اختلاف قابل توجهی بهتر از دادگان کامل عمل کرده‌است. در هر سه معیار ارزیابی دادگان باکیفیت نسبت به دادگان تصادفی و دادگان کامل بهبود یافته‌است فقط در مورد معیار بازیابی و در روش پیشنهادی به مقداری جزئی دادگان با کیفیت نسبت به دو دادگان دیگر افت بازیابی را نشان می‌دهد.



(آ) برای نمونهٔ کامل (ب) برای نمونه پروژه‌های با کیفیت (ج) برای نمونه پروژه‌های تصادفی

شکل ۳.۵: مقایسهٔ دقت برحسب دوره برای ده پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر



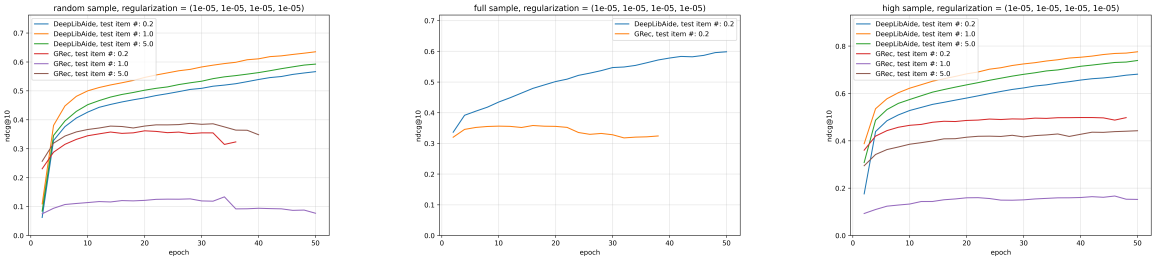
(آ) برای نمونه پروژه‌های تصادفی (ب) برای نمونهٔ کامل (ج) برای نمونه پروژه‌های با کیفیت

شکل ۴.۵: مقایسهٔ بازیابی برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر

۲.۵.۵ مقایسهٔ نتایج ارزیابی در طول روند یادگیری

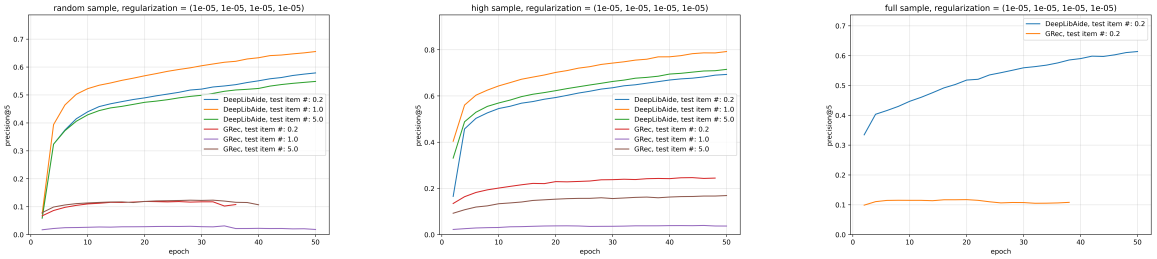
همان‌طور که در شکل ۳.۵ و ۴.۵ و ۵.۵ و ۶.۵ و ۸.۵ و ۷.۵ نشان داده شده‌است دقت و بازیابی با افزایش دوره و گذشت روند یادگیری بهبود پیدا می‌کند ولی برای دوره‌های بالاتر از حد مشخصی نتایج دچار کاهش می‌شود. در برخی موارد یکی از الگوریتم‌ها دچار توقف زودهنگام می‌شود و برای همین یادگیری در آن‌جا متوقف می‌شود. مقایسهٔ دقت و بازیابی برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر در شکل‌ها دیده می‌شود.

در شکل۴.۵ج الگوریتم جی‌رک در شروع کار وضعیت بهتری در معیار بازیابی برای پنج پیشنهاد اول نسبت به الگوریتم پیشنهادی دارد ولی در ادامه نوسان زیادی داشته و در نهایت در آخرین دوره پایین‌تر از الگوریتم پیشنهادی قرار می‌گیرد. به عبارت دیگر روش پیشنهادی به‌طور فرایندهای رشد پیوسته‌ای را تجربه کرده‌است.



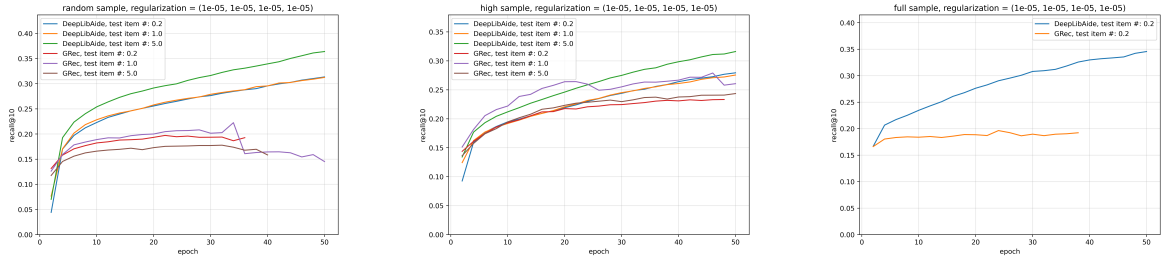
(آ) برای نمونه پروژه‌های با کیفیت **(ب)** برای نمونهٔ کامل **(ج)** برای نمونه پروژه‌های تصادفی

شکل ۵.۵: مقایسهٔ سَکه برحسب دوره برای ده پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر



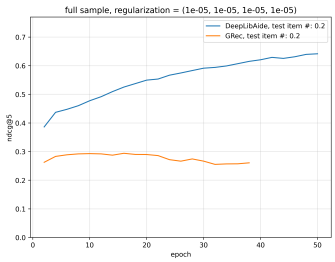
(آ) برای نمونهٔ کامل **(ب)** برای نمونه پروژه‌های با کیفیت **(ج)** برای نمونه پروژه‌های تصادفی

شکل ۶.۵: مقایسهٔ دقت برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر

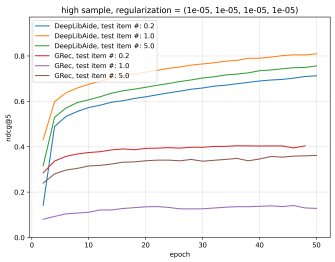


(آ) برای نمونهٔ کامل **(ب)** برای نمونه پروژه‌های با کیفیت **(ج)** برای نمونه پروژه‌های تصادفی

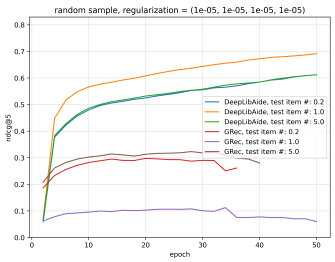
شکل ۷.۵: مقایسهٔ بازیابی برحسب دوره برای ده پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر



(آ) برای نمونهٔ کامل



(ب) برای نمونه پروژه‌های با کیفیت



(ج) برای نمونه پروژه‌های تصادفی

شکل ۸.۵: مقایسهٔ سَکه برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر

۳.۵.۵ ارزیابی به تفکیک تعداد دادهٔ آزمون

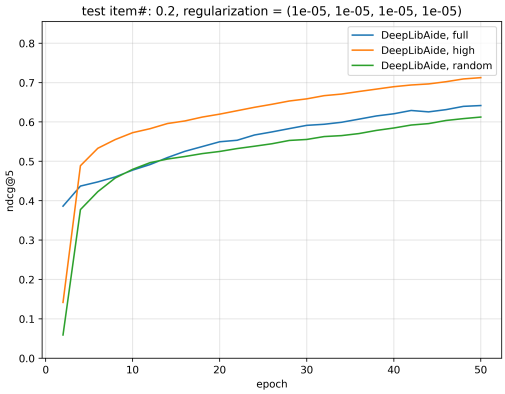
در جدول ۴.۵ نتایج ارزیابی به تفکیک تعداد کتابخانه در دادهٔ آزمون برای هر پروژه مشخص شده‌است. در این جدول میانگین عملکرد روش پیشنهادی و روش پایه مشخص شده‌است. در این شکل مشاهده می‌شود که روش DeepLibAide تحت همهٔ شرایط مختلف دادگان و روش‌های مختلف تقسیم‌بندی افرازها عملکرد بهتری نسبت به روش پایه از خود نشان می‌دهد. برای نمونه پروژه‌های تصادفی بازیابی در الگوریتم جی‌رک حالت پایداری ندارد ولی در روش پیشنهادی رشد پایداری با افزایش دوره ملاحظه می‌شود.

