



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

سید محمد مسعود صدرنژاد

استاد راهنما

دکتر صادق علی اکبری

پاییز ۱۴۰۱



دانشگاه شهید بهشتی
دانشکده مهندسی و علوم کامپیوتر

پایان نامه کارشناسی ارشد فناوری اطلاعات - گرایش معماری سازمانی
تحت عنوان:

استفاده از داده‌های تجربی برای پشتیبانی از انتخاب فناوری در تصمیم‌گیری معماری نرم افزار

در تاریخ ۱۴۰۱/۰۷/۲۷ پایان نامه دانشجو، سید محمد مسعود صدرنژاد، توسط کمیته تخصصی داوران مورد بررسی و تصویب
نهایی قرار گرفت.

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

با سپاس و قدردانی از

استاد گرامی، جناب آقای دکتر صادق علی اکبری که به ثمر رسیدن این پایان نامه، بدون زحمات پیوسته و راهنمایی های خیرخواهانه و بی دریغشان محال بود.

برادر عزیزم مهدی صدرنژاد، به خاطر مشورت ها و منابع ابری که برای انجام پردازش های مورد نیاز در این پایان نامه به من اهدا کرد.

کلیه حقوق مادی مترتب بر نتایج مطالعات،
ابتکارات و نوآوری‌های ناشی از تحقیق موضوع
این پایان‌نامه متعلق به دانشگاه شهید بهشتی
می‌باشد.

به نام خدا

نام و نام خانوادگی: سید محمد مسعود صدرنژاد

عنوان پایان نامه: استفاده از داده‌های تجربی برای پشتیبانی از انتخاب فناوری در تصمیم‌گیری معماری نرم افزار

استاد راهنما: دکتر صادق علی اکبری

اینجانب سید محمد مسعود صدرنژاد تهیه کننده پایان نامه 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 گشته است 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: سید محمد مسعود صدرنژاد

تاریخ و امضا:

تقدیم به

دلسوزترین و فداکارترین همراهان زندگی؛

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

چکیده

استفاده از خرد جمعی^۱ معماران نرم افزار از طریق کاویدن مخازن پروژه های نرم افزار^۲ های معتبر آزاد/متن باز^۳، یکی از حوزه های پژوهشی است که در سال های اخیر با استقبال بیشتری روبرو شده است. با توجه به رویش روزافزون فناوری های در دسترس، انتخاب فناوری های مناسب از میان آن ها به یکی از چالش های پیش روی معماران نرم افزار تبدیل شده است. سامانه توصیه گر راهکاری مناسب برای پشتیبانی از معمار در انتخاب فناوری است. پژوهش های صورت گرفته در این حوزه از طیف وسیعی از روش های یادگیری ماشین برای ساخت این سامانه ها استفاده کردند ولی تأثیر کیفیت و اندازه پروژه های استفاده شده مورد مطالعه قرار نگرفته است. همچنین دقت پیشنهادهایشان با توجه به پیشرفت های اخیر در الگوریتم های یادگیری عمیق چندان مناسب نیست و به مشکل شروع سرد مبتلا هستند. این پژوهش پس از استخراج داده کیفیت پروژه ها، با توسعه دو سامانه توصیه گر مبتنی بر یادگیری عمیق به نام DeepLibAide و مبتنی بر محتوا به نام ContentLibAide و توصیه گر تلفیقی با هدف حل مشکل شروع سرد، به مقایسه نتایج اجرای آن ها روی نمونه های مختلف از کیفیت پروژه ها می پردازد. مقایسه نتایج DeepLibAide نشان می دهد که به واسطه استفاده از تابع فعال ساز خطی، حذف تبدیل ویژگی ها و جمع زدن بردارهای جاساز لایه ها، نتایج ارزیابی مدل طراحی شده در فراسنج^۴ های مختلف، نسبت به روش پایه در معیارهای دقت، بازیابی و سود انباشته کاهش یافته نرمال شده و ضرر آزمون بهبود یافته است. دقت و بازیابی آموزش مدل توسط دادگان بزرگتر، دادگان حاوی فقط پروژه های با کیفیت تر و انتخاب تصادفی پروژه ها مقایسه شده است. DeepLibAide نسبت به روش پایه به طور میانگین هفت درصد در معیار بازیابی بهبود یافته. معماری خط لوله طراحی شده از ابتدای استخراج یا به روزرسانی دادگان ورودی و ساخت نمونه ها تا مرحله آموزش مدل ها با فراسنج های مختلف و ذخیره سازی پیشنهادهای و ارزیابی و مقایسه آن ها با روش پایه در نمودارهای مختلف طوری طراحی شده است که کل برنامه به طور خودکار و با حداقل نیاز به دستکاری اجرا شود.

واژگان کلیدی: انتخاب فناوری، سامانه توصیه گر^۵، یادگیری عمیق، کتابخانه نرم افزاری^۶، استخراج دادگان

¹wisdom of the crowd

²mining software repositories

³free/open source software

⁴parameter

⁵recommender system

⁶software library

فهرست مطالب

۱	مقدمه	۱
۲	۱.۱ تعریف مسئله	۲
۳	۲.۱ اهمیت و ضرورت موضوع	۳
۴	۳.۱ اهداف پایان نامه	۴
۶	۴.۱ ساختار پایان نامه	۶
۷	۲ ادبیات و مفاهیم اولیه	۷
۸	۱.۲ معماری نرم افزار	۸
۸	۱.۱.۲ انواع تصمیم های معماری نرم افزار	۸
۹	۲.۱.۲ اجزای تصمیم معماری نرم افزار	۹
۱۰	۲.۲ روش های پشتیبانی از تصمیم گیری	۱۰
۱۰	۱.۲.۲ روش های استنتاج براساس دانش خبره	۱۰
۱۱	۲.۲.۲ سامانه های توصیه گر مبتنی بر الگوریتم های یادگیری سنتی	۱۱
۱۱	۱.۲.۲.۲ یادگیری قانون انجمنی	۱۱
۱۲	۲.۲.۲.۲ پالایش همکارانه	۱۲
۱۲	۳.۲.۲.۲ مبتنی بر محتوا	۱۲
۱۳	۳.۲.۲ سامانه های توصیه گر مبتنی بر یادگیری عمیق	۱۳

۱۳	گراف شبکه پیچشی	۱.۳.۲.۲
۱۳	پالایش همکارانه گراف عصبی	۲.۳.۲.۲
۱۴	شبکه پیچشی گراف سبک	۳.۳.۲.۲

۳ کارهای پیشین

۱۶	طبقه‌بندی کارهای پیشین	۱.۳
۲۰	پژوهش‌های پایه برای سامانه توصیه‌گر کتابخانه	۲.۳
۲۱	جمع‌بندی کارهای پیشین	۳.۳

۴ روش پیشنهادی

۲۶	جمع‌آوری داده	۱.۴
۲۶	دادگان‌های مورد استفاده	۱.۱.۴
۲۷	استخراج فناوری‌های مورد استفاده	۲.۱.۴
۲۸	حذف موارد کتابخانه‌های اشتباه با تطبیق الگو	۱.۲.۱.۴
۲۸	استخراج داده کیفیت و اندازه پروژه‌ها	۳.۱.۴
۲۸	فرایند استخراج داده کیفیت و اندازه پروژه‌ها	۱.۳.۱.۴
۲۹	حذف پروژه‌های بی‌فایده برای مراحل بعد	۲.۳.۱.۴
۳۰	استخراج داده برچسب‌ها برای پروژه‌ها	۴.۱.۴
۳۰	ساخت داشبورد تعاملی برای اکتشاف داده‌ها	۲.۴
۳۱	ارائه سامانه توصیه‌گر بهبود یافته	۳.۴
۳۲	ترکیب خطی داده‌های کیفیت پروژه‌ها	۱.۳.۴
۳۳	استفاده از روش‌های رایج اعتبارسنجی متقابل	۲.۳.۴
۳۴	روش یک قلم آخر را بیرون بگذار	۱.۲.۳.۴
۳۴	روش یک سبد را بیرون بگذار	۲.۲.۳.۴
۳۵	روش جداسازی تقسیم زمانی مبتنی بر کاربر	۳.۲.۳.۴

۳۵	تقسیم‌بندی داده به دادهٔ آزمون، آموزش و اعتبارسنجی	۳.۳.۴
۳۶	بازنمایی ماتریس مجاورت دادهٔ آزمون	۴.۳.۴
۳۶	آموزش مدل یادگیری عمیق	۵.۳.۴
۳۷	تبیین روش پیشنهادی با مثال	۱.۵.۳.۴
۳۹	تلفیق با سامانهٔ توصیه‌گر مبتنی بر محتوا	۶.۳.۴
۳۹	جمع‌بندی روش پیشنهادی	۴.۴

۵ ارزیابی

۴۲		
۴۳	دادگان مورد استفاده	۱.۵
۴۴	معیارهای ارزیابی	۲.۵
۴۵	پیاده‌سازی	۳.۵
۴۶	فراسنج‌های مؤثر در اجرا	۴.۵
۴۸	نتایج ارزیابی	۵.۵
۴۹	مقایسهٔ میانگین دو روش به‌طور کلی	۱.۵.۵
۵۱	مقایسهٔ نتایج ارزیابی در طول روند یادگیری	۲.۵.۵
۵۳	ارزیابی به تفکیک تعداد دادهٔ آزمون	۳.۵.۵
۵۳	تاثیر کیفیت و حجم دادهٔ موجود در دادگان آزمون روی نتایج ارزیابی	۴.۵.۵
۵۵	ضریب تنظیم و کاهش مسئلهٔ بیش‌برازش	۵.۵.۵
۵۵	روند تغییرات ضرر هنگام آموزش مدل	۶.۵.۵

۶ نتیجه‌گیری و کارهای آینده

۵۸		
۵۹	جمع‌بندی	۱.۶
۶۰	کارهای آینده	۲.۶
۶۰	استخراج دانش سطح بالا یا پایه‌و‌اساس تصمیم‌های معماری	۱.۲.۶
۶۱	تفسیرپذیری مدل	۱.۱.۲.۶

۶۱	استفاده از داده‌های متنوع‌تر	۲.۱.۲.۶
۶۲	بازنمایی و مدیریت دانش معماری	۳.۱.۲.۶
۶۲	مطالعه روندها و تغییرات در تصمیمات انتخاب فناوری	۲.۲.۶
۶۲	روش‌های ارزیابی متنوع‌تر	۳.۲.۶
۶۳	مطالعه سایر انواع تصمیم‌های معماری	۴.۲.۶

۶۴

مراجع

فهرست شکل‌ها

۱.۲	جانمایی معماری نرم‌افزار در درخت موضوعی مهندسی نرم‌افزار [۲]	۹
۱.۳	درخت موضوعی پژوهش‌ها در زمینه پشتیبانی از تصمیم‌گیری در معماری نرم‌افزار	۱۶
۱.۴	شمای کلی روش پیشنهادی در قالب نمودار فعالیت	۲۵
۲.۴	داشبورد تحلیل و مقایسه فناوری‌های به کار رفته	۳۱
۳.۴	شمای کلی معماری و مولفه‌های تشکیل‌دهنده سامانه توصیه‌گر در قالب نمودار استقرار	۴۱
۱.۵	توزیع کیفی پروژه‌ها در هر کدام از نمونه‌ها در قالب نمودار جعبه‌ای	۴۴
۲.۵	مقایسه میانگین میزان اختلاف بهبود بازیابی در روش پیشنهادی نسبت به روش پایه برحسب جایگاه پیشنهاد	۴۹
۳.۵	مقایسه دقت برحسب دوره برای ده پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر	۵۱
۴.۵	مقایسه بازیابی برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر	۵۱
۵.۵	مقایسه سکه برحسب دوره برای ده پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر	۵۲
۶.۵	مقایسه دقت برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر	۵۲
۷.۵	مقایسه بازیابی برحسب دوره برای ده پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر	۵۲
۸.۵	مقایسه سکه برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر	۵۳
۹.۵	مقایسه سکه برحسب دوره در روش پیشنهادی شامل نمونه کامل و باکیفیت و تصادفی در شرایط برابر	۵۵

- ۱۰.۵ مقایسهٔ دقت برحسب دوره در روش پیشنهادی شامل نمونهٔ کامل و باکیفیت و تصادفی در شرایط برابر ۵۶
- ۱۱.۵ مقایسهٔ بازیابی برحسب دوره در روش پیشنهادی شامل نمونهٔ کامل و باکیفیت و تصادفی در شرایط برابر ۵۶
- ۱۲.۵ مقایسهٔ سود انباشتهٔ کاهش یافته هنجار شده برحسب ضریب تنظیم در روش پیشنهادی و روش پایه در شرایط برابر ۵۷
- ۱۳.۵ مقایسهٔ ضرر برحسب دوره در روش پیشنهادی و روش پایه در شرایط برابر ۵۷

فهرست جداول

۱.۳	مقایسهٔ پژوهش‌های پایه	۲۲
۱.۵	آمارهایی دربارهٔ دادگان‌های آزمایشی	۴۳
۲.۵	انحراف معیار نتایج ارزیابی در افرازاها	۴۸
۳.۵	مقایسهٔ بازیابی، سکه و دقت بین روش پیشنهادی و پایه	۵۰
۴.۵	دقت، بازیابی و سکه برای روش پیشنهادی برای تعداد کتابخانه‌های داخل آزمون مختلف	۵۴

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 درباره 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 منجر به نیاز به مطالعه روش‌های تصمیم‌گیری در حوزه معماری نرم‌افزار شده است.