۴.۵.۵ تاثیر کیفیت و حجم دادهٔ موجود در دادگان آزمون روی نتایج ارزیابی

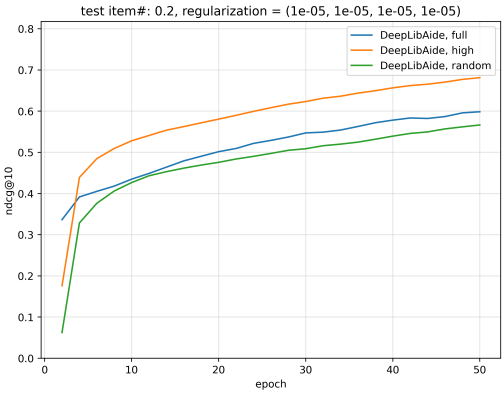
علاوه بر اختلاف قابل توجه کارایی مدل طراحی شده در تمام حالت‌ها نسبت به روش پایه، این مدل برای دادهٔ پروژه‌های باکیفیت‌تر گیت‌هاب نیز به‌طور جداگانه آموزش داده شده‌است و مقایسهٔ نتایج شکل‌های ۹.۵ و ۱۰.۵ و ۱۱.۵ نشان می‌دهد که مدل به ترتیب برای دادگان‌های بسیار بزرگ (حتی با وجود پروژه‌های کم‌کیفیت) از همه بهتر و سپس داده‌های پروژه‌های با کیفیت و در آخر پروژه‌های کم‌کیفیت کارایی بهتری نشان می‌دهد.

جدول ۴.۵: دقت، بازیابی و سَکه برای روش پیشنهادی برای تعداد کتابخانه‌های داخل آزمون مختلف

تعداد ک.	دادگان	نام روش	تعداد کتابخانه پیشنهادی ۵			تعداد کتابخانه پیشنهادی ۱۰		
			دقت	بازیابی	سَکه	دقت	بازیابی	سَکه
۱	high_quality	DeepLibAide	۰/۷۹۲	۰/۱۶۲	۰/۸۰۹	۰/۷۳۲	۰/۲۷۶	۰/۷۷۶
		GRec	۰/۰۳۷	۰/۱۸۶	۰/۱۲۸	۰/۰۲۶	۰/۲۶۱	۰/۱۵۲
	random_quality	DeepLibAide	۰/۶۵۶	۰/۱۹۷	۰/۶۹۱	۰/۵۶۱	۰/۳۱۳	۰/۶۳۵
		GRec	۰/۰۱۸	۰/۰۹۲	۰/۰۶۰	۰/۰۱۵	۰/۱۴۵	۰/۰۷۷
۵	high_quality	DeepLibAide	۰/۷۱۵	۰/۲۰۶	۰/۷۵۶	۰/۶۵۱	۰/۳۱۶	۰/۷۳۹
		GRec	۰/۱۶۹	۰/۱۶۶	۰/۳۶۲	۰/۱۲۴	۰/۲۴۴	۰/۴۴۳
	random_quality	DeepLibAide	۰/۵۴۹	۰/۲۵۳	۰/۶۱۲	۰/۴۶۱	۰/۳۶۴	۰/۵۹۲
		GRec	۰/۱۰۷	۰/۱۰۴	۰/۲۸۰	۰/۰۸۲	۰/۱۵۸	۰/۳۴۸
۲۰%	full	DeepLibAide	۰/۴۵۲	۰/۱۵۸	۰/۴۸۱	۰/۳۷۵	۰/۲۴۵	۰/۴۴۳
		GRec	۰/۱۰۸	۰/۱۳۳	۰/۲۶۱	۰/۰۸۴	۰/۱۹۲	۰/۳۲۴
	high_quality	DeepLibAide	۰/۶۹۰	۰/۱۶۸	۰/۷۱۰	۰/۶۲۸	۰/۲۸۰	۰/۶۷۹
		GRec	۰/۲۴۵	۰/۱۵۵	۰/۴۰۴	۰/۲۰۰	۰/۲۳۳	۰/۴۹۸
	random_quality	DeepLibAide	۰/۵۷۹	۰/۲۰۰	۰/۶۱۲	۰/۴۹۰	۰/۳۱۴	۰/۵۶۶
		GRec	۰/۱۰۸	۰/۱۳۲	۰/۲۶۲	۰/۰۸۴	۰/۱۹۳	۰/۳۲۴



(آ) برای پنج پیشنهاد اول



(ب) برای ده پیشنهاد اول

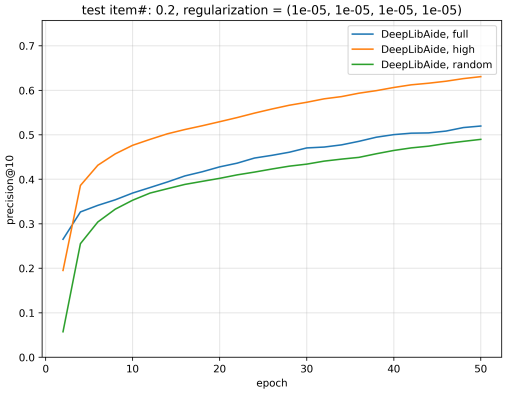
شکل ۹.۵: مقایسهٔ سَکه برحسب دوره در روش پیشنهادی شامل نمونهٔ کامل و باکیفیت و تصادفی در شرایط برابر

۵.۵.۵ ضریب تنظیم و کاهش مسئلهٔ بیش‌برازش

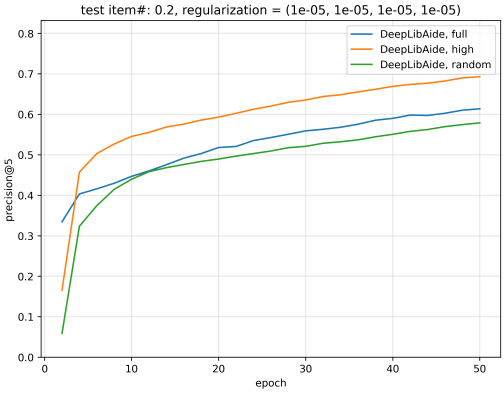
همان‌طور که در شکل‌های ۱۲.۵ دیده می‌شود افزایش ضریب تنظیم به واسطهٔ کاهش مشکل بیش‌برازش، باعث بهبود عملکرد هر دو الگوریتم در زمینهٔ سَکه می‌شود ولی افزایش آن به مقدار ۱۰ به توان منفی پنج باعث ساده شدن بیش از حد مدل و افت عملکرد آن می‌شود.

۶.۵.۵ روند تغییرات ضرر هنگام آموزش مدل

همان‌طور که در شکل ۱۳.۵ مشخص است روند یادگیری هر سه الگوریتم با توجه به دادهٔ اعتبارسنجی مستقل از کیفیت داده و حجم دادگان به‌طور مشابهی پیش می‌رود.

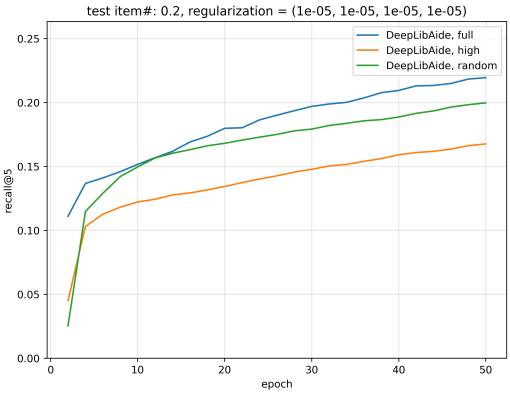


(آ) برای ده پیشنهاد اول

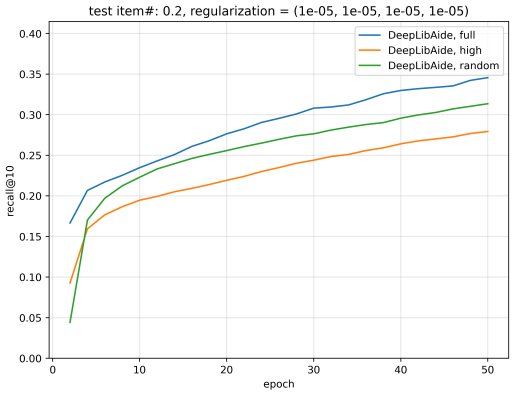


(ب) برای پنج پیشنهاد اول

شکل ۱۰.۵: مقایسهٔ دقت برحسب دوره در روش پیشنهادی شامل نمونهٔ کامل و باکیفیت و تصادفی در شرایط برابر

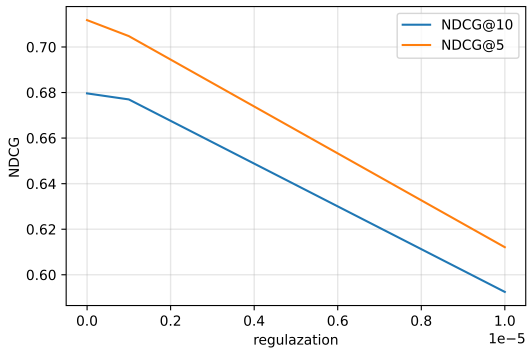


(آ) برای پنج پیشنهاد اول

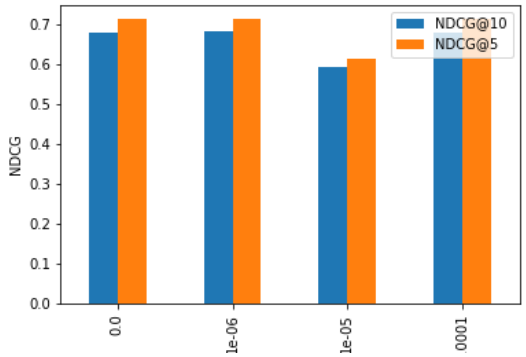


(ب) برای ده پیشنهاد اول

شکل ۱۱.۵: مقایسهٔ بازیابی برحسب دوره در روش پیشنهادی شامل نمونهٔ کامل و باکیفیت و تصادفی در شرایط برابر

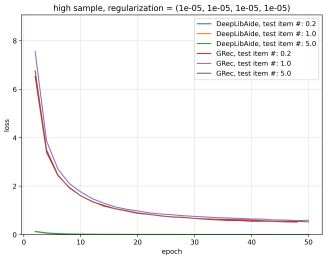


(آ) نمودار معمولی

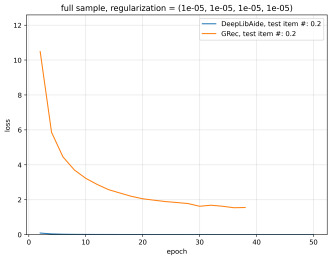


(ب) نمودار میله‌ای

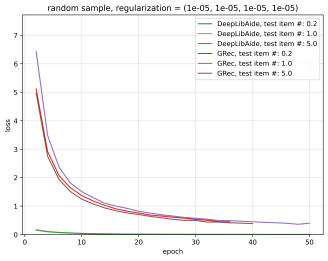
شکل ۱۲.۵: مقایسهٔ سود انباشتهٔ کاهش‌یافته هنجارشده برحسب ضریب تنظیم در روش پیشنهادی و روش پایه در شرایط برابر



(آ) برای نمونه پروژه‌های با کیفیت



(ب) برای نمونهٔ کامل



(ج) برای نمونه پروژه‌های تصادفی

شکل ۱۳.۵: مقایسهٔ ضرر برحسب دوره در روش پیشنهادی و روش پایه در شرایط برابر

فصل ۶

نتیجه‌گیری و کارهای آینده

نوآوری‌های پایان‌نامه در این بخش دسته‌بندی و جمع‌بندی و کارهای پیشنهادی آینده بیان شده‌اند.

۱.۶ جمع‌بندی

فناوری‌های پیشنهادی توسط سامانهٔ توصیه‌گر طراحی شده در این پژوهش برای پشتیبانی از تصمیم‌گیری توسط معمار نرم‌افزار می‌تواند در جریان ایجاد یا تکامل نرم‌افزار می‌تواند مورد استفاده قرار گیرد. استفاده از داده‌های تجربی، علاوه بر دقت ریاضی بیشتر، به افزایش تنوع داده‌های قابل استفاده و کاهش مشکلات ارتباطات انسانی و نمونه‌گیری هم کمک می‌کند.

نوآوری‌های این پژوهش به‌طور خلاصه شامل موارد زیر است:

- **نحوهٔ استخراج و ساخت دادگان ورودی**

– **استفاده از روشی متفاوت برای ساخت دادگان ورودی:** استفاده از الگوهای عبارات منظم در پرسمان‌های اس‌کیوال، باعث امکان اجرای کل خط‌لولهٔ استخراج و به‌روزرسانی مجموعه داده در کمتر از ۳ دقیقه بر اساس آخرین دادگان رسمی گیت‌هاب می‌شود.

– **استخراج دادهٔ مورد نیاز در توصیه‌گر مبتنی برمحتوا:** موضوعات برچسب‌خورده به پروژه‌ها از طریق رابط کاربری برنامه‌نویسی گیت‌هاب استخراج و با استفاده از پردازش زبان طبیعی روی متن توضیحات پروژه‌ها تکمیل شد.

– **محاسبهٔ یک عدد برای سابقه و کیفیت پروژه‌ها:** اطلاعاتی دربارهٔ پروژه‌ها نظیر تعداد ستاره، انشعاب، نویسنده، کامیت، نویسنده‌ها و کامیت‌های یک سال اخیر و مدت زمان عمر پروژه استخراج و سپس با ترکیب خطی آن‌ها برای هر پروژه یک عدد نهایی برای کیفیت آن‌ها به دست آمده‌است.

– **حذف موارد کتابخانه‌های اشتباه:** با توجه به الگوی عبارت منظم ایجادشده بر اساس قراردادهای[1] پروژه‌ها مِیون برای نوشتن متن شناسهٔ مصنوع و شناسهٔ سازمان، همهٔ کتابخانه‌های اشتباه استخراج‌شده از گیت‌هاب از دادگان حذف می‌شوند.