پژوهشگران این حوزه تأکید می‌کنند که تصمیم‌گیری درباره معماری نرم‌افزار، فرایندی پیچیده است و معمولاً چگونگی تصمیم‌گیری معمار واضح نیست. گاهی حتی خود فرد تصمیم‌گیر هم نمی‌تواند مشخص کند که چگونه تصمیم به خصوصی را گرفته است. این پژوهش در قالب یک سامانه توصیه‌گر^۱ با پیشنهاد فناوری، در اتخاذ تصمیم‌های معماری نرم‌افزار در حوزه انتخاب فناوری به معمار نرم‌افزار یاری می‌رساند.

۱.۱ تعریف مسئله

تصمیم‌گیری توسط معمار، نیازمند دانش قبلی درباره زمینه، هدف تصمیم‌گیری یا معیارهای ارزیابی، انتخاب‌های بدیل^۲ 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 درباره هدف، انتخاب‌های بدیل و وضعیت هر انتخاب در قبال هر معیار است. در بیشتر مواقع استفاده از روش‌های استنتاج براساس دانش خبره در زمینه تصمیم‌گیری معماری نرم‌افزار به‌تنهایی کافی نیست زیرا قسمت دشوار فعالیت معمار اخذ دانش مورد نیاز درباره 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 درباره همین سه مورد است. دانش سطح بالای مورد نیاز برای استفاده از روش‌های استنتاج براساس دانش خبره، دانش ضمنی^۳ معماران خبره است که مخصوصاً با محبوبیت بیشتر روشگان چابک، معمولاً مستند نشده و به‌طور صریح بیان نمی‌شود.

¹recommender system

²alternatives

³implicit

این درحالی است که خود تصمیم‌های معماری در سوابق یک پروژه 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 درباره معماری که خود تصمیم‌های اتخاذ شده است با بررسی کدمنبع قابل استخراج است ولی دانش سطح بالای استفاده شده 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 کمی‌سازی 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 کمی‌سازی^۳، اندازه‌گیری^۴ و به‌طور کلی روش‌های مهندسی برای تمام بخش‌های عملیات معماری یک زمینه پژوهشی شایسته توجه است. کمی‌سازی 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 مطالعه پژوهش‌های پیشین و بررسی خلأهای موجود در حوزه 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 بر مبنای 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 ارائه یک مدل که بتواند به‌طور خودکار با فناوری‌های جدید تکمیل شود و وابسته به جزئیات خصوصیات هر فناوری نباشد امری ضروری است.

¹alternatives

²rationale

³quantification

⁴measurement

⁵craftmanship

برای این منظور استفاده از روش‌های خودکار برای استخراج فناوری‌ها و وابسته کردن تصمیم‌گیری به نحوه استفاده از آن‌ها به جای ویژگی‌های خود فناوری‌ها، باعث افزایش انعطاف در برابر رشد و تغییرات فناوری‌ها در طی زمان شود. از سوی دیگر (ج) هرچند که پژوهش‌های اندکی روی تصمیم‌گیری^۱ درباره انتخاب فناوری توسط معماران انجام شده است ولی موضوع انتخاب فناوری، بیشتر در پژوهش‌های اخیر مورد توجه قرار گرفته است. 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 مطالعه میدانی و مصاحبه و ... برای ارزیابی یافته‌ها نیست.

۳.۱ اهداف پایان‌نامه

به‌طور نظری هر سامانه نرم‌افزاری، دارای معماری است حتی اگر آن معماری مستند نشده باشد یا طراحان آن در دسترس نباشند اما در عمل استخراج همه دانش معماری درخصوص پایه‌و‌اساس تصمیم‌ها با فقط در اختیار 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 پشته فناوری موجود است. برای این منظور می‌توان فناوری‌هایی که در پروژه‌های معتبر با هم به کار می‌روند را مطالعه کرد یا شبکه وابستگی‌های آن‌ها و نسخه‌های به‌کار رفته آن‌ها را شناسایی و مورد مطالعه قرارداد. این روش همه پایه‌و‌اساس تصمیم‌ها را مشخص نمی‌کند ولی تا همین جا می‌شود پیشنهادهای خوبی به معمار برای استفاده از فناوری‌ها از روی مدل ساخته شده از روی پروژه‌های پیشین داد؛ مثلاً معمار می‌تواند مجموعه‌ای از انتخاب‌های فناوری فعلی را (فارغ از پایه و اساس انتخاب‌هایش) به سامانه 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 ستاره بالا در کنار یک چارچوب وب مانند اسپرینگ^۲ چه برنامه‌ای برای نگاشت رابطه به شی عمده‌تاً مورد استفاده قرار گرفته است. سپس معمار می‌تواند با استفاده از داده‌هایی که دریافت کرده انتخاب‌های فعلی فناوری را بهبود دهد. برای این منظور مفروضاتی برای

¹decision-making

²Spring

کوچک‌تر کردن دامنه 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) داده آموزشی به کدهای منتشر شده از پروژه‌های جاوا و کتابخانه‌های مخزن میون محدود شده‌است.

یکی از نقاط ضعف روش بیان‌شده برای پیشنهاد فناوری به معمار بر اساس داده فناوری‌های به کار رفته در پروژه‌های پیشین، شروع سرد است. مسئله 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 مراحل باشد که هنوز به بلوغ کافی نرسیده، داده کافی درباره فناوری‌های مورد استفاده در آن پروژه وجود ندارد و بدون داشتن داده کافی در مورد یک پروژه، نمی‌توان فناوری‌هایی برای استفاده در آن پیشنهاد کرد. برای حل این مسئله از منابع داده ثانویه برای بررسی شباهت محتوای پروژه جدید با پروژه‌های پیشین استفاده شده‌است و به این ترتیب حتی در صورت نداشتن داده کافی درباره انتخاب‌های فناوری موجود در یک پروژه، می‌توان پیشنهادهایی به معمار ارائه داد.

سوال‌های اصلی

۱. با مطالعه فناوری‌های استفاده‌شده در معماری پروژه‌های پیشین، چگونه می‌توان انتخاب‌های فناوری انجام گرفته

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^۱ هستند، یا به عبارت دیگر هنوز داده کافی درباره انتخاب‌های فناوری موجود در آنها

وجود ندارد، چگونه بدون وجود داده کافی درباره انتخاب‌های پیشین در یک پروژه، می‌توان پیشنهادی

شخصی‌سازی‌شده برای چنین پروژه‌هایی ارائه داد؟ چگونه می‌توان داده‌های جانبی درباره 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 پیشنهادی تولیدشده

توسط مدل دارد؟

فرضیه‌ها

¹cold 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 بقیه فناوری‌ها هم باید توافق داشته باشند.
 ۳. با داشتن موارد استفاده از فناوری‌ها در پروژه‌های گذشته، می‌توان به معمار گزارش‌های سودمندی برای تصمیم‌گیری درباره 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 ارائه پیشنهادها شود و می‌توان از اطلاعات جانبی برای حل مسئله شروع سرد در یک سامانه 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 اتخاذشده پیشین در تصمیم‌گیری معماری نرم‌افزار استفاده شود. این رویکرد توجه ویژه‌ای به اهمیت نقش عامل انسانی و دانش معمار به عنوان پیش‌رو در تصمیم‌گیری نهایی در مورد انتخاب‌های فناوری دارد.

۴.۱ ساختار پایان‌نامه

این پایان‌نامه شامل شش فصل است. فصل دوم دربرگیرنده ادبیات موضوع و تعاریف اولیه مرتبط با مسئله پایان‌نامه و روش‌های مورد استفاده در این حوزه است. در فصل سوم کارهای مرتبطی که در زمینه انتخاب فناوری انجام شده به تفصیل بیان می‌گردد. در فصل چهارم روش پیشنهادی این پژوهش برای حل مسئله فوق مورد بررسی قرار گرفته است. 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 به کار رفته در پژوهش‌های پایه می‌شود. در انتها با در نظر گرفتن معیارها و فراسنج‌های بیان‌شده، به مقایسه روش پیشنهادی با پژوهش پایه پرداخته شده است. فصل ششم به نتیجه‌گیری و پیشنهادهایی برای کارهای آتی خواهد پرداخت.

فصل ۲

ادبیات و مفاهیم اولیه

این فصل دربرگیرنده ادبیات موضوع و تعاریف اولیه مرتبط با این پایان نامه است.

۱.۲ معماری نرم افزار

طراحی نرم افزار هم به معنای «فرایند تعریف معماری، مولفه ها، واسطه ها و سایر مشخصات یک سامانه یا مولفه» و هم «نتیجه فرایند مذکور» است. اگر به طراحی به عنوان یک فرایند نگاه شود، طراحی یکی از فعالیت هایی خواهد بود که در چرخه عمر ایجاد نرم افزار^۱ مورد توجه قرار می گیرد. طراحی به عنوان یک فعالیت به معنای تحلیل نیازمندی ها برای تولید توصیفی از ساختار داخلی برنامه است که اساس ساخت نرم افزار^۲ است.

تعریف ۱.۲ (معماری نرم افزار). معماری نرم افزار مجموعه ساختارهای مورد نیاز برای استدلال در مورد سامانه، شامل عناصر نرم افزاری، روابط بین آنها و خصوصیات هر دو است و در فازهای اولیه فرایند ایجاد نرم افزار با هدف رفع نیازهای خاص معماری^۳ و تسهیل اهداف حرفه طراحی می شود.

در نسخه سوم کتاب پیکره دانش مهندسی نرم افزار^۴، معماری نرم افزار بخشی از فرایند طراحی نرم افزار و طراحی نرم افزار بخشی از چرخه عمر نرم افزار است. جایگاه معماری نرم افزار در درخت موضوعی مهندسی نرم افزار در شکل ۱.۲ نمایش داده شده است. هدف از طراحی نرم افزار پر کردن خلأ میان نیازمندی ها تا ساخت نرم افزار است [۲]. گره هایی از این درخت که به مباحث مطرح شده در این پژوهش مرتبط هستند با خط توپر مشخص شده اند.

مطابق شکل ۱.۲، ساختار و معماری نرم افزار^۵، ذیل عنوان طراحی نرم افزار^۶ مورد بررسی قرار می گیرد و ذیل موضوع ساختار و معماری نرم افزار به مفاهیم ساختارها و دیدگاه های معماری^۷، سبک های معماری، الگوهای طراحی، تصمیم های طراحی معماری، خانواده های برنامه ها و چارچوب ها^۸ پرداخته شده است [۲].

۱.۱.۲ انواع تصمیم های معماری نرم افزار

همان طور که در بخش قبل بیان شد، از بین مباحث مطرح ذیل عنوان ساختار و معماری نرم افزار، مفهوم تصمیم های طراحی معماری بیش از همه به هدف این پژوهش نزدیک است.

¹software development life cycle

²software construction

³architecturally significant requirements

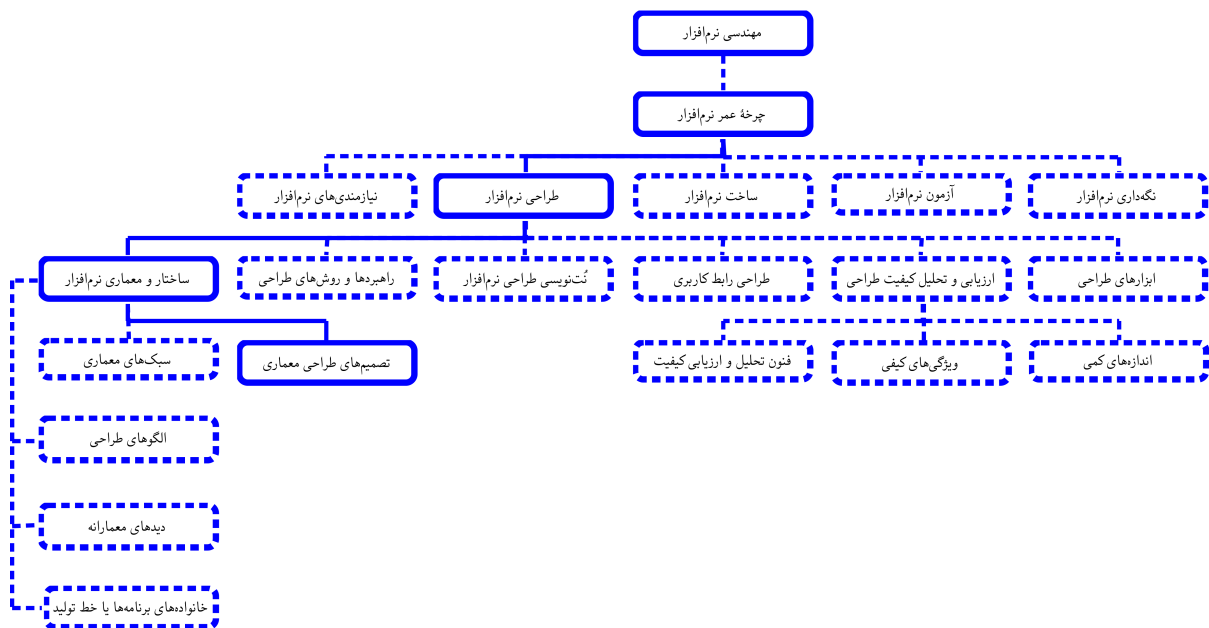
⁴Software Engineering Body of Knowledge (SWEBOK)

⁵software structure and architecture

⁶software design

⁷architectural viewpoints

⁸families of programs and frameworks



شکل ۱۰۲: جانمایی معماری نرم افزار در درخت موضوعی مهندسی نرم افزار [۲]

تعریف ۲.۲ (تصمیمات معماری نرم افزار). 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 مطالعه انجام شده نزدیک به ۲۵ درصد از تصمیمات معماری در پروژه‌ها مرتبط با انتخاب‌های فناوری بودند [۳].

۲.۱.۲ اجزای تصمیم معماری نرم افزار

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

۱. زمینه تصمیم: انتخاب فناوری

۲. هدف تصمیم‌گیری: انطباق فناوری برای نگاشت رابطه به شی^۱ که با پشته 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 این که کدام تصمیم‌ها از بین شش تصمیم اصلی دیگر معمار قرار است با انتخاب فناوری محقق شود در هدف تصمیم‌گیری جای می‌گیرد.