[1] conventions

- **ساخت یک داشبورد تعاملی برای اکتشاف داده‌های ورودی.**

- **الگوریتم سامانهٔ توصیه‌گر و بهبود عملکرد آن**

- **استفاده از یک الگوریتم جدیدتر مبتنی بر یادگیری عمیق:** الگوریتم مورد استفاده در توصیه‌گر DeepLibAide برای ارائهٔ فهرست پیشنهادها، به جای چسباندن بردارهای جاساز به‌دست‌آمده از لایه‌های مختلف، آن‌ها را با ضرایب مشخصی با هم جمع می‌کند که این باعث کاهش مشکل بیش‌همواری می‌شود و همچنین با استفاده از یک تابع فعال‌ساز خطی و حذف بردارهای وزن، باعث سادگی مدل و در نتیجه بهبود دقت و بازیابی و سَکه می‌شود. از این‌رو این پژوهش در زمینهٔ رویکرد حل مسئله نوآوری دارد.
- **ارائه یک سامانهٔ توصیه‌گر تلفیقی:** سامانهٔ توصیه‌گر پیشنهادی در این پژوهش نتایج توصیه‌گر مبتنی بر محتوا و توصیه‌گر مبتنی بر پالایش همکارانه را تلفیق می‌کند. تلفیق با روش مبتنی بر محتوا به واسطهٔ استفاده از یک دادگان متفاوت از دادگان استفادهٔ کتابخانه‌ها در پروژه‌ها، باعث بهبود مسئلهٔ شروع سرد می‌شود.
- **مطالعهٔ میزان همبستگی بزرگ بودن و کیفیت پروژه‌ها با کیفیت پیشنهادها:** در این پژوهش داده‌هایی از دادگان‌ها استخراج شده که دلالت بر میزان بزرگی و محبوبیت پروژه‌ها دارد؛ مثلاً تعداد ستاره‌ها یا انشعاب در گیت‌هاب بیانگر میزان محبوبیت و تعداد کامیت و مدت سابقهٔ پروژه بیان‌گر اندازهٔ پروژه است. می‌توان تأثیر این دو فاکتور را به‌طور جداگانه مورد بررسی قرار داد.

۲.۶ کارهای آینده

۱.۲.۶ استخراج دانش سطح بالا یا پایه‌واساس تصمیم‌های معماری

در این پژوهش برای استخراج ورودی‌های تصمیم‌گیری، مسیر استنتاج، برعکس مسیر عادی تصمیم‌گیری و از پایین به بالا پیموده شده‌است. به بیان دیگر از خود تصمیم‌ها، یک مدل آموزش داده شد. اما در مدل یادگیری عمیق پیشنهاد شده در این پژوهش، وزن‌های آموزش داده‌شده در مدل، معنای مشخصی ندارند و نمی‌توان طرز کار مدل

برای رسیدن به یک تصمیم مشخص را شرح داد. درحالی‌که یکی از پرسش‌های درخور بررسی برای پژوهش‌های آتی این است که چگونه و تا چه اندازه دانش ضمنی معماری مورد نیاز برای تصمیم‌گیری در زمینهٔ معماری نرم‌افزار را می‌توان از داده‌های تجربی پروژه‌های نرم‌افزاری استخراج کرد؟

۱.۱.۲.۶ تفسیرپذیری مدل

یکی از اهداف پژوهشی که به‌طور پراکنده در کارهای گذشته مورد توجه قرار گرفته استخراج دانش سطح بالای معماری نرم‌افزار مثل پایه‌واساس تصمیم‌ها بر اساس تصمیم‌های معماری موجود در پروژه‌ها است. به این هدف با دو رویکرد می‌توان پرداخت. رویکرد اول استفاده از مدل‌های یادگیری ماشین با قابلیت تفسیرپذیری[1] بیشتر به جای مدل پیشنهاد شده در این پژوهش و پژوهش‌های پایه است.

۲.۱.۲.۶ استفاده از داده‌های متنوع‌تر

رویکرد دوم برای استخراج دانش سطح بالای معماری، استفاده از داده‌های بیشتر دربارهٔ تصمیم‌ها و اطلاعاتی پیرامون تصمیم‌ها است؛ مثلاً می‌توان خود تغییر انجام‌گرفته در فناوری‌های مورد استفاده در پروژه در هر کامیت را بررسی کرد و به این ترتیب دانش سطح بالای معماری مانند «مجموعهٔ انتخاب‌های بدیل» برای هر تصمیم انتخاب فناوری را مشخص کرد زیرا براساس یک یافتهٔ تجربی که روی پروژهٔ آپاچی اسپارک[2] و آپاچی هدوپ[3] انجام گرفته، مجموعه کتابخانه‌هایی که در هر کامیت حذف و با کتابخانه‌های دیگری جایگزین شدند، با دقتی کافی به عنوان انتخاب‌هایی بدیل برای کتابخانه‌های اضافه‌شده محسوب می‌شوند. در یک نمونهٔ دیگر با بررسی متن کامیت‌ها یا متن ایراد در سامانهٔ پیگیری وظایف و اتصال آن به کامیت‌هایی که در آن‌ها کتابخانه‌ها تغییر کردند و حاشیه‌نویسی واژگان مرتبط با معماری به‌کار رفته در این متن‌ها از طریق هستان‌شناسی دی‌بی‌پدیا[4] می‌توان پایه‌واساس تصمیم‌های معماری که به حذف یا اضافه یا تغییر یک فناوری در پروژه مرتبط هستند را استخراج کرد. در یک نمونهٔ دیگر با مطالعهٔ ارتباط میان افرادی که انتخاب‌های فناوری را تغییر دادند با خود فناوری‌ها می‌توان تخصص‌های افراد و در نتیجه اعضای تیم معماری نرم‌افزار را مشخص کرد؛ بنابراین با فقط داشتن خود تصمیم‌های معماری لزوماً دانش سطح بالای سودمندی در زمینهٔ پایه‌واساس معماری قابل استخراج نیست ولی با در نظر گرفتن اطلاعات پیرامون هر

[1] interpretability
[2] Apache Spark
[3] Apache Hadoop
[4] DBPedia

تصمیم در کنار خود تصمیم‌ها، بخشی از دانش معماری، با قابلیت استفادهٔ مجدد در پروژه‌های بعدی قابل استخراج است.

۳.۱.۲.۶ بازنمایی و مدیریت دانش معماری

دانش ضمنی مرتبط با این تصمیم‌ها شامل هدف تصمیم‌گیری، انتخاب‌های بدیل، معیارهای مورد نظر برای ارزیابی هر گزینه و وضعیت هر گزینه تا حدودی از روی تصمیم‌های معماری استخراج‌شده و ارتباط آن‌ها با ویژگی‌هایی از پروژه‌های نرم‌افزار قابل بازیابی است. درصورت استخراج دانش سطح بالای معماری، بازنمایی آن در قالب یک هستان‌شناسی و در ادامه استفاده از آن برای استدلال دربارهٔ معماری نیز فعالیتی سودمند است. به عبارت دیگر دانش معماری استخراج شده می‌تواند مستقیماً به معمار در تصمیم‌گیری کمک کند یا در یکی از رویکردهای استنتاج براساس دانش خبره تصمیم‌گیری مانند سامانه‌های مدیریت دانش به عنوان ورودی الگوریتم‌های تصمیم‌گیری مورد استفاده قرار گیرد. همچنین می‌توان از دانش به‌دست‌آمده از تصمیمات پیشین در الگوریتم‌های تصمیم‌گیری برای تحلیل بده‌بستان‌ها نیز استفاده کرد.

۲.۲.۶ مطالعهٔ روندها و تغییرات در تصمیمات انتخاب فناوری

نحوهٔ تکامل و رشد پروژه‌ها و بررسی روندها در تغییر فناوری‌ها را می‌توان با بررسی سوابق تغییرات و در نظر گرفتن زمان رخداد هر تغییر در تصمیم‌های انتخاب فناوری مطالعه کرد. از این مطالعه مشخص می‌شود که مثلاً یک نسخه از کتابخانه‌های مورد استفاده در پروژه به مرور با یک نسخهٔ دیگر جایگزین شده‌است.

۳.۲.۶ روش‌های ارزیابی متنوع‌تر

در این پژوهش از داده‌های موجود تجربی و روش‌های رایج در یادگیری ماشین برای ارزیابی برون‌خطی[1] دستاوردها استفاده شده‌است ولی برای ارزیابی می‌توان از دادهٔ به‌دست آمده از یک منبع ثانویه (مثلا stackoverflow) به عنوان دادهٔ معیار استفاده و تصمیم‌های معماری پیشنهادشده را با آن مقایسه کرد. روش‌های ارزیابی برخط مانند مصاحبه[2]، آزمایش[3]، مطالعهٔ میدانی یا مشاهدهٔ مستقیم[4] و پرسشنامه[5] به ترتیب بیان‌شده در پژوهش‌های حوزهٔ

[1] offline
[2] interview
[3] experiment
[4] observation
[5] questionnaire

معماری نرم‌افزار از اهمیت بالایی برخوردار هستند و می‌توان از آن‌ها نیز برای ارزیابی سامانهٔ توصیه‌گر استفاده کرد.

۴.۲.۶ مطالعهٔ سایر انواع تصمیم‌های معماری

علاوه بر کتابخانه‌های مورد استفاده، می‌توان نسخهٔ مناسب هر کتابخانه را نیز پیشنهاد داد. همچنین با بررسی مستندات نرم‌افزار، سامانهٔ پیگیری وظایف[1] و ... می‌توان دریافت که برای مدیریت پیکربندی از چه فناوری‌ای استفاده شده یا با بررسی کدمنبع، زبان‌برنامه‌نویسی و الگوهای معماری مورد استفاده را نیز می‌توان دریافت.

[1] issue tracking system

مراجع

[1] S. Farshidi, S. Jansen, R. de Jong, and S. Brinkkemper, "A decision support system for software technology selection," Journal of Decision Systems, vol.27, no.sup1, pp.98–110, 2018.

[2] P. Bourque, R. E. Fairley, et al. Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press, 2014.

[3] M. Soliman, M. Galster, A. R. Salama, and M. Riebisch, "Architectural knowledge for technology decisions in developer communities: An exploratory study with stackoverflow," in 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), pp.128–133, IEEE, 2016.

[4] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, pp.165–174, 2019.

[5] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pp.639–648, 2020.

[6] M. Mahabaleshwar, Tool support for architectural decision making in large software intensive projects. Ph.D. thesis, Technische Universität München, 2020.

[7] M. Bhat, C. Tinnes, K. Shumaiev, A. Biesdorf, U. Hohenstein, and F. Matthes, "Adex: A tool for automatic curation of design decision knowledge for architectural decision recommendations," in 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), pp.158–161, IEEE, 2019.

[8] J. van der Ven, Preserving and reusing architectural design decisions. Ph.D. thesis, University of Groningen, 2019.

[9] M. Bhat, K. Shumaiev, and F. Matthes, "Towards a framework for managing architectural design decisions," in Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings, pp.48–51, 2017.

[10] P. T. Nguyen, J. Di Rocco, and D. Di Ruscio, "Mining software repositories to support oss developers: A recommender systems approach.," in IIR, 2018.

[11] E. T. McGee, A decision support system for selecting between designs for dynamic software product lines. Ph.D. thesis, Clemson University, 2018.

[12] H. Van Vliet and A. Tang, "Decision making in software architecture," Journal of Systems and Software, vol.117, pp.638–644, 2016.

[13] S. Dasanayake, J. Markkula, S. Aaramaa, and M. Oivo, "Software architecture decision-making practices and challenges: an industrial case study," in 2015 24th Australasian Software Engineering Conference, pp.88–97, IEEE, 2015.

[14] I. C. Lopes Silva, P. H. Brito, B. F. dos S. Neto, E. Costa, and A. A. Silva, "A decision-making tool to support architectural designs based on quality attributes," in Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp.1457–1463, 2015.

[15] A. Zalewski, K. Borowa, and A. Ratkowski, "On cognitive biases in architecture decision making," in European Conference on Software Architecture, pp.123–137, Springer, 2017.

[16] G. Pedraza-García, H. Astudillo, and D. Correal, "Dvia: Understanding how software architects make decisions in design meetings," in Proceedings of the 2015 European Conference on Software Architecture Workshops, pp.1–7, 2015.

[17] A. Manjunath, M. Bhat, K. Shumaiev, A. Biesdorf, and F. Matthes, "Decision making and cognitive biases in designing software architectures," in 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), pp.52–55, IEEE, 2018.

[18] S. Farshidi, S. Jansen, and J. M. van der Werf, "Capturing software architecture knowledge for pattern-driven design," Journal of Systems and Software, vol.169, p.110714, 2020.

[19] A. Shokri, J. C. Santos, and M. Mirakhorli, "Arcode: Facilitating the use of application frameworks to implement tactics and patterns," in 2021 IEEE 18th International Conference on Software Architecture (ICSA), 2021.

[20] T. Bi, P. Liang, A. Tang, and X. Xia, "Mining architecture tactics and quality attributes knowledge in stack overflow," Journal of Systems and Software, vol.180, p.111005, 2021.

[21] F. L. De La Mora and S. Nadi, "Which library should i use?: A metric-based comparison of software libraries," in 2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER), pp.37–40, IEEE, 2018.

[22] M. A. Saied, A. Ouni, H. Sahraoui, R. G. Kula, K. Inoue, and D. Lo, "Improving reusability of software libraries through usage pattern mining," Journal of Systems and Software, vol.145, pp.164–179, 2018.

[23] A. Ouni, R. G. Kula, M. Kessentini, T. Ishio, D. M. German, and K. Inoue, "Search-based software library recommendation using multi-objective optimization," Information and Software Technology, vol.83, pp.55–75, 2017.

[24] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, and M. Di Penta, "Crossrec: Supporting software developers by recommending third-party libraries," Journal of Systems and Software, vol.161, p.110460, 2020.

[25] H. Yu, X. Xia, X. Zhao, and W. Qiu, "Combining collaborative filtering and topic modeling for more accurate android mobile app library recommendation," in Proceedings of the 9th Asia-Pacific Symposium on Internetware, pp.1–6, 2017.

[26] X. Zhao, S. Li, H. Yu, Y. Wang, and W. Qiu, "Accurate library recommendation using combining collaborative filtering and topic model for mobile development," IEICE TRANSACTIONS on Information and Systems, vol.102, no.3, pp.522–536, 2019.

[27] B. Abu-Salih, H. Alsawalqah, B. Elshqeirat, T. Issa, P. Wongthongtham, and K. K. Premi, "Toward a knowledge-based personalised recommender system for mobile app development," Journal of Universal Computer Science, vol.27, no.2, pp.208–229, 2021.

[28] Y. M. Mileva, V. Dallmeier, M. Burger, and A. Zeller, "Mining trends of library usage," in Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, pp.57–62, 2009.

[29] C. Teyton, J.-R. Falleri, M. Palyart, and X. Blanc, "A study of library migrations in java," Journal of Software: Evolution and Process, vol.26, no.11, pp.1030–1052, 2014.

[30] R. G. Kula, D. M. German, A. Ouni, T. Ishio, and K. Inoue, "Do developers update their library dependencies?," Empirical Software Engineering, vol.23, no.1, pp.384–417, 2018.

[31] J. S. van der Ven and J. Bosch, "Making the right decision: supporting architects with design decision data," in European Conference on Software Architecture, pp.176–183, Springer, 2013.

[32] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, and F. Matthes, "Automatic extraction of design decisions from issue management systems: a machine learning based approach," in European Conference on Software Architecture, pp.138–154, Springer, 2017.

[33] X. Li, P. Liang, and Z. Li, "Automatic identification of decisions from the hibernate developer mailing list," in Proceedings of the Evaluation and Assessment in Software Engineering, pp.51–60, Association for Computing MachineryNew YorkNYU-nited States, 2020.