۳. مجموعه انتخاب‌های بدیل برای محقق شدن هدف تصمیم‌گیری: فناوری‌های هایبرنت^۲ 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 شده است.

موارد یک تا چهار پایه‌واساس^۳ 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 پشته فناوری درحال استفاده. برای مثال این که «آیا این فناوری می‌تواند در بالای پشته فناوری کنونی اجرا شود؟ آیا می‌تواند با پشته فعلی ارتباط برقرار کند؟ تا چه اندازه 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 نحوه رسیدن از دانش سطح بالا که توسط یک شخص خبره تأمین شده به دانش سطح پایین یا تصمیمات نهایی است؛

¹Object Relation Mapping

²Hibernate

³rationale

مثلاً در روش‌های تصمیم‌گیری چند معیاره با یک تحلیل بده‌بستان می‌توان نقطه 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 در صورت 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 درباره تمام آن اقلام (به خاطر تعداد زیادشان) اطلاعات کافی به دست آورد. سامانه توصیه‌گر در چنین شرایطی فهرستی مرتب شده از اقلام را به کاربر پیشنهاد کند یا می‌تواند نظر کاربر درباره هر قلم را پیش‌بینی کند. یک نوع ساده سامانه‌های توصیه‌گر پیشنهاد‌های غیرشخصی^۱ 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. دسته دیگر پیشنهاد‌های نیمه‌شخصی ارائه می‌کنند و دسته دیگر پیشنهاد‌های کاملاً شخصی برای کاربر تولید می‌کنند و این سامانه‌ها با این که از دقت بالاتری برخوردارند؛ ولی از آنجایی که برای ارائه پیشنهاد نیازمند داده‌هایی درباره کاربر جدید هستند، مشکل شروع سرد دارند.

۱.۲.۲.۲ یادگیری قانون انجمنی

یادگیری قانون انجمنی^۲ مجموعه‌ای از قانون^۳ 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^۴ استفاده می‌شود به این صورت 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

^۱non-personalized recommendations

^۲association rule learning

^۳rule

^۴confidence

معیار لیفت^۱ است که برای شناسایی قانون‌های مشتبه^۲ 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 شده است.

۲.۲.۲.۲ پالایش همکارانه

پالایش همکارانه^۳ روشی رایج برای ساخت سامانه‌های توصیه‌گر است. ورودی این سامانه‌ها، تعدادی کاربر، تعدادی قلم^۴ و سابقه بازخورد کاربران نسبت به اقلام است؛ مثلاً در مسئله مورد بررسی در این پژوهش، کاربران همان پروژه‌ها و اقلام هم کتابخانه‌ها هستند. بازخوردهای کاربرها نسبت به قلم‌ها به دو دسته بازخورد صریح^۵ و بازخورد ضمنی^۶ تقسیم می‌شوند. در مواردی که داده به شکل بازخورد صریح ارائه می‌شود کاربر بازخورد را به‌طور واضح نسبت به آن قلم اعلام کرده است مثلاً درباره آن قلم نظری نوشته یا آن را پسندیده است. ولی در بیشتر مواقع، کاربر بازخورد خود را صراحتاً اعلام نکرده ولی از سابقه فعالیت‌های کاربر می‌شود اطلاعاتی درباره بازخورد کاربر نسبت به اقلام به دست آورد؛ مثلاً استفاده یک فناوری در یک پروژه یک بازخورد ضمنی است.

۳.۲.۲.۲ مبتنی بر محتوا

در سامانه توصیه‌گر مبتنی بر محتوا^۷ برخلاف سامانه‌های پالایش همکارانه که از داده بازخورد کاربران به اقلام برای محاسبه 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 این صورت 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 بر محتوا می‌توان از محتوای پروژه‌ها و محاسبه میزان شباهت آن‌ها با هم استفاده کرد.

¹lift

²misleading

³collaborative filtering

⁴item

⁵explicit feedback

⁶implicit feedback

⁷content-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 رابطه غیرمستقیم میان آنان نیز استفاده شود. به عبارت دیگر استفاده از شبکه‌های عصبی گراف این امکان را فراهم می‌کند که هر دوی اطلاعات مرتبه پایین و اطلاعات مرتبه بالا^۱ که از گراف‌ها قابل استخراج است، در جریان ارائه پیشنهادها مورد استفاده قرار گیرد [۴].

۱.۳.۲.۲ گراف شبکه پیچشی

گراف شبکه پیچشی^۲ الگوریتمی برای انتقال بردار ویژگی‌های ورودی^۳ و هموار کردن تأثیر بردار ویژگی‌ها در یک گراف است. طرز کار این الگوریتم به این صورت است که در هر لایه از شبکه عصبی، تمام گره‌های گراف بردار ویژگی همسایگانشان را دریافت می‌کنند، این بردارها را با استفاده از یک تابع تجمع مانند میانگین، تجمع می‌کنند و سپس آن گره با یک بردار جدید در گراف بازنمایی می‌شود و در لایه بعدی شبکه عصبی همین مراحل تکرار می‌شوند با این تفاوت که ورودی هر گره، بردارهای به‌روز شده همسایگانش از لایه قبل است. درواقع تفاوت این الگوریتم با یک شبکه عصبی ساده در این است که یک مرحله پیش‌پردازش قبل از اجرای هر لایه وجود دارد که در آن، هر گره بردارهای همسایگانش را با هم تجمع می‌کند.

۲.۳.۲.۲ پالایش همکارانه گراف عصبی

الگوریتم پالایش همکارانه گراف عصبی^۴ یک الگوریتم بر پایه گراف شبکه پیچشی است که هدف آن ساخت یک سامانه 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 می‌چسبند.

¹high-order

²Graph Convolutional Network

³input feature vector

⁴Neural Graph Collaborative Filtering

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

در رابطه ۱.۲ مقدار $\mathbf{e}_u^{(k+1)}$ بردار جاساز یک کاربر u در لایه $k+1$ و $\mathbf{e}_u^{(k)}$ بردار جاساز همان کاربر در لایه قبل است. درواقع این رابطه نشان می‌دهد بردار جاساز یک کاربر در هر لایه را چگونه می‌توان از بردار جاساز آن در لایه قبل به دست آورد. σ یک تابع فعال‌ساز غیرخطی مانند سیگموئید^۱ و همچنین $\mathbf{W}_1$ و $\mathbf{W}_2$ ماتریس‌های وزن تبدیل ویژگی هستند. همچنین $\mathcal{N}_i$ و $\mathcal{N}_u$ به ترتیب برابر مجموعه اقلامی که با کاربر u تعامل داشتند و مجموعه کاربرانی که با قلم i تعامل داشتند است. رابطه 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 رابطه بالا باید با هم تغییر کند.

۳.۳.۲.۲ شبکه پیچشی گراف سبک

الگوریتم پالایش همکارانه گراف عصبی کاملاً از گراف شبکه پیچشی اقتباس شده است و برای همین تمام جزئیات آن را بدون تغییر چندانی استفاده کرده‌است درحالی‌که بسیاری از این جزئیات برای مسئله 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 پرارجع‌ترین کارهای انجام‌شده در این راستا شبکه پیچشی گراف سبک^۲ است که یک الگوریتم بر پایه پالایش همکارانه گراف عصبی است که در حوزه پالایش همکارانه برای ساخت سامانه‌های توصیه‌گر مورد استفاده قرار می‌گیرد [۵]. این الگوریتم مشابه پالایش همکارانه گراف عصبی است با این تفاوت مهم که بردارهای جاساز لایه‌های بالاتر از رابطه 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.

^۱sigmoid

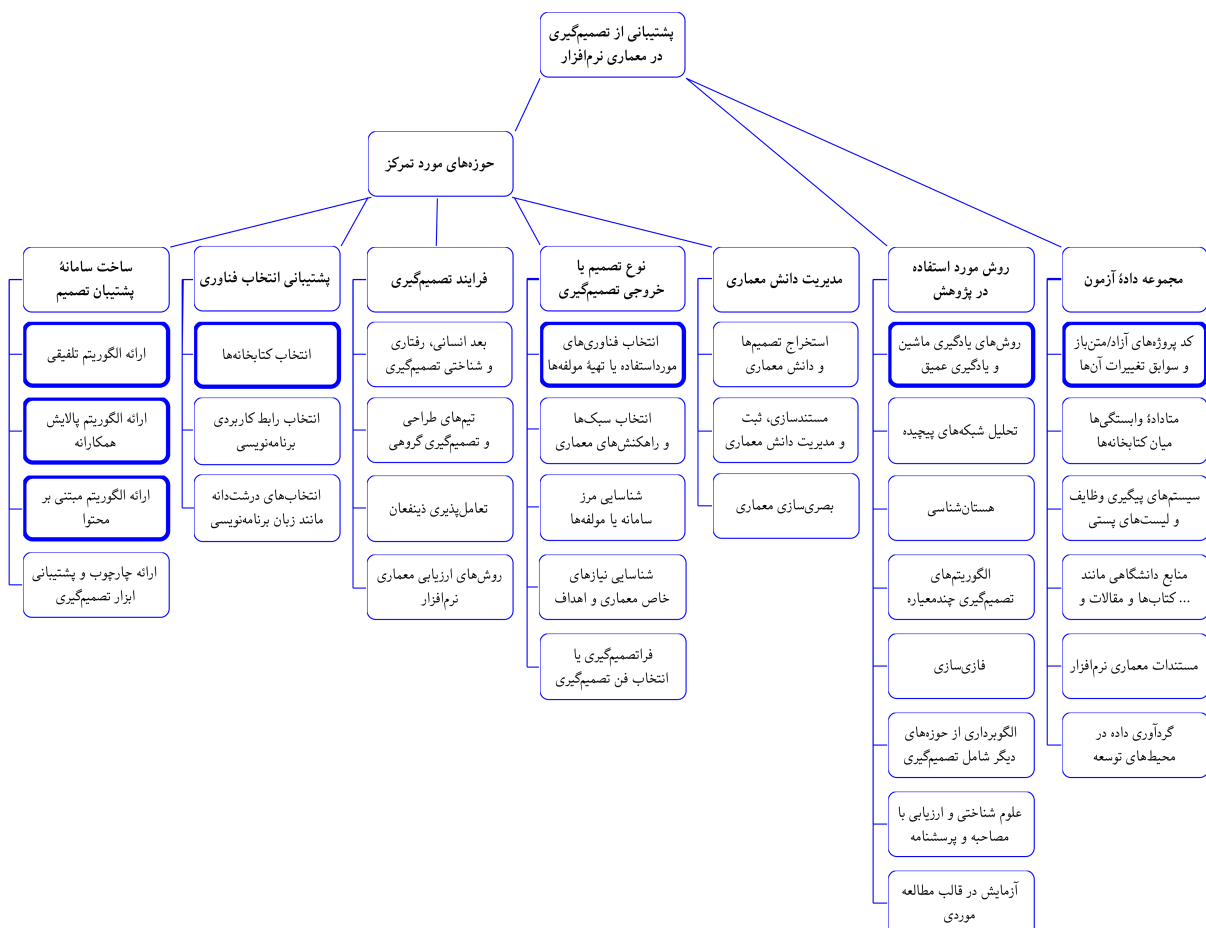
^۲Light Graph Convolution Network Filtering

فصل ۳

کارهای پیشین

پژوهش در زمینه مهندسی نرم افزار ترکیبی از پژوهش داده محور^۱ و پژوهش در علوم اجتماعی^۲ است. از یک سو برخی جنبه های توسعه نرم افزار مانند افزایش صحت یا بهینه بودن الگوریتم ها، قابلیت اثبات ریاضی دارند از سوی دیگر ساخت نرم افزار یک فعالیت اجتماعی است که اعتبارسنجی و قطعیت در آن ممکن نیست. برای همین دو الگوارۀ^۳ را می توان برای پژوهش در زمینه مهندسی نرم افزار متصور شد. الگوارۀ اول از چشم انداز علوم رفتاری^۴ است و الگوارۀ دوم از چشم انداز علوم طراحی^۵ است که در این الگوارۀ، پژوهش به عنوان یک فعالیت حل مسئله دیده می شود. به طور خاص در پژوهش های حوزه تصمیم گیری معماری نرم افزار نیز هر دو الگوارۀ قابل مشاهده است.

۱.۳ طبقه بندی کارهای پیشین



شکل ۱.۳: درخت موضوعی پژوهش ها در زمینه پشتیبانی از تصمیم گیری در معماری نرم افزار

حوزه های مورد تمرکز در پژوهش های پیشین تصمیم گیری در معماری نرم افزار، مطابق شکل ۱.۳ به چند دسته

¹data-driven
²social science
³paradigm

⁴behavioral science
⁵design science

زیر تقسیم می‌شوند:

● **ساخت سامانه پشتیبان تصمیم:** این دسته از پژوهش‌ها به جنبه‌های فنی ابزارهای یاری‌رسان تصمیم‌گیری معماری نرم‌افزار پرداخته‌اند. پژوهش‌های [۶]، [۷]، [۸]، [۹] و [۱۰] درباره مسائل مرتبط با ساخت این ابزارها هستند و یک چارچوب برای این ابزارها پیشنهاد کردند یا یک نسخه پیاده‌سازی شده از ابزار پیشنهادی خود را معرفی کرده‌اند. پژوهش [۱۱] مربوط به نحوه استفاده از این سامانه در خط تولید محصولات نرم‌افزاری^۱ است. همچنین یک دسته از پژوهش‌ها یکی از ابزارهای پیشنهاد شده را در شرایط واقعی آزمایش یا مورد مطالعه موردی^۲ قرار داده‌اند.