[34] M. S. Ramaiah, T. Prabhakar, D. Rambabu, et al., "Archvoc–towards an ontology for software architecture," in Second Workshop on Sharing and Reusing Architectural Knowledge-Architecture, Rationale, and Design Intent (SHARK/ADI'07: ICSE Workshops 2007), pp.5–5, IEEE, 2007.

[35] Z. J. F. Al-Bayati, Coupling ontology with reference architectures to facilitate the instantiation process of software system architectures. Ph.D. thesis, University of Salford, 2019.

[36] W. Pinnoo, Development of an ontology for the problem space of. Ph.D. thesis, Ghent University, 2016.

[37] S. Moaven, J. Habibi, H. Ahmadi, and A. Kamandi, "A fuzzy model for solving architecture styles selection multi-criteria problem," in 2008 Second UKSIM European Symposium on Computer Modeling and Simulation, pp.388–393, IEEE, 2008.

[38] S. Moaven and J. Habibi, "A fuzzy-ahp-based approach to select software architecture based on quality attributes (fassa)," Knowledge and Information Systems, pp.1–29, 2020.

[39] M. Razavian, B. Paech, and A. Tang, "Empirical research for software architecture decision making: An analysis," Journal of Systems and Software, vol.149, pp.360–381, 2019.

[40] F. Thung, D. Lo, and J. Lawall, "Automated library recommendation," in 2013 20th Working conference on reverse engineering (WCRE), pp.182–191, IEEE, 2013.

[41] Q. He, B. Li, F. Chen, J. Grundy, X. Xia, and Y. Yang, "Diversified third-party library prediction for mobile app development," IEEE Transactions on Software Engineering, 2020.

[42] M. Chouchen, A. Ouni, and M. W. Mkaouer, "Androlib: Third-party software library recommendation for android applications," in International Conference on Software and Software Reuse, pp.208–225, Springer, 2020.

[43] B. Li, Q. He, F. Chen, X. Xia, L. Li, J. Grundy, and Y. Yang, "Embedding app-library graph for neural third party library recommendation," in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp.466–477, 2021.

[44] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining github," Empirical Software Engineering, vol.21, no.5, pp.2035–2071, 2016.

[45] Z. Meng, R. McCreadie, C. Macdonald, and I. Ounis, "Exploring data splitting strategies for the evaluation of recommendation models," in Fourteenth ACM conference on recommender systems, pp.681–686, 2020.

[46] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," Advances in neural information processing systems, vol.26, 2013.

واژه‌نامه فارسی به انگلیسی

آ

آپاچی اسپارک ..... Apache Spark

آپاچی هدوپ ..... Apache Hadoop

ا

استنتاج ..... Reasoning

اسپرینگ ..... Spring

اعتبارسنجی متقابل ..... Cross Validation

اعتماد ..... Confidence

افراز ..... Fold

الگوریتم ژنتیک ..... Genetic Algorithm

الگوواره ..... Paradigm

انتخاب‌های بدیل ..... Alternatives

اندازهٔ بردار جاساز ..... Embedding Vector Size

اندازهٔ دسته ..... Batch Size

اندازهٔ لایه ..... Layer Size

اندازه‌گیری ..... Measurement

اندروید آرسنال ..... Android Arsenal

انشعاب ..... Fork

ب

بازبینی کد ..... Code Review

بازخورد صریح ..... Explicit Feedback

بازخورد ضمنی ..... Implicit Feedback

بازیابی ..... Recall

بایت ..... Byte

بردار ویژگی‌های ورودی ..... Input Feature Vector

برون‌خطی ..... Offline

بصری‌سازی ..... Visualization

بهینه‌ساز ادم ..... Adam Optimizer

ت

تابع فعال‌سازی خطی ... Linear Activation Function
تبدیل ویژگی ... Feature Transformation
تفسیرپذیری ... Interpretability
تلفیق ... Hybrid
توصیه‌گر تلفیقی ... Hybrid Recommender
توقف زودهنگام ... Early Stopping
تکامل نرم‌افزار ... Software Evolution

ج

جاوا ... Java

ح

حذف تصادفی پیام ... Message Dropout

خ

خانواده‌های برنامه‌ها و چارچوب‌ها ... Families Of Programs And Frameworks
خشک ... Crisp
خوشه‌بندی سلسله مراتبی ... Hierarchical Clustering

د

دادهٔ اعتبارسنجی ... Validation Data
دانش خبره ... Expert Knowledge
درخواست واکشی ... Pull Request
دفتر جوپیتر ... Jupyter Notebook
دقت ... Precision
دوره ... Epoch
دیدگاه‌های معماری ... Architectural Viewpoints
دی‌بی‌پدیا ... Dbpedia

ز

زیست‌بوم ... Ecosystem

س

ساخت نرم‌افزار ... Software Construction
ساختار و معماری نرم‌افزار ... Software Structure And Architecture
سامانهٔ توصیه‌گر ... Recommender System
سامانهٔ پیگیری ایرادها ... Bug Tracker System
سامانهٔ پیگیری وظایف ... Issue Tracking System
سود انباشتهٔ کاهش‌یافته هنجارشده ... Normalized Discounted Cumulative Gain

سوگیری‌های شناختی ..... Cognitive Bias

سیگموید ..... Sigmoid

ش

شباهت کسینوسی ..... Cosine Similarity

شبکهٔ اجتماعی ..... Social Network

شبکهٔ پیچشی گراف سبک ..... Light Graph Convolution Network Filtering

شبکهٔ پیچشی گراف سبک ..... Light Graph Convolutional Network

شروع سرد ..... Cold Start

ص

صوری ..... Formal

ض

ضریب تنظیم ..... Regularization Coefficent

ط

طراحی نرم‌افزار ..... Software Design

ع

علوم اجتماعی ..... Social Science

علوم رفتاری ..... Behavioral Science

علوم طراحی ..... Design Science

ف

فایل دودویی ..... Binary File

فراسنج ..... Parameter

ق

قانون ..... Rule

قراردادها ..... Conventions

قلم ..... Item

ل

لیفت ..... Lift

م

مبتنی بر محتوا ..... Content Based

متلب ..... Matlab

محدوده ..... Scope

مدیر بسته‌های مِیوِن ..... Maven

مدیریت دانش ..... Knowledge Management

مشتبه ..... Misleading
مصنوع ..... Artifact
مطالعهٔ موردی ..... Case Study
معماری نرم‌افزار ..... Software Architecture
موضوعات ..... Topics

ن

نامپای ..... Numpy
نسخه ..... Version
نوظهوری ..... Novelty
نگاشت رابطه به شی ... Object Relation Mapping

ه

هایبرنت ..... Hibernate
هستان‌شناسی ..... Ontology

و

واسط برنامه‌نویسی کاربردی ..... Api
ویژگی‌های کیفی ..... Quality Attribute

پ

پالایش همکارانه ..... Collaborative Filtering
پالایش همکارانه گراف عصبی ..... Neural Graph Collaborative Filtering
پانداس ..... Pandas
پایتورچ ..... Pytorch
پایه‌واساس ..... Rationale
پردازش زبان طبیعی . Natural Language Processing
پشتیبانی ..... Support
پوچ ..... Null
پیشنهادهای غیرشخصی ..... Unpersonalized Recommendation

چ

چرخهٔ عمر ایجاد نرم‌افزار .. Software Development Life Cycle

ک

کاویدن مخازن پروژه‌های نرم‌افزار .. Mining Software Repositories
کتابخانه‌ها ..... Libraries
کمّی‌سازی ..... Quantification