● **فن یا الگوریتم تصمیم‌گیری:** پژوهش‌های بسیاری درباره نحوه به کارگرفتن الگوریتم‌های تصمیم‌گیری در زمینه 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 مطالعه موردی با داده‌های واقعی مورد استفاده قرار داده‌اند و درواقع یک مثال از به کارگیری یک فن تصمیم‌گیری ارائه کردند. هریک از فنون تصمیم‌گیری شاخصه‌های به خصوصی 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، درجه دشواری به کارگیری 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^۳ 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

¹software product line

²case study

³decision-making behaviour

نیز بهره گرفته‌اند. سوگیری‌های شناختی^۱ 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 به کارگیری 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 نحوه 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 حوزه مورد بررسی در پژوهش‌های پیشین است. در پژوهش‌های [۲۱] و [۱]، [۲۲]، [۲۳] و [۲۴]، [۲۵]، [۲۶] و [۲۷] تصمیم‌گیری درباره 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 عهده معمار است.

• **مدیریت دانش^۲ معماری:** پژوهش‌های [۳۱]، [۳۲] و [۳]، [۳۳] و [۸] به جای توجه به ابزارهایی که مستقیماً به خود مرحله تصمیم‌گیری مرتبط هستند به روش‌ها و ابزارهایی پرداختند که در استخراج، مستندسازی و ثبت شدن تصمیم‌های معماری، توجیه پشت هر تصمیم و مجموعه 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:

¹cognitive bias

²knowledge management

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

● **الگوریتم‌های تصمیم‌گیری چندمعیاره:** بخشی از پژوهش‌های انجام شده در زمینه تصمیم‌گیری معماری نرم‌افزار معطوف به استفاده از روش‌های استنتاج^۱ براساس دانش خبره^۲ 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^۳ بین ویژگی‌های کیفی^۴ هم‌آورد است.

● **فازی‌سازی:** در منطق فازی به جای تخصیص مقادیر درست و غلط به متغیرها از هر عدد حقیقی در بازه 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 بازه 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. نقطه مقابل روش‌های فازی، روش‌های خشک^۵ است. تصمیم‌گیری معماری نرم‌افزار به خاطر تفاسیر متعددی که تصمیم‌گیرندگان انسانی از یک مفهوم مشترک دارند، یک مسئله غیرقطعی و نادقیق است [۳۷] [۳۸].

● **الگوبرداری از حوزه‌های دیگر شامل تصمیم‌گیری:** برای پژوهش در زمینه تصمیم‌گیری معماری نرم‌افزار می‌توان به حوزه‌های دیگری که شامل تصمیم‌گیری می‌شوند نیز مراجعه کرد؛ مثلاً نظریه‌های اقتصاد کلاسیک فرض می‌کنند که مصرف‌کننده بر مبنای 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 داده 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 رده بعد برای مجموعه داده آزمون قرار می‌گیرد. استفاده از مصاحبه نسبت به پرسشنامه در مقالات جدیدتر

¹reasoning²expert knowledge³trade-offs⁴quality attribute⁵crisp

محبوبیت بیشتری پیدا کرده است [۳۹]. برخی مستندات معماری نرم افزار در شرکت ها و سازمان ها را مورد بررسی قرار دادند مثلاً تصمیم های معماری را از روی متن آن ها استخراج کردند. برخی از پژوهش ها هم مجموعه داده آزمون را از محیط های توسعه گردآوری کردند. در این موارد استفاده از آزمایش نسبت به مطالعه موردی در سال های اخیر محبوبیت بیشتری پیدا کرده است [۳۹].

۲.۳ پژوهش های پایه برای سامانه توصیه گر کتابخانه

در بیشتر پژوهش های پایه انجام شده در زمینه پیشنهاد کتابخانه از دو ابزار پم واکر^۱ یا از دادگان اندروید آرسنال^۲ استفاده شده است.

پژوهش لیب رک اولین مقاله ای است که دقیقاً به مسئله پیشنهاد کتابخانه پرداخته است. این مقاله کدها و دادگان ورودی یا خروجی خود را منتشر نکرده است ولی بر اساس گزارش های داخل متن از زبان برنامه نویسی سی شارپ برای پیاده سازی روش خود استفاده کرده است. معیار پشتیبانی^۳ یا درصدی از پروژه ها که از کتابخانه های یک کاربر جدید را هم استفاده کردند محاسبه می کند. سپس مشخص می کند چه پروژه هایی از دقیقاً همان کتابخانه های کاربر جدید استفاده کردند و چه پروژه هایی با دقت مشخصی از کتابخانه های کاربر جدید استفاده کردند سپس معیار اعتماد محاسبه می شود و تعدادی قانون استخراج می شود. این قوانین مشخص می کند که این کتابخانه ها زیاد رخ داده که با هم در یک پروژه بیایند. در مرحله بعد قوانین انجمنی از قوانین به دست آمده در مرحله قبل استخراج می شوند. هر کدام از این قوانین مشخص می کنند که اگر از چند کتابخانه خاص استفاده کردید، احتمال زیادی دارد که به کتابخانه دیگری هم نیاز داشته باشید.

پژوهش بعدی که به حل این مسئله پرداخته، لیب کاپ است. لیب کاپ کد خود را منتشر نکرده ولی یک ابزار برای بصری سازی^۴ دادگان و نتایج خود ایجاد کرده که آن را در یک صفحه گیت هاب^۵ معرفی کرده است. پس از لیب کاپ، لیب فایندر با روشی متفاوت به این مسئله پرداخته است.

پژوهش بعدی کراس رک بوده است که از پالایش همکارانه استفاده کرده است. کراس رک با زبان برنامه نویسی

¹<https://github.com/raux/PomWalker>

²<https://github.com/malibdata/MALib-Dataset>

³support

⁴visualization

⁵<https://saiedmoh.github.io/LibCUP>

جاوا توسعه یافته و کد خود را در گیت‌هاب^۱ منتشر کرده‌است. کراس‌رک ابتدا تشابه همه پروژه‌ها با پروژه مورد نظر را با معیار شباهت کسینوسی^۲ محاسبه می‌کند. سپس از این اطلاعات برای پیدا کردن پروژه‌ها با بیشترین میزان ارتباط با پروژه مورد نظر یا همسایه‌های آن استفاده می‌کند و با استفاده از پالایش همکارانه^۳ بر اساس کتابخانه‌هایی که پروژه‌های همسایه استفاده کردند برای ارائه فهرست سی عدد پیشنهاد استفاده می‌کند. نوآوری اصلی این پژوهش این است که برای هر کتابخانه مقدار فراوانی اصطلاح - معکوس فراوانی متن^۴ را محاسبه کرده‌است که این باعث افزایش نوظهوری^۵ 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 زبان برنامه‌نویسی متلب^۶ نوشته و در گیت‌هاب^۷ منتشر کرده‌است. پژوهش بعدی اندرولیب است که هیچ کدی منتشر نکرده‌است و بر خلاف روش‌های پایه 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 کتابخانه پایتورچ^۸ استفاده می‌کند و کدش را در گیت‌هاب^۹ منتشر کرده‌است.

۳.۳ جمع‌بندی کارهای پیشین

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

^۱<https://github.com/crossminer/CrossRec>

^۲cosine similarity

^۳Collaborative Filtering

^۴tf-idf

^۵novelty

^۶MatLab

^۷<https://github.com/libseek/LibSeek>

^۸PyTorch

^۹<https://github.com/fio1982/GRec>

جدول ۱۰۳: مقایسه پژوهش‌های پایه

نام روش	سال انتشار	دادگان خام مورد استفاده، محدودیت‌های اعمال شده و اندازه نمونه مستخرج برای ارزیابی روش	الگوریتم مورد استفاده
LibRec [۴۰]	۲۰۱۳	پانصد عدد پروژه 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 میون از بین ۴۰۹۳۶ تا کتابخانه و بررسی استفاده آن‌ها در ۳۸۰۰۰ تا پروژه 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 ^۱ به نام خوشه‌بندی فاصله‌ای مبتنی بر چگالی برنامه‌ها با آلودگی ^۲ معروف به DBSCAN
LibFinder [۲۳]	۲۰۱۸	۶۰۸۳ تا کتابخانه میون در ۳۲۷۶۰ پروژه گیت‌هاب	الگوریتم ژنتیک مرتب‌سازی بدون سلطه ^۳

ادامه در صفحه بعد

^۱hierarchical clustering^۲Density-based spatial clustering of applications with noise^۳non-dominated sorting genetic algorithm

جدول ۱۰۳ - ادامه از صفحه قبل

نام روش	سال انتشار	دادگان خام مورد استفاده، محدودیت‌های اعمال شده و اندازه نمونه مستخرج برای ارزیابی روش	الگوریتم مورد استفاده
CrossRec [۲۴]	۲۰۲۰	۱۲۰۰ تا پروژه تصادفی مستخرج از طریق رابط برنامه‌نویسی کاربردی گیت‌هاب	پالایش همکاریانه مبتنی بر کاربر-کاربر با در نظر گرفتن شباهت کسینوسی و محاسبه فراوانی اصطلاح - معکوس فراوانی متن
LibSeek [۴۱]	۲۰۲۰	دادگان اندروید آرسنال ^۱ حاوی پروژه‌های سیستم عامل اندروید	پالایش همکاریانه
AndroLib [۴۲]	۲۰۲۰	دادگان ۲۷۵۲ تا برنامه اندروید مستخرج از اندروید آرسنال و امتیاز ۹۷۹ تا کتابخانه اندروید در گوگل پلی ^۲	الگوریتم ژنتیک ^۳ توصیه مبتنی بر جستجو چندهدفه ^۴
GRec [۴۳]	۲۰۲۱	۳۱۴۳۲ تا برنامه اندروید و ۷۵۲ کتابخانه و ۵۳۷۰۱۱ مورد استفاده از کتابخانه در پروژه‌ها	پالایش همکاریانه گراف عصبی

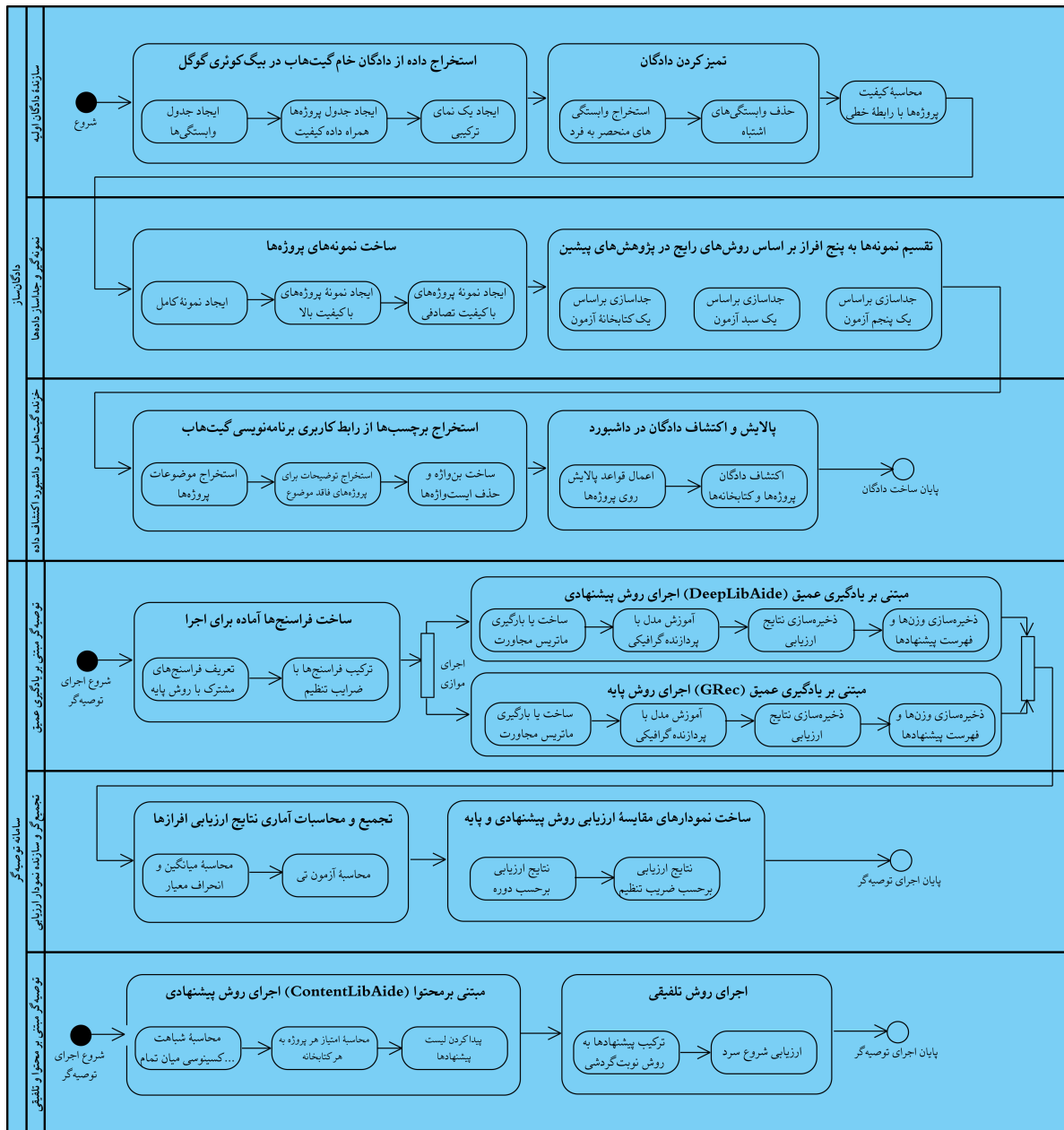
^۱Android Arsenal^۲Google Play^۳genetic algorithm^۴multi-objective recommendation

search-based

فصل ۴

روش پیشنهادی

در شکل ۱.۴ شمای کلی روش پیشنهادی در قالب نمودار فعالیت نمایش داده شده است. همان طور که در شکل مشخص شده روال کلی کار شامل دو بخش دادگان ساز و سامانه توصیه گر است. دادگان ساز ورودی های سامانه توصیه گر را تأمین می کند.



شکل ۱.۴: شمای کلی روش پیشنهادی در قالب نمودار فعالیت

ساخت دادگان به طور کلی شامل چند مرحله ساخت دادگان اولیه، ساخت نمونه و تقسیم نمونه‌ها به افرازا 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 سامانه 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 مرحله بعد نتایج به دست آمده را تجمیع و یک سری محاسبات آماری بر اساس نتایج ارزیابی افزای انجام می‌دهد. سپس نمودارهای مقایسه ارزیابی بین روش پیشنهادی و پایه را رسم می‌کند. همچنین یک روش مبتنی بر محتوا به نام ContentLibAide نیز پیشنهاد شده است 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 گرفته است. 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 نسخه داده‌های گیت‌هاب به‌روزرسانی کرد.

۱.۱.۴ دادگان‌های مورد استفاده

در بخش ۲.۱.۴ از داده کامل کامیت‌ها و کد محتوای کامیت‌ها برای بیش از سه میلیون پروژه سایت گیت‌هاب با نرخ به‌روزرسانی هفته‌ای یک‌بار^۱ استفاده شده است. این دادگان توسط خود گیت‌هاب پشتیبانی و به‌روزرسانی می‌شود. در بخش ۳.۱.۴ از دادگان تمام رویدادها^۲ در سایت گیت‌هاب با نرخ به‌روزرسانی روزانه^۳ برای استخراج داده‌های مرتبط با کیفیت پروژه‌ها استفاده شده است. همچنین رابط برنامه‌نویسی کاربردی^۴ سایت گیت‌هاب برای دریافت داده‌های موضوعات و متن توضیحات هر پروژه مورد استفاده قرار گرفته است.

^۱ <https://console.cloud.google.com/marketplace/product/github/github-repos>

^۲ event

^۳ <https://www.gharchive.org/>

^۴ API

۲.۱.۴ استخراج فناوری‌های مورد استفاده

در مخازن میون فایل‌های XML حاوی تمام داده‌های مورد نیاز برای ساختن^۱ پروژه‌های جاوا در فایل‌هایی با نامی شبیه pom.xml ذخیره می‌شوند. برای استخراج محتوای فایل‌های pom.xml در پروژه‌های زبان جاوا، پرمسما نوشته شده در ضمیمه ۴.۲.۶ تهیه و روی دادگان گیت‌هاب اجرا شد. در این پرمسما آخرین نسخه 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^۲ آن به pom.xml ختم شود پیدا شد. سپس بررسی شد که محتوای فایل دودویی^۳ نباشد. نام پروژه و مسیر فایل و محتوای آن برای پردازش‌های بعدی در یک جدول جدید ذخیره شد. در جدول خروجی این پرمسما 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 پرمسما بیان‌شده اطلاعات تمام وابستگی‌های موجود در پروژه‌ها شامل گروه^۴، مصنوع^۵، نسخه^۶ و محدوده^۷ 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 همه 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 به صورت مرتب‌شده از زیاد به کم در جدول خروجی مشخص شده‌است.
- در بسیاری از موارد نسخه 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 پرمسما بیان‌شده مقدار این نسخه‌ها با استفاده از عبارات منظم پیدا شده و در جدول خروجی قرار گرفته‌است.

¹build

²path

³binary file

⁴group

⁵artifact

⁶version

⁷scope

- 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^۱ پروژه‌ها می‌یون برای نوشتن متن شناسهٔ مصنوع و شناسهٔ سازمان^۲، الگوی عبارت منظم زیر ایجاد شد:

```
1 | ~[a-zA-Z][a-zA-Z._0-9-]+:[a-zA-Z._0-9-]+$
```