گ

گرادیان کاهشی ..... Gradient Descent

گراف شبکه پیچشی ..... Graph Convolutional Network

گروه ..... Group

گوگل پلی ..... Google Play

گیت‌هاب ..... Github

ی

یادگیری عمیق ..... Deep Learning

یادگیری قانون انجمنی .. Association Rule Learning

واژه‌نامه انگلیسی به فارسی

A

بهینه‌ساز ادم ..... Adam Optimizer

انتخاب‌های بدیل ..... Alternatives

اندروید آرسنال ..... Android Arsenal

آپاچی هدوپ ..... Apache Hadoop

آپاچی اسپارک ..... Apache Spark

واسط برنامه‌نویسی کاربردی ..... Api

دیدگاه‌های معماری ..... Architectural Viewpoints

مصنوع ..... Artifact

یادگیری قانون انجمنی .. Association Rule Learning

B

اندازهٔ دسته ..... Batch Size

علوم رفتاری ..... Behavioral Science

فایل دودویی ..... Binary File

سامانهٔ پیگیری ایرادها ..... Bug Tracker System

بایت ..... Byte

C

مطالعهٔ موردی ..... Case Study

بازبینی کد ..... Code Review

سوگیری‌های شناختی ..... Cognitive Bias

شروع سرد ..... Cold Start

پالایش همکارانه ..... Collaborative Filtering

اعتماد ..... Confidence

مبتنی بر محتوا ..... Content Based

قراردادها ..... Conventions

شباهت کسینوسی ..... Cosine Similarity

خشک ..... Crisp

اعتبارسنجی متقابل ..... Cross Validation

D

دی‌بی‌پدیا ..... Dbpedia

یادگیری عمیق ..... Deep Learning

علوم طراحی ..... Design Science

E

توقف زودهنگام ..... Early Stopping

زیست‌بوم ..... Ecosystem

اندازهٔ بردار جاساز ..... Embedding Vector Size

دوره ..... Epoch

دانش خبره ..... Expert Knowledge

بازخورد صریح ..... Explicit Feedback

F

خانواده‌های برنامه‌ها و چارچوب‌ها ..... Families Of Programs And Frameworks

تبدیل ویژگی ..... Feature Transformation

افراز ..... Fold

انشعاب ..... Fork

صوری ..... Formal

G

الگوریتم ژنتیک ..... Genetic Algorithm

گیت‌هاب ..... Github

گوگل پلِی ..... Google Play

گرادیان کاهشی ..... Gradient Descent

گراف شبکه پیچشی ..... Graph Convolutional Network

گروه ..... Group

H

هایبرنت ..... Hibernate

خوشه‌بندی سلسله مراتبی .. Hierarchical Clustering

تلفیق ..... Hybrid

توصیه‌گر تلفیقی ..... Hybrid Recommender

I

بازخورد ضمنی ..... Implicit Feedback

بردار ویژگی‌های ورودی ..... Input Feature Vector

تفسیرپذیری ..... Interpretability

سامانهٔ پیگیری وظایف ..... Issue Tracking System

قلم ..... Item

J

جاوا ..... Java

دفتر جوپیتر ..... Jupyter Notebook

K

مدیریت دانش ..... Knowledge Management

L

اندازهٔ لایه ..... Layer Size

کتابخانه‌ها ..... Libraries

لیفت ..... Lift

شبکهٔ پیچشی گراف سبک ..... Light Graph Convolution Network Filtering

شبکهٔ پیچشی گراف سبک ..... Light Graph Convolutional Network

تابع فعال‌سازی خطی .. Linear Activation Function

M

متلب ..... Matlab

مدیر بسته‌های مِیوِن ..... Maven

اندازه‌گیری ..... Measurement

حذف تصادفی پیام ..... Message Dropout

کاویدن مخازن پروژه‌های نرم‌افزار .. Mining Software Repositories

مشتبه ..... Misleading

N

پردازش زبان طبیعی . Natural Language Processing

پالایش همکارانه گراف عصبی ..... Neural Graph Collaborative Filtering

سود انباشتهٔ کاهش‌یافته هنجارشده ..... Normalized Discounted Cumulative Gain

نوظهوری ..... Novelty

پوچ ..... Null

نامپای ..... Numpy

O

نگاشت رابطه به شی ... Object Relation Mapping

برون‌خطی ..... Offline

هستان‌شناسی ..... Ontology

P

پانداس ..... Pandas

الگوواره ..... Paradigm

فراسنج ..... Parameter

دقت ..... Precision

درخواست واکشی ..... Pull Request

پایتورچ ..... Pytorch

Q

ویژگی‌های کیفی ..... Quality Attribute

کمّی‌سازی ..... Quantification

R

پایه‌واساس ..... Rationale

استنتاج ..... Reasoning

بازیابی ..... Recall

سامانهٔ توصیه‌گر ..... Recommender System

ضریب تنظیم ..... Regularization Coefficent

قانون ..... Rule

S

محدوده ..... Scope

سیگموید ..... Sigmoid

شبکهٔ اجتماعی ..... Social Network

علوم اجتماعی ..... Social Science

معماری نرم‌افزار ..... Software Architecture

ساخت نرم‌افزار ..... Software Construction

طراحی نرم‌افزار ..... Software Design

چرخهٔ عمر ایجاد نرم‌افزار .. Software Development Life Cycle

تکامل نرم‌افزار ..... Software Evolution

ساختار و معماری نرم‌افزار . Software Structure And Architecture

اسپرینگ ..... Spring

پشتیبانی ..... Support

T

موضوعات ..... Topic

U

پیشنهادهای غیرشخصی ..... Non-personalized Recommendations

V

دادهٔ اعتبارسنجی ..... Validation Data

بصری‌سازی ..... Visualization

نسخه ..... Version

ضمیمه ۱: پرسمان استخراج وابستگی‌ها

```
CREATE TEMP FUNCTION
  ExtractVersion(version STRING
  ) AS (
 REGEXP_EXTRACT(`version`, r "\${(.*?)}")
);
CREATE TEMP FUNCTION
  ExtractDependencyBlock(
  content STRING) AS (
 REGEXP_EXTRACT_ALL(
  REGEXP_REPLACE(
   `content`, r "<exclusions>[\s\S]*?</exclusions>",
   ""
  ),
  r "<dependency>[\s\S]*?</dependency>"
 )
);
CREATE
OR REPLACE TABLE `ace-resolver-359805.github.1_java_dependency` AS WITH `pom_path` AS (
 SELECT
  `id`,
  `repo_name`,
  `path`
 FROM
  `bigquery-public-data.github_repos.files` `files`
 WHERE
  `files`.`path` LIKE "%pom.xml"
),
`pom_content` AS (
 SELECT
  `pom_path`.`repo_name`,
  ARRAY_AGG(
   STRUCT(
    `pom_path`.`path`,
    `contents`.`content`,
    ExtractDependencyBlock(`contents`.`content`) AS `dependency_block`
```

```
    ARRAY(
     SELECT
      AS STRUCT REGEXP_EXTRACT(
       `dependency_block`, r "<groupId>(.*?)</groupId>"
      ) `group`,
      REGEXP_EXTRACT(
       `dependency_block`, r "<artifactId>(.*?)</artifactId>"
      ) `artifact`,
      (
       SELECT
        IF (
         REGEXP_CONTAINS(`version`, r '\${[\w.-]*?}'),
         (
          SELECT
           REGEXP_EXTRACT(
            `content`,
            CONCAT(
             "<",
    )
   ) `pom`
  FROM
   `pom_path`
   JOIN `bigquery-public-data.github_repos.contents` `contents` ON `contents`.`id` = `pom_path`.`id`
  WHERE
   `contents`.`binary` = FALSE
  GROUP BY
   1
),
`pom_dependency` AS (
 SELECT
  `repo_name`,
  ARRAY_LENGTH(`pom`) `pom_count`,
  ARRAY_AGG(
   STRUCT(
    `path`,
    ARRAY_LENGTH(`dependency_block`) AS `dependency_count`,
```

```
            ExtractVersion(`
                version`),
            ">(.*?)</",
            ExtractVersion(`
                version`)
          )
        )
      ),
      `version`
    )
  FROM
    (
     SELECT
      REGEXP_EXTRACT(
       `dependency_block`, r
           "<version>(.*?)</
          version>"
      ) `version`
    )
) `version`,
REGEXP_EXTRACT(
 `dependency_block`, r "<
    scope>(.*?)</scope>"
) `scope`
    FROM
     UNNEST(`dependency_block`)
         `dependency_block`
    ORDER BY
     `version` DESC,
     `group` DESC,
     `artifact` DESC
   ) AS `dependency`
  )
 ) `pom`
FROM
 `pom_content`,
 UNNEST(`pom`)
GROUP BY
 1,
 2
)
SELECT
 `repo_name`,
 `pom_count`,
 SUM(`dependency_count`) `
    total_dependency_count`,
 ARRAY_AGG(
  STRUCT(
```

```
    `path`, `dependency_count`, `
        dependency`
  )
 ) `pom`
FROM
 `pom_dependency`,
 UNNEST(`pom`)
GROUP BY
 1,
 2
ORDER BY
 `total_dependency_count` DESC,
 `pom_count` DESC;
```