سپس با استفاده از الگوی بالا همهٔ کتابخانه‌های اشتباه استخراج شده از گیت‌هاب از دادگان حذف شدند. آخرین پژوهش پایه انجام شده در این حوزه که از داده‌های پروژه‌ها می‌یون 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:

```
1 | org.scalatest:scalatest_${scala.binary.version}
2 | ${project.groupId}:naked-objects-fixture
3 | ${groupId}:${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

¹conventions

²<https://maven.apache.org/guides/mini/guide-naming-conventions.html>

که بیشتر به اندازه پروژه اشاره دارند و داده 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 بیگ کوئری 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 ضمیمه 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 رخ داده توسط کاربران یکتا برای هر پروژه شمارش می‌شود. همچنین زبان اول هر پروژه زبانی است که بیشترین حجم بایت^۱ از محتوای پروژه به آن تخصیص یافته‌است. در مواردی گیت‌هاب هیچ زبانی را نتوانسته تشخیص دهد که در این صورت مقدار پوچ^۲ درج شده‌است. همچنین باید توجه کرد که یک کامیت ممکن است در چند پروژه مختلف باشد مثلاً در حالتی که پروژه‌ها از روی یکدیگر انشعاب شده باشند یا ممکن است نام یک پروژه در طی زمان تغییر کرده باشد. در نهایت دادگان حاوی ۱۴۳,۸۸۰ پروژه گیت‌هاب با داده هفته 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 انگیزه اصلی از استفاده از داده کیفیت و اندازه 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 این که گیت‌هاب به عنوان یک شبکه اجتماعی^۳ طراحی شده کسی خیلی از این قابلیت‌های گیت‌هاب استفاده نمی‌کند.

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

¹byte

²null

³social 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 سامانه پیگیری وظایف و ... ای که استفاده شده که جدا از گیت‌هاب است. مثال معروف آن پروژه‌های بنیاد موزیلا است که سامانه پیگیری ایرادها^۱ خودش را دارد و از گیت‌هاب پروژه‌های بنیاد موزیلا کاملاً جداست. برای همین نمی‌شود حکم کلی داد و فرض کرد که همه پروژه‌ها از قابلیت‌های درخواست واکشی^۲ و بازبینی کد^۳ و سامانه پیگیری وظایف^۴ 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 مسئله توصیه‌گر کمکی نمی‌کند.

۴.۱.۴ استخراج داده 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^۵، بن‌واژه 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^۶ برای ارائه 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 درباره 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 فیلتر شده 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

¹bug tracker system

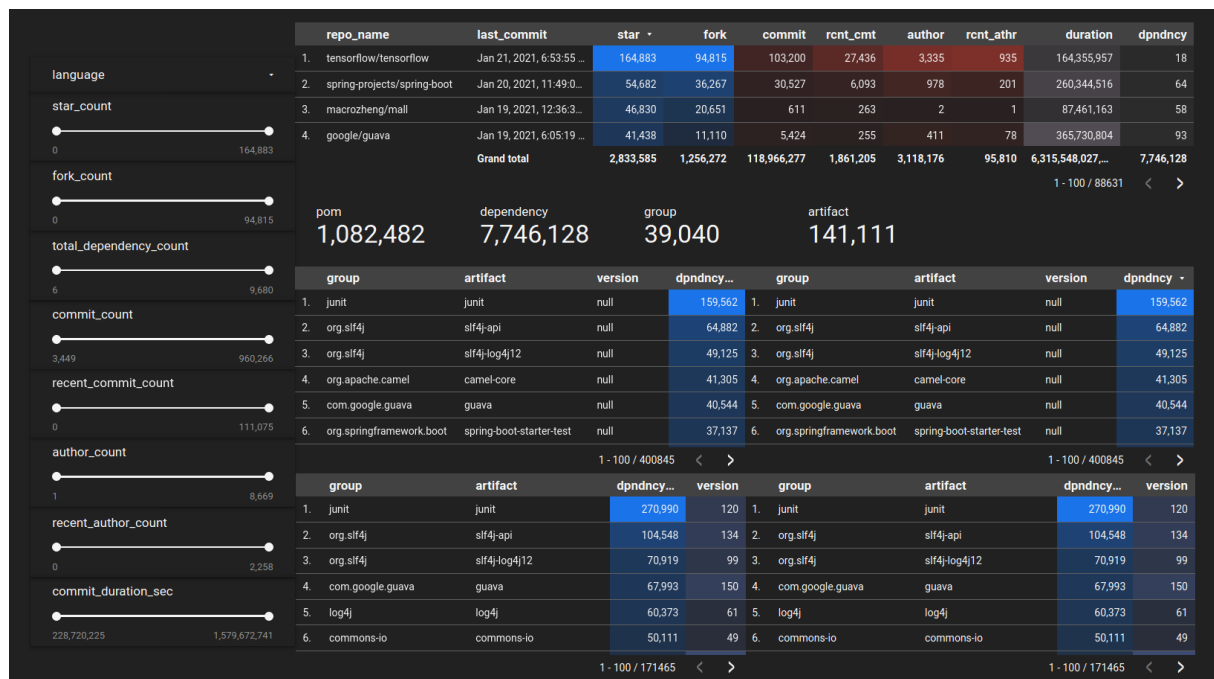
²pull request

³code review

⁴issue tracking system

⁵natural language processing

⁶content based



شکل ۲.۴: داشبورد تحلیل و مقایسه فناوری‌های به کار 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 به کار 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 ارائه سامانه توصیه گر بهبود یافته

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

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

فهرست مرتب شده از اعضای مجموعه L است که در رابطه نوشته شده در فرمول ۱.۴ با نام های $O_{DeepLibAide}$ و $O_{ContentLibAide}$ مشخص شدند. همچنین $(R_{ij})_{1 \leq i \leq n, 1 \leq j \leq m}$ بیان گر استفاده یا عدم استفاده از کتابخانه l_j در پروژه p_i در دادگان ورودی است.

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

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

$$R_{ij} = \begin{cases} 1 & \text{استفاده} \\ NULL & \text{عدم استفاده} \end{cases} \quad (1.4)$$

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

۱.۳.۴ ترکیب خطی داده های کیفیت پروژه ها

یکی از موضوعات مورد بررسی در بخش های بعدی تاثیر کیفیت پروژه های موجود در دادگان ورودی بر توصیه های 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، پروژه ها بر اساس محبوبیت و کیفیت آن ها دسته بندی شده است و می توان نتایج ارزیابی سامانه توصیه گر را برای هر کدام را با دیگری مقایسه کرد.

$$q(p, j) = \frac{c_{pj}}{\max[(c_{ij})_{1 \leq i \leq P}]} \quad (2.4)$$

اطلاعات استخراج شده درباره هر پروژه شامل تعداد ستاره، انشعاب، نویسنده، کامیت، نویسنده ها و کامیت های یک سال اخیر و مدت زمان عمر پروژه، مقیاس و واحد متفاوتی دارند مثلاً برای ستاره های اعطا شده به یک پروژه، تعداد آن ها شمارش می شود ولی برای مدت زمان عمر یک پروژه تعداد ثانیه های بازه زمانی بین اولین و آخرین کامیت در آن پروژه بررسی می شود و معمولاً تعداد کامیت ها از تعداد نویسنده های یک پروژه بسیار بیشتر است. در رابطه ۲.۴ هر کدام از این فقره های اطلاعات به صورت c_{ij} به عنوان مقدار یک ویژگی کیفی از یک پروژه در یک ماتریس پروژه

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

$$s_p = \frac{\sum_{j \in I}^Q w_j \times q(p, j)}{\sum_{j \in I}^Q w_j} \quad (3.4)$$

رابطه ۳.۴ یک میانگین وزنی از تمام اطلاعات به دست آمده درباره یک پروژه است و حاصل آن یک عدد امتیاز در بازه 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 داده پروژه‌های با کیفیت که از کتابخانه‌هایی مشابه من استفاده کرده‌اند به من کتابخانه‌هایی را پیشنهاد کن و معیار محاسبه 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 برنامه نوشته شده در این پژوهش و برنامه روش پایه استخراج شده است. این نمونه‌ها «کامل» و «با کیفیت» و «با کیفیت متوسط» نام‌گذاری شده‌اند. در نمونه کامل ۳۱۴۰۰ تا پروژه وجود دارد که با تعداد پروژه‌های جی‌رک یا بیس‌لاین فعلی دقیقاً مساوی است. در نمونه «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 شده است. 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^۱ در ادبیات حوزه سامانه‌های توصیه‌گر شامل روش «یک قلم آخر را بیرون بگذار»^۲ و روش «یک سبد را بیرون بگذار»^۳ و روش «جداسازی تقسیم زمانی مبتنی بر کاربر»^۴ استفاده شده است [۴۵]. در هر کدام از این روش‌ها، دادگان به پنج افراز^۵ تقسیم می‌شود که به این ترتیب جمعاً ۱۵ افراز وجود دارد. روش تقسیم دادگان در هر کدام

^۱cross validation

^۲leave-one-last-item-out

^۳leave-one-basket-out

^۴temporal-split-user-based

^۵fold

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

روش پایه و پیشنهادشده در این پژوهش، در هر کدام از این ۱۵ افراز برای مقایسه 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 نود بار اجرای برنامه توصیه‌گر نیاز بوده‌است. برنامه برای هر کدام از این نود افراز به‌طور جداگانه اجرا می‌شود و میانگین و انحراف معیار نتایج ارزیابی بین افرازا محاسبه می‌شود.

۱.۲.۳.۴ روش یک قلم آخر را بیرون بگذار

این روش برای جداسازی دادگان به این صورت است که در هر افراز برای هر پروژه فقط یک کتابخانه‌آزمون 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 این که تمام دادگان موجود یک‌بار به عنوان داده‌آزمون 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 این صورت 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 اندازه هر سبد برابر پنج کتابخانه در نظر گرفته شده‌است. این روش جداسازی داده‌ها در پژوهش الگوریتم شبکه پیچشی گراف سبک و پژوهش پایه 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 مبتنی بر کاربر

این روش برای جداسازی دادگان به این صورت است که در هر افراز برای هر پروژه درصد مشخصی از کتابخانه‌های آن به عنوان داده آزمون وجود دارد که از آن کتابخانه‌ها استفاده کرده‌است به این ترتیب کتابخانه‌های داده آزمون برای هر پروژه با پروژه 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 داده آزمون در نظر گرفته شده‌است. مزیت این روش این است که به تعداد افرازهای کمتری نیاز دارد و در نتیجه خیلی سریع‌تر اجرای آن تمام می‌شود و کل دادگان ورودی را هم پوشش می‌دهد یا به عبارت دیگر کل دادگان حتماً یک‌بار به عنوان داده آزمون مورد استفاده قرار می‌گیرد. اما اشکال این روش این است که نسبت به روش یک قلم را بیرون بگذار، کتابخانه‌های بیشتری را برای داده آزمون استفاده می‌کند و چون تعداد داده‌های آزمون برای هر پروژه با پروژه 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 داده آزمون، آموزش و اعتبارسنجی

به جز اعتبارسنجی متقابل برای داده‌های آزمون و آموزش برای تفکیک داده اعتبارسنجی^۱ از داده آموزش به این صورت 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 داده اعتبارسنجی سامانه کنار گذاشته می‌شود و از ۹۹ بخش دیگر برای آموزش سامانه استفاده می‌شود. برای محاسبه معیارهای ارزیابی، پیشنهادهای خروجی سامانه در نهایت با داده آزمون مقایسه می‌شود و همین مراحل برای چهار قسمت دیگر نیز تکرار و نتایج ارزیابی دوباره محاسبه می‌شود و

¹validation data

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

۴.۳.۴ بازنمایی ماتریس مجاورت داده آزمون

به طور کلی دادگان ورودی مورد استفاده در الگوریتم‌های پالایش همکارانه شامل مجموعه 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.

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix} \quad (4.4)$$

ماتریس مجاورت این گراف دو بخشی در رابطه ۴.۴ بیان شده است. 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 خانه ماتریس وجود یا عدم وجود یال بین دو راس را با یک و صفر مشخص می‌کند. این ماتریس به عنوان ورودی الگوریتم شبکه پیچشی گراف سبک برای آموزش استفاده می‌شود و به این ترتیب بردار جاساز برای یک پروژه یا یک کتابخانه در اولین لایه شبکه، سطر مربوط به آن پروژه یا کتابخانه در ماتریس مجاورت رابطه ۴.۴ است.

۵.۳.۴ آموزش مدل یادگیری عمیق