ضمیمه ۲: پرسمان استخراج اطلاعات مخازن

```
CREATE
OR REPLACE TABLE `ace-resolver-359805.github.2_java_repository` AS WITH `dependency` AS (
 SELECT
  `repo_name`,
  ARRAY_AGG(`id`) [ OFFSET (0) ] `id`
 FROM
  (
   SELECT
    ARRAY_AGG(`repo_name`) [ OFFSET (0) ] `repo_name`,
    `id`
   FROM
    (
     SELECT
      DISTINCT `repo_name`,
      `id`
     FROM
      (
       SELECT
        `repo_name`,
        `repo`.`id`,
        `created_at`
       FROM
        `ace-resolver-359805.github.1_java_dependency`
        LEFT JOIN `githubarchive.month.*` ON `repo_name` = `repo`.`name`
       ORDER BY
        `created_at` DESC
      )
    )
   WHERE
    `id` IS NOT NULL
   GROUP BY
    `id`
  )
GROUP BY
```

```
  `repo_name`
),
`languages` AS (
 SELECT
  `languages`.`repo_name`,
  `first_language`.`language` [
      OFFSET (0) ] `language`
 FROM
  `bigquery-public-data.
      github_repos.languages` `
      languages`
 LEFT JOIN (
  SELECT
   `repo_name`,
   ARRAY_AGG(`name`) `language`
  FROM
   `bigquery-public-data.
       github_repos.languages` `
       t1`,
   UNNEST(`language`)
  WHERE
   `bytes` = (
    SELECT
     MAX(`bytes`)
    FROM
     `bigquery-public-data.
         github_repos.languages`
          `t2`,
     UNNEST(`language`)
    WHERE
     `t1`.`repo_name` = `t2`.`
         repo_name`
   )
  GROUP BY
   1
 ) `first_language` ON `
     languages`.`repo_name` = `
     first_language`.`repo_name`
),
`repo_stars` AS (
 SELECT
  `repo`.`name` `repo_name`,
  COUNT(DISTINCT `actor`.`id`) `
      star_count`,
 FROM
  `githubarchive.month.*`
 WHERE
  `type` = "WatchEvent"
```

```
 GROUP BY
  `repo`.`name`
),
`repo_forks` AS (
 SELECT
  `repo`.`name` `repo_name`,
  COUNT(DISTINCT `actor`.`id`) `
     fork_count`,
 FROM
  `githubarchive.month.*`
 WHERE
  `type` = "ForkEvent"
 GROUP BY
  `repo`.`name`
),
`repo_commits` AS (
 SELECT
  `repo_name`,
  COUNT(`commit`) `commit_count
     `,
  COUNT(DISTINCT `author`.`email
     `) `author_count`,
  MIN(`author`.`time_sec`) `
     first_commit_sec`,
  MAX(`author`.`time_sec`) `
     last_commit_sec`,
  MAX(`author`.`time_sec`)- MIN
     (`author`.`time_sec`) `
     commit_duration_sec`
 FROM
  `bigquery-public-data.
     github_repos.commits`,
  UNNEST(`repo_name`) `repo_name
     `
 WHERE
  `repo_name` IN (
   SELECT
    `repo_name`
   FROM
    `ace-resolver-359805.github
       .1_java_dependency`
  )
 GROUP BY
  1
),
`repo_recent_commits` AS (
 SELECT
  `repo_name`,
```

```
)
SELECT
 `languages`.`repo_name`,
 `languages`.`language`,
 IFNULL(`repo_stars`.`star_count`, 0) `star_count`,
 IFNULL(`repo_forks`.`fork_count`, 0) `fork_count`,
 `commit_count`,
 IFNULL(`recent_commit_count`, 0) `recent_commit_count`,
 `author_count`,
 `recent_author_count`,
 `first_commit_sec`,
 `last_commit_sec`,
 `commit_duration_sec`
FROM
 `dependency`
 LEFT JOIN `languages` ON `dependency`.`repo_name` = `languages`.`repo_name`
 LEFT JOIN `repo_stars` ON `dependency`.`repo_name` = `repo_stars`.`repo_name`
 COUNT(`commit`) `recent_commit_count`,
 COUNT(DISTINCT `author`.`email`) `recent_author_count`
FROM
 `bigquery-public-data.github_repos.commits`,
 UNNEST(`repo_name`) `repo_name`
WHERE
 `repo_name` IN (
  SELECT
   `repo_name`
  FROM
   `ace-resolver-359805.github.1_java_dependency`
 )
 AND author.time_sec >= UNIX_SECONDS(
  TIMESTAMP "2020-01-01 00:00:00+00"
 )
GROUP BY
 1
```

```
LEFT JOIN `repo_forks` ON `
    dependency`.`repo_name` = `
    repo_forks`.`repo_name`
JOIN `repo_commits` ON `
    dependency`.`repo_name` = `
    repo_commits`.`repo_name`
LEFT JOIN `repo_recent_commits`
     ON `dependency`.`repo_name`
     = `repo_recent_commits`.`
    repo_name`
ORDER BY
 `star_count` DESC,
 `language`;
```

Abstract:

A popular research area in recent years has been mining software repositories of reliable free/open-source software projects to harness the crowd's wisdom. Due to the growing number of available technologies, selecting an appropriate one has become a challenge for software architects. So, building a recommender system is a suitable solution for supporting architects in technology selection. For building these systems, researchers use a variety of machine learning methods. However, the quality and size of the project have not been examined. Additionally, their recommendations do not perform well enough considering the recent advancements in deep learning algorithms, and they have the cold start problem. In this study, two recommender systems are developed once quality data is extracted. The first is a deep learning-based recommender called DeepLibAide, while the second is a content-based recommender called ContentLibAide and a hybrid recommender that solves the cold start problem. Furthermore, we compare the results of their implementation on different samples based on different levels of project quality. DeepLibAide shows significant improvement in the criteria of accuracy, recall, and normalized cumulative gain and training loss with respect to the baseline method. This is because it uses the linear activation function, removing features transformation and summing the embedding vectors of different layers. A dataset containing only high-quality projects and a dataset containing a random sample of projects

was compared for precision and recall. On average, DeepLibAide improves recall criteria by 7 percent. Throughout the pipeline architecture, raw input data is extracted or updated, samples are built, and models are trained. Several parameters are used in the pipeline, and recommendations are stored. After that, various diagrams are used to evaluate and compare them with the baseline method. It is possible to execute the entire program automatically and with negligible overhead.

Keywords: technology selection, recommender system, deep learning, software library, dataset extraction

Shahid Beheshti University
Faculty of Computer Science and Engineering

Using empirical data to support technology selection in software architecture decision-making

By

Seyed Mohammad Masoud Sadrnezhaad

A THESIS SUBMITTED
FOR THE DEGREE OF
MASTER OF SCIENCE

Supervisor:

Dr. Sadegh Aliakbary

July 2022