در این فاز یک شبکه پیچشی گراف سبک آموزش داده شده است که DeepLibAide نام‌گذاری شده است. علت کارایی بالای روش مورد استفاده در این پژوهش، استفاده از شبکه پیچشی گراف سبک است. این الگوریتم آخرین دستاورد موجود در حوزه کاربرد یادگیری عمیق برای ساخت سامانه‌های توصیه‌گر است. شبکه پیچشی گراف سبک به خاطر استفاده از تابع فعال‌سازی خطی^۱ و جمع‌زدن بردارهای جاساز به دست آمده از لایه‌های مختلف با

¹linear activation function

ضرایب مشخص مطابق با رابطه ۵.۴ و حذف توابع وزن عملکرد بسیار بهتری نسبت به روش پایه جی رک دارد و در صورت افزایش تعداد لایه ها کمتر دچار مشکل بیش همواری^۱ می شود. الگوریتم پالایش همکارانه گراف عصبی به خاطر چسباندن بردارهای جاساز به دست آمده از لایه های مختلف برای رسیدن به بردار جاساز نهایی، دچار مشکل بیش همواری است ولی در الگوریتم مورد استفاده در این پژوهش بردارهای جاساز با ضریب یک تقسیم بر یک به علاوه شماره لایه با هم جمع می شوند به این ترتیب علاوه بر استفاده از داده ارتباطات لایه های بالاتر میان پروژه ها و کتابخانه ها، تاثیر لایه های بالاتر نسبت به لایه های پایین تر یادگیری عمیق کاهش می یابد.

$$\mathbf{e}_u = \sum_{k=0}^K \alpha_k \mathbf{e}_u^{(k)} \quad (5.4)$$

همان طور که از مقایسه رابطه های ۱.۲ و ۲.۲ مشخص می شود، پیچیدگی های غیر ضروری طراحی الگوریتم مورد استفاده در پژوهش پایه برای پالایش گروهی باعث کاهش عملکرد و کارایی آن شده است که با حذف آن ها عملکرد بهتری حاصل می شود. روش فعلی از دو جزء ضروری تشکیل شده است. این دو جزء گراف پیچشی سبک و ترکیب لایه ها است. تابع فعال سازی غیر خطی و تبدیل بردارهای ویژگی ها با توابع وزن کنار گذاشته شده است. این ها دو عملیات استاندارد در شبکه پیچشی گراف هستند که باعث افزایش پیچیدگی مرحله آموزش مدل می شود.

الگوریتم مورد استفاده در توصیه گر DeepLibAide برای ارائه فهرست پیشنهادها، به جای چسباندن بردارهای جاساز به دست آمده از لایه های مختلف، آن ها را با ضرایب مشخصی با هم جمع می کند و همچنین با استفاده از یک تابع فعال ساز خطی و حذف بردارهای وزن، باعث سادگی مدل و در نتیجه بهبود دقت و بازیابی و سکه می شود.

۱.۵.۳.۴ تبیین روش پیشنهادی با مثال

به خاطر خطی و ساده تر بودن شبکه پیچشی گراف سبک نسبت به الگوریتم های پیش از آن، روش کار آن را راحت تر می توان در قالب یک مثال بیان کرد. پروژه ContainX/openstack4j از ۳۱ کتابخانه و پروژه google/caliper از ۲۲ کتابخانه استفاده می کنند که سه تا از آن ها به نام های زیر با هم اشتراک دارند و در کل پنجاه کتابخانه دارند:

com.google.guava:guava •

¹over-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 حاوی 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 شبکه پیچشی گراف سبک با دو لایه استفاده شود. در لایه اول یادگیری عمیق به فقط همسایه‌های مستقیم هر راس توجه می‌شود که برای پروژه‌ها، همسایه‌ها در این لایه کتابخانه‌های مورد استفاده آن‌ها و برای کتابخانه‌ها، پروژه‌هایی که از آن استفاده کردند، است. در لایه دوم یادگیری عمیق به روابط مرتبه 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 رابطه ۲.۲ از زاویه دید یک پروژه و سپس جایگزین کردن بردار جاساز لایه یک با رابطه محاسبه آن برحسب بردار لایه صفر یا ورودی مسئله در نهایت ۶.۴ به دست می‌آید [۵].

$$\mathbf{e}_p^{(2)} = \sum_{i \in \mathcal{N}_p} \frac{1}{\sqrt{|\mathcal{N}_p|} \sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(1)} = \sum_{i \in \mathcal{N}_p} \frac{1}{|\mathcal{N}_i|} \sum_{v \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_p|} \sqrt{|\mathcal{N}_v|}} \mathbf{e}_v^{(0)} \quad (6.4)$$

اگر در رابطه بالا پروژه p را `ContainX/openstack4j` در نظر بگیریم ۱ کتابخانه‌های استفاده شده در آن و v پروژه‌ای که در لایه دوم از کتابخانه‌های مشترکی با p استفاده کرده یعنی همان `google/caliper` خواهد شد. در نهایت بردار جاساز لایه دوم که حاصل در نظر گرفتن روابط مرتبه دوم بین یک پروژه با پروژه دیگر است با بردار جاساز همان پروژه در لایه قبل که حاصل در نظر گرفتن روابط مستقیم آن پروژه با همسایگانش یعنی کتابخانه‌ها است با وزن‌های مشخصی جمع می‌شود. وزن لایه اول یک تقسیم بر دو و وزن لایه دوم یک تقسیم بر سه است به این ترتیب تاثیر روابط مرتبه‌های بالاتر ضعیف‌تر می‌شود. در رابطه ۶.۴ هر کدام از مقادیر N برابر تعداد همسایه‌های مستقیم آن راس در گراف است یعنی مثلاً N_p با عدد ۳۱ جاگذاری 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 سامانه توصیه گر مبتنی بر محتوا

در کنار روش پیشنهاد شده مبتنی بر یادگیری عمیق که در بخش قبل معرفی شد، یک روش مبتنی بر محتوا به نام ContentLibAide نیز پیشنهاد شده است. روش کار این سامانه برای ارائه فهرست پیشنهاد به یک پروژه جدید به این صورت است که بر اساس برجسب های استخراج شده برای پروژه ها در بخش ۴.۱.۴ میزان شباهت کسینوسی تمام آن ها را با پروژه جدید محاسبه می کند. سپس برای محاسبه پیش بینی وضعیت یک پروژه در قبال یک کتابخانه عدد شباهت تمام پروژه هایی که آن کتابخانه را استفاده کردند را با هم جمع می کند. تعداد ده تا از کتابخانه هایی که مقدار پیش بینی شده برایشان از همه بیشتر باشد را به صورت مرتب شده در نهایت به پروژه مورد نظر پیشنهاد می کند.

در نهایت فهرست پیشنهاد های به دست آمده از روش مبتنی بر یادگیری عمیق و روش مبتنی بر محتوا به صورت نوبت گردشی^۱ با هم تلفیق^۲ می شوند. تلفیق روش یادگیری عمیق که در واقع مبتنی بر پالایش همکارانه کار می کند با یک روش مبتنی بر محتوا باعث حل شدن مشکل شروع سرد می شود. هرچند که روش ContentLibAide از نظر دقت و بازیابی پیشنهاد ها عملکرد قابل رقابتی با روش DeepLibAide ندارد ولی تلفیق این دو روش از این نظر مفید است که اگر پروژه جدیدی به سامانه مراجعه کند که قبلاً از کتابخانه ای استفاده نکرده یا از تعداد کتابخانه های کمی استفاده کرده روش مبتنی بر یادگیری عمیق امکان ارائه پیشنهاد به وی را ندارد ولی با داشتن برجسب های آن پروژه می توان از روش مبتنی بر محتوا برای ارائه پیشنهاد ها بهره جست.

۴.۴ جمع بندی روش پیشنهادی

در شکل ۳.۴ شمای کلی معماری و مولفه های تشکیل دهنده سامانه توصیه گر در قالب نمودار استقرار نمایش داده شده است. برحسب این نمودار بخش قابل توجه کدهای نوشته شده روی کولب گوگل اجرا می شوند و مولفه های آن به سرویس بیگ کوئری گوگل و دیتا استودیو گوگل نیز وابستگی دارند.

در این پژوهش با استفاده از تجمیع سه تا دادگان مختلف که با نرخ هفتگی به روزرسانی می شوند، یک دادگان از ۱۴۳۸۸۰ پروژه گیت هاب^۳ که از مدیر بسته های میون^۴ استفاده می کنند و فهرست تمام کتابخانه های مورد استفاده در

¹round-robin

²hybrid

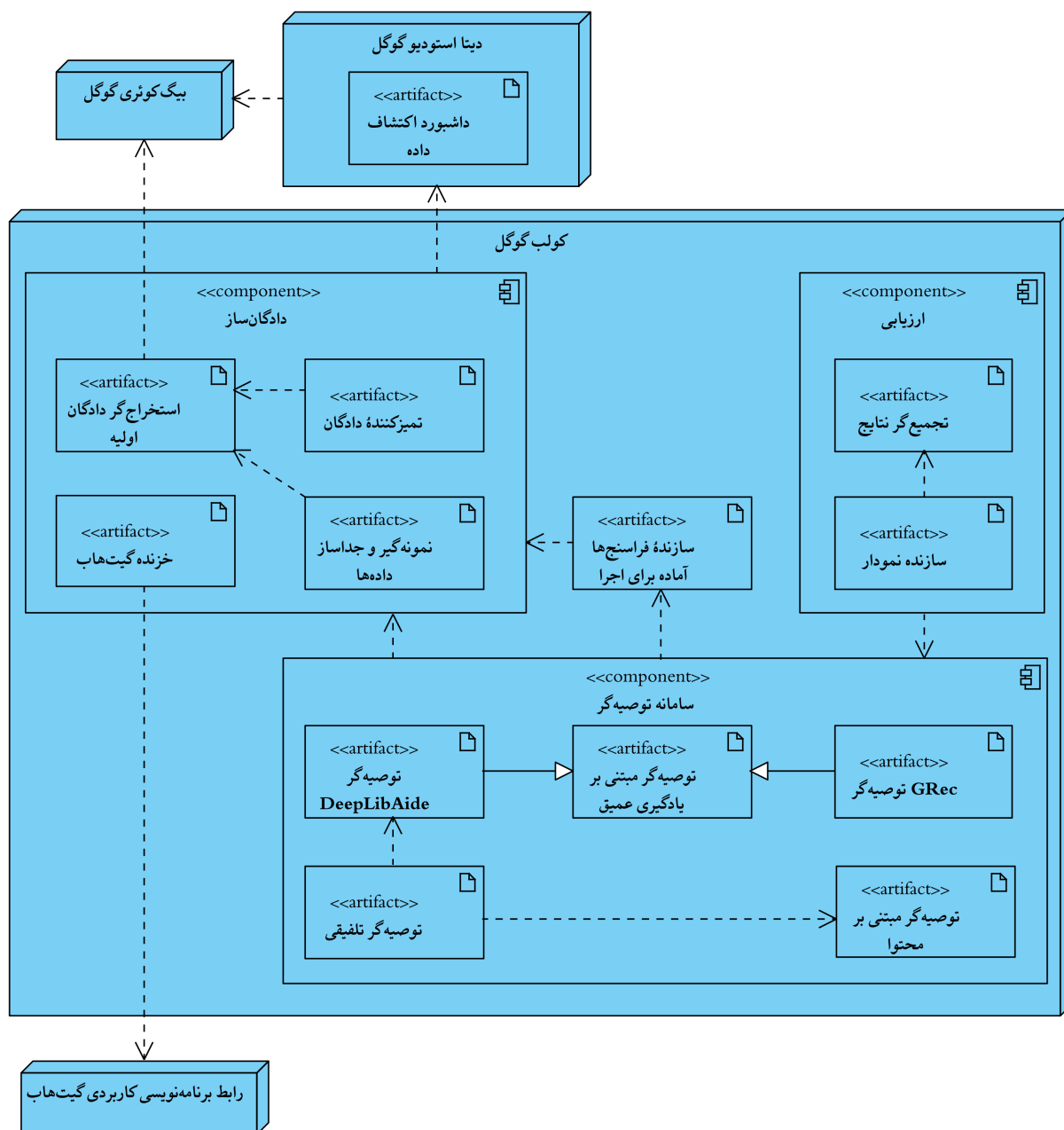
³GitHub

⁴Maven

آنها و فاصله بین اولین و آخرین کامیت، آمار تعداد کامیت، تعداد نویسنده، ستاره، انشعاب^۱، نویسندگان و کامیت‌ها در یک سال اخیر ساخته شده است. از ترکیب خطی هنجار شده آمارهای بیان شده، یک عدد به عنوان کیفیت هر پروژه به دست آمده است. یک داشبورد هم برای اکتشاف داده‌ها و اعمال فیلترها روی آنها طراحی شده است. علاوه بر این، با استفاده از واسط برنامه نویسی کاربردی^۲ گیت‌هاب تگ‌های پروژه‌ها از طریق موضوعات^۳ برچسب خورده و در صورت عدم وجود موضوعات، برچسب‌ها از طریق پردازش متن توضیحات پروژه‌ها استخراج شدند.

سپس با دو سامانه توصیه گر از داده‌های تجربی^۴ پروژه‌های نرم‌افزاری زبان برنامه نویسی جاوا^۵، برای پیشنهاد فهرست مرتب شده کتابخانه‌ها^۶ به عنوان تصمیم معماری نرم‌افزار^۷ در زمینه انتخاب فناوری استفاده شده است. سامانه اول با استفاده از یادگیری عمیق^۸ یک مدل شبکه پیچشی گراف سبک^۹ را آموزش می‌دهد و سامانه دوم که نتایجش در یک توصیه گر تلفیقی^{۱۰} با روش اول ترکیب می‌شود، یک سامانه توصیه گر مبتنی بر محتوا است. همچنین مدل‌های آموزش داده شده به طور جداگانه برای انتخاب کتابخانه توسط معمار نرم‌افزار در جریان ایجاد یا تکامل نرم‌افزار^{۱۱} می‌توانند مورد استفاده قرار گیرند.

¹fork²api³topic⁴empirical⁵Java⁶libraries⁷software architecture⁸deep learning⁹Light Graph Convolutional Network¹⁰hybrid recommender¹¹software evolution



شکل ۳.۴: شمای کلی معماری و مولفه های تشکیل دهنده سامانه توصیه گر در قالب نمودار استقرار

فصل ۵

ارزیابی

در این بخش ابتدا آمارهایی دربارهٔ سه دادگان اصلی استخراج شده بیان شده است. سپس معیارهای ارزیابی معرفی می‌شوند. توضیحاتی دربارهٔ پژوهش پایه اصلی ارائه می‌شود. سپس جزئیاتی از پیاده‌سازی مطرح شده است. 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.

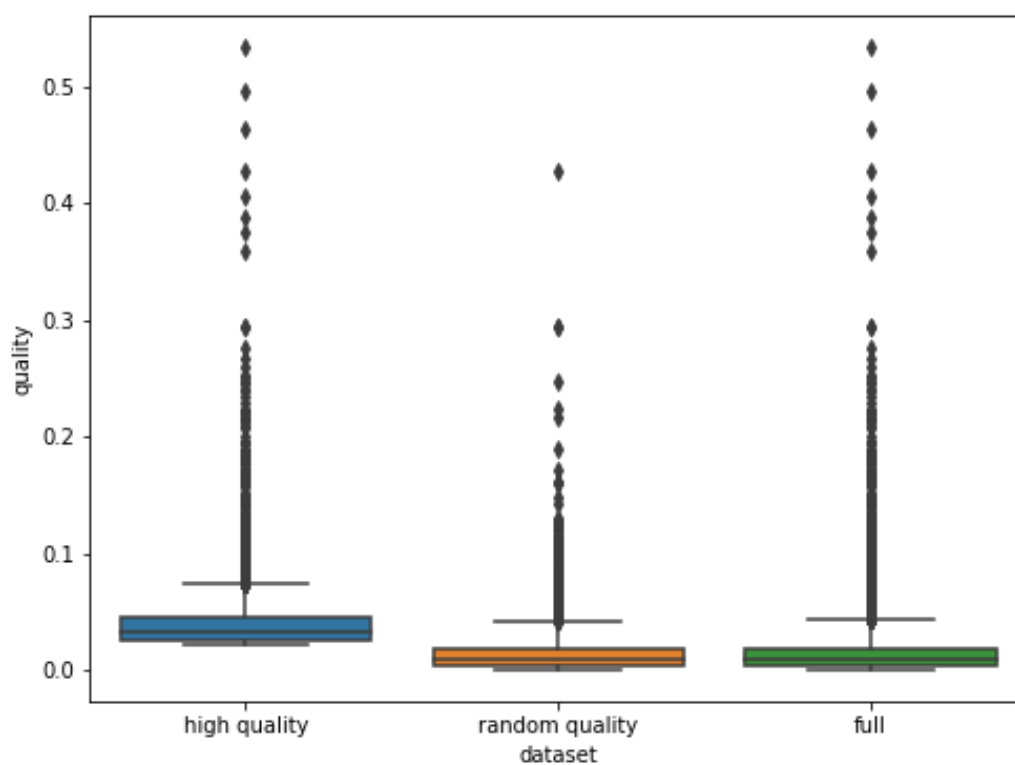
از روی این تعداد پروژه سه نمونه ساخته شده است. جزئیات مراحل ساخت نمونه‌ها در بخش ۲.۳.۴ بیان شده است. مطابق جدول ۱.۵ نمونهٔ اول که کامل‌ترین نمونه است حاوی ۳۱۴۰۰ پروژه است. نمونهٔ دوم حاوی بیست درصد پروژه‌های نمونهٔ اول است که بیشترین کیفیت را دارند بنابراین نمونهٔ دوم حاوی ۶۲۸۰ پروژه است و نمونهٔ سوم هم حاوی همان تعداد پروژه است با این تفاوت که پروژه‌های نمونهٔ سوم به صورت 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 ماهیتاً تُنک هستند.

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



شکل ۱۰۵: 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 محاسبه 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، داده موجود به دو قسمت داده آموزشی و داده آزمون تقسیم شده‌است و برای ارزیابی پیشنهادهای خروجی سامانه، با کتابخانه‌های داخل فایل آزمون مقایسه می‌شود.

۱. دقت^۱ در این جا دقت به معنای نسبت فناوری‌هایی که به درستی پیشنهادشده به کل فناوری‌های پیشنهادشده

است. این معیار با فرمول ۱۰۵ محاسبه می‌شود.

¹precision

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

۲. **بازیابی**^۱ در این جا 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.

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

۳. **سود انباشته** کاهش یافته **هنجارشده**^۲ (سکه) نشان می‌دهد تا چه اندازه پیشنهادهای درست در بالای فهرست پیشنهادها هستند. روش محاسبه سکه به این صورت 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 دست آمده را تقسیم به مجموع لگاریتم‌های جایگاه‌ها به علاوه یک می‌کنیم تا به عددی در بازه 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 کتابخانه 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^۳ زبان پایتون در زمینه هوش مصنوعی مانند دفتر جویپتر^۴ و پانداس^۵ و نامپای^۶ و کتابخانه‌های پردازش زبان طبیعی مانند ان‌ال‌تی‌کی

^۱recall

^۲Normalized Discounted Cumulative Gain

^۳ecosystem

^۴jupyter notebook

^۵pandas

^۶numpy

- Google Colab Pro+ خدمات گوگل برای اجرای برنامه‌های با پردازش سنگین گرافیکی
- Google BigQuery خدمات گوگل برای اجرای پرسمان روی حجم داده 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 گرفته شده برای مقایسه نتایج ارزیابی روش پیشنهادی با روش پایه مشخص شده است.

- **بهینه‌ساز:** از بهینه‌ساز ادم^۱ استفاده شده است. همچنین با بهینه‌ساز گرادیان کاهشی^۲ مقدار بهینه برای فراسنج ضریب تنظیم به دست آمده است.

- **نحوه ساخت ماتریس مجاورت:** روش‌های مختلف بهنجارش هنگام ساخت ماتریس مجاورت وجود دارد. در این روش از یک ماتریس مجاورت هنجارشده استفاده شده است.

- **ضریب تنظیم:** ضریب تنظیم^۳ یکی از فراسنج‌های یادگیری عمیق است که برای حل مسئله بیش‌برازش^۴ استفاده می‌شود. ضریب تنظیم برای چهار حالت مختلف در روش پیشنهادی و روش پایه بررسی و در نمودارهای بعدی مقایسه شده است.

- **توقف زودهنگام^۵:** روش پیشنهادی و روش پایه اگر در چند دوره 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 نشده است.

- **دوره^۶:** ارزیابی روش پیشنهادی نشان می‌دهد که مدل پیشنهادشده حتی برای دوره^۷های بسیار بالاتر از ۵۰ هم کارایی بهتری دارد ولی به خاطر منابع محدود پردازشی تا ۵۰ دوره نمودارها رسم شدند.

¹Adam optimizer

²gradient descent

³Regularization Coefficient

⁴over-fitting

⁵early stopping

⁶epoch

⁷Epoch

- **تعداد پیشنهاد:** تمام ارزیابی‌ها برای از یک پیشنهاد تا ده پیشنهاد اول به صورت جداگانه انجام شده است. برای تمام پیشنهادها وضعیت روش پیشنهادی نسبت به روش پایه بهتر است.
- **اندازه بردار جاساز:** اندازه بردار جاساز^۱ به طور یکسان در تمام لایه‌ها و مساوی اندازه لایه گرفته شده است چون روش پیشنهادی از تبدیل ویژگی^۲ استفاده نمی‌کند و در نهایت برای رسیدن به لیست جاسازی لایه آخر، تمام بردارهای جاساز لایه‌های مختلف را با ضرایب مختلفی با هم جمع می‌کند. اندازه بردار جاساز برابر ۳۲ در نظر گرفته شده است.
- **اندازه لایه:** اندازه لایه^۳ در شرایط کاملاً مساوی با پژوهش پایه مقایسه شده است. اندازه هر لایه ۳۲ است.
- **حذف تصادفی پیام:** حذف تصادفی پیام^۴ در شرایط کاملاً مساوی با پژوهش پایه مقایسه شده است. با توجه به این که در مدل‌های خطی، حذف تصادفی پیام معادل ضریب تنظیم مرتبه اول L2 عمل می‌کند، و این که از ضریب تنظیم برای جلوگیری از بیش‌برازش^۵ استفاده شده، این فراسنج برابر صفر در نظر گرفته شده است [۴۶] [۵].
- **نرخ یادگیری:** نرخ یادگیری^۶ 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 شده است.
- **اندازه دسته:** اندازه دسته^۷ برابر تعداد نمونه‌ها یا پروژه‌هایی است که به عنوان ورودی به مدل داده می‌شود. تعداد کل نمونه‌ها تقسیم بر اندازه دسته برابر تعداد دسته‌هایی است که در هر دوره به مدل داده می‌شود. هدف از تقسیم نمونه‌ها به دسته‌های کوچک‌تر مصرف حافظه اصلی کمتر است. در این پژوهش اندازه دسته‌ها برابر نصف تعداد نمونه‌ها در دادگان تصادفی و با کیفیت بالا در نظر گرفته شده است.

¹embedding vector size²feature transformation³Layer Size⁴Message Dropout⁵over-fitting⁶Learning Rate (lr)⁷Batch 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 افرازاها برای مقایسه دو روش قابل استناد است.

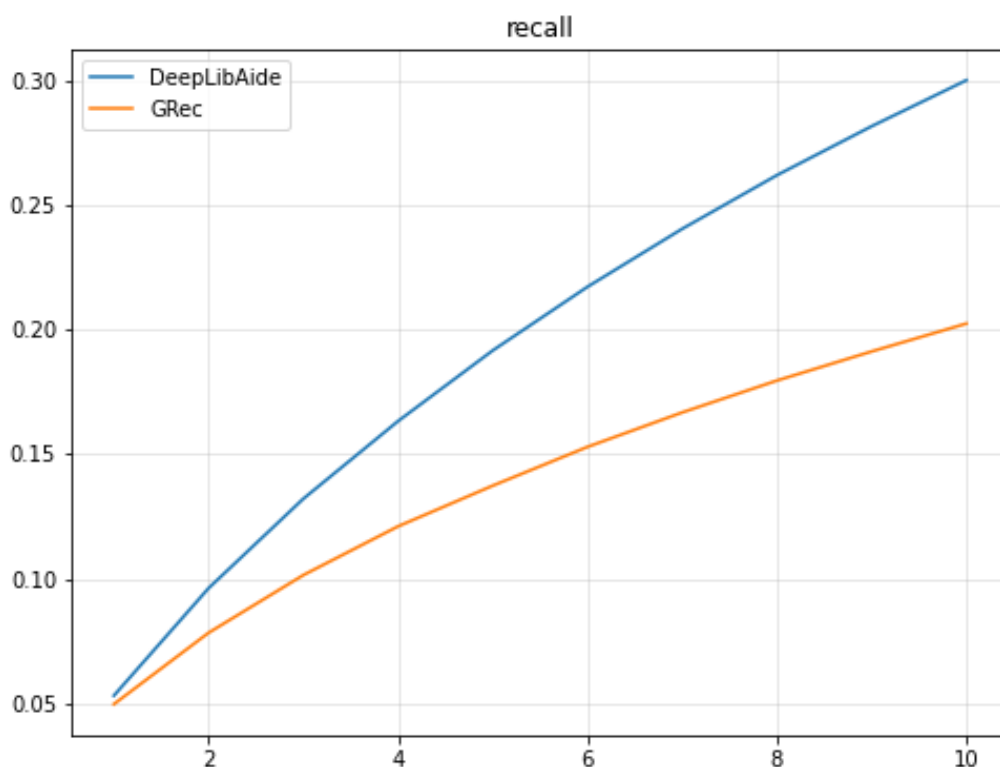
جدول ۲.۵: انحراف معیار نتایج ارزیابی در افرازاها

دادگان	تعداد کتابخانه‌های داخل آزمون	سکه 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 مبتنی بر 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 مقایسه میانگین دو روش به طور کلی

در شکل ۲.۵ میانگین میزان اختلاف بهبود بازیابی در روش پیشنهادی نسبت به روش پایه برحسب جایگاه پیشنهاد مورد بررسی قرار گرفته است. همان طور که از شکل پیداست در تعداد پیشنهادهاى بیشتر اختلاف بازیابی بهبود بیشتری نسبت به روش پایه پیدا می کند.



شکل ۲.۵: مقایسه 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 سکه مقایسه شدند. روش پیشنهادی در این پژوهش با اختلاف قابل توجهی عملکرد بهتری نسبت به روش پایه نشان داده است. همان طور که در جدول ۳.۵ مشخص است بازیابی، سکه و دقت روش پیشنهادی روی هر سه دادگان کامل، با کیفیت بالا و با کیفیت تصادفی نسبت به روش پایه بهبود داشته است.

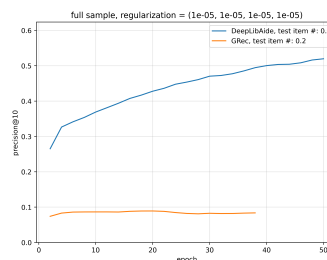
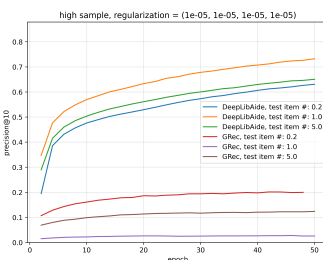
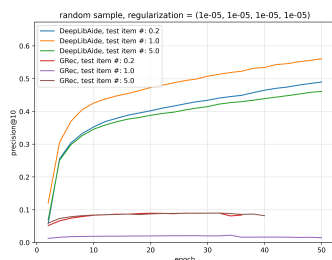
در جدول ۳.۵ روش پیشنهادی در این پژوهش که LibDeepAide نام دارد با روش پایه روی سه دادگان موجود

جدول ۳.۵: مقایسه بازیابی، سکه و دقت بین روش پیشنهادی و پایه

دادگان	نام روش	بازیابی پنج پیشنهاد اول	بازیابی ده پیشنهاد اول	سکه پنج پیشنهاد اول	سکه ده پیشنهاد اول	دقت پنج پیشنهاد اول	دقت ده پیشنهاد اول	دوره توقف زود هنگام
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، سکه و دقت مقایسه شده است. همچنین مشخص شده که هر کدام از الگوریتم‌ها هنگام اجرای روی هر یک از دادگان در کدام دوره به توقف زود هنگام برخورد کرده اند. به طور مثال الگوریتم DeepLibAide در معیار بازیابی در بازه دو درصد تا حداکثر ده درصد هم برای پنج پیشنهاد اول و هم برای ده پیشنهاد اول نسبت به روش پایه بهبود داشته است. همچنین بازیابی روش DeepLibAide همانند روش پایه در تمام دادگان‌ها، برای ده پیشنهاد اول نسبت به پنج پیشنهاد اول بهبود نه تا یازده درصدی داشته است که این نتیجه با انتظارمان مطابقت دارد زیرا افزایش تعداد پیشنهادها باعث بهبود بازیابی می شود ولی روی دقت تأثیر عکس دارد.

در روش جی رک اختلاف نتایج ارزیابی برای دادگان کامل و دادگان تصادفی که اندازه بسیار کوچکتری از دادگان کامل دارد تقریباً یکسان است ولی روش پیشنهادی برای دادگان تصادفی که اندازه کوچکتری دارد با اختلاف قابل توجهی بهتر از دادگان کامل عمل کرده است. در هر سه معیار ارزیابی دادگان باکیفیت نسبت به دادگان تصادفی و دادگان کامل بهبود یافته است فقط در مورد معیار بازیابی و در روش پیشنهادی به مقداری جزئی دادگان با کیفیت نسبت به دو دادگان دیگر افت بازیابی را نشان می دهد.

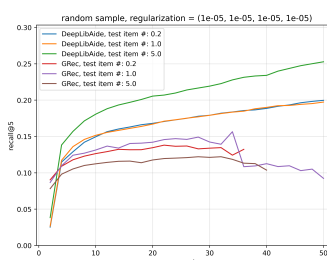
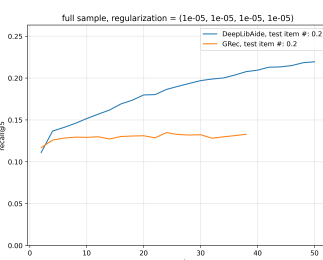
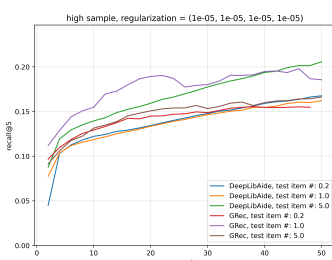


(آ) برای نمونه کامل

(ب) برای نمونه پروژه‌های با کیفیت

(ج) برای نمونه پروژه‌های تصادفی

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



(آ) برای نمونه پروژه‌های تصادفی

(ب) برای نمونه کامل

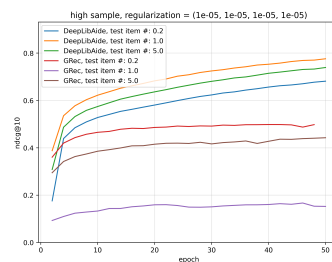
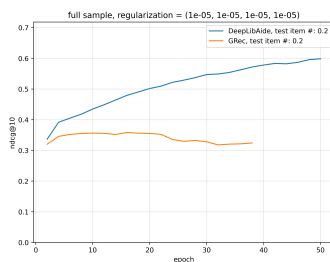
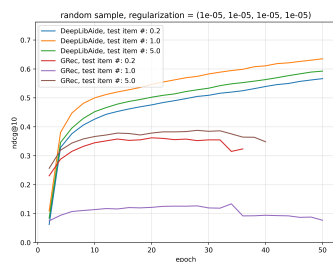
(ج) برای نمونه پروژه‌های با کیفیت

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

۲.۵.۵ مقایسه 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 شده است 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 زود هنگام می‌شود و برای همین یادگیری در آنجا متوقف می‌شود. مقایسه دقت و بازیابی برحسب دوره برای پنج پیشنهاد اول در روش پیشنهادی و روش پایه در شرایط برابر در شکل‌ها دیده می‌شود.

در شکل ۴.۵ ج 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 فزاینده‌ای رشد پیوسته‌ای را تجربه کرده است.

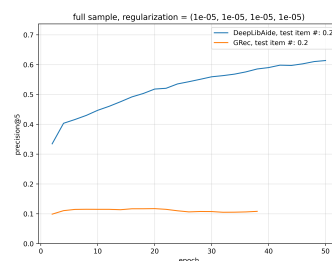
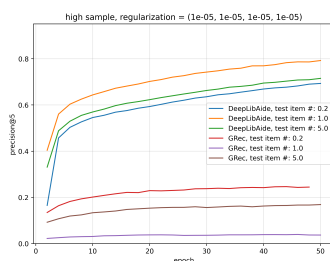
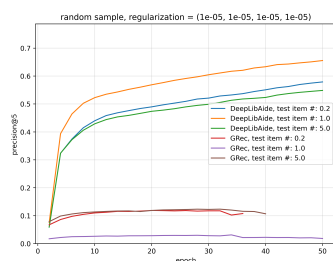


(ج) برای نمونه پروژهای تصادفی

(ب) برای نمونه کامل

(آ) برای نمونه پروژهای با کیفیت

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

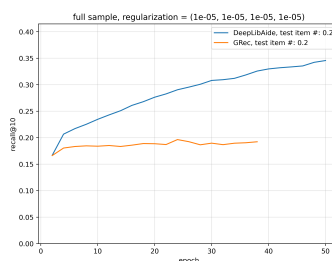
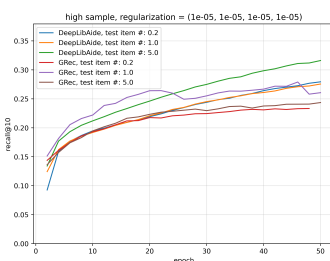
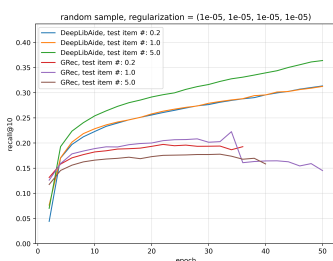


(ج) برای نمونه پروژهای تصادفی

(ب) برای نمونه پروژهای با کیفیت

(آ) برای نمونه کامل

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

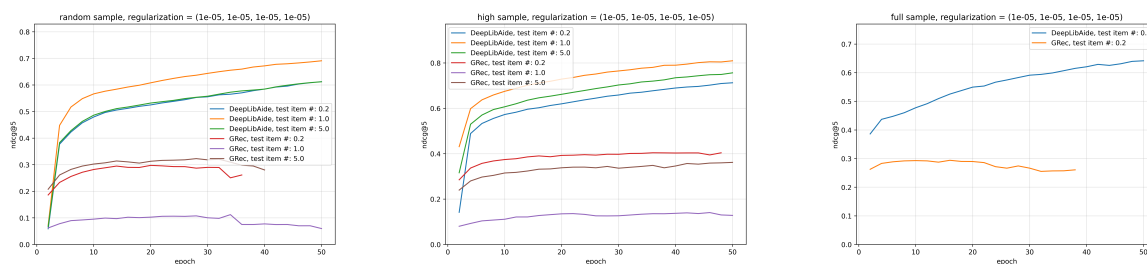


(ج) برای نمونه پروژهای تصادفی

(ب) برای نمونه پروژهای با کیفیت

(آ) برای نمونه کامل

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



(ج) برای نمونه پروژهای تصادفی

(ب) برای نمونه پروژهای با کیفیت

(آ) برای نمونه کامل

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

۳.۵.۵ ارزیابی به تفکیک تعداد داده آزمون

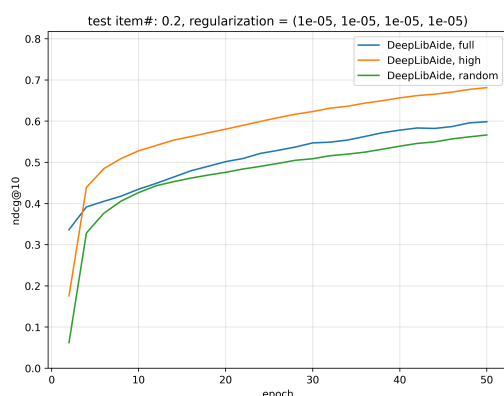
در جدول ۴.۵ نتایج ارزیابی به تفکیک تعداد کتابخانه در داده آزمون برای هر پروژه مشخص شده است. در این جدول میانگین عملکرد روش پیشنهادی و روش پایه مشخص شده است. در این شکل مشاهده می شود که روش DeepLibAide تحت همه شرایط مختلف دادگان و روش های مختلف تقسیم بندی افرازا عملکرد بهتری نسبت به روش پایه از خود نشان می دهد. برای نمونه پروژه های تصادفی ارزیابی در الگوریتم جی رک حالت پایداری ندارد ولی در روش پیشنهادی رشد پایداری با افزایش دوره ملاحظه می شود.

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

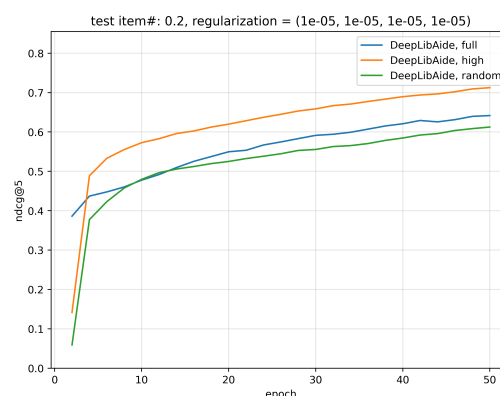
علاوه بر اختلاف قابل توجه کارایی مدل طراحی شده در تمام حالت ها نسبت به روش پایه، این مدل برای داده پروژه های با کیفیت تر گیت هاب نیز به طور جداگانه آموزش داده شده است و مقایسه نتایج شکل های ۹.۵ و ۱۰.۵ و ۱۱.۵ نشان می دهد که مدل به ترتیب برای دادگان های بسیار بزرگ (حتی با وجود پروژه های کم کیفیت) از همه بهتر و سپس داده های پروژه های با کیفیت و در آخر پروژه های کم کیفیت کارایی بهتری نشان می دهد.

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

تعداد ک.	دادگان	نام روش	تعداد کتابخانه پیشنهادی ۵			تعداد کتابخانه پیشنهادی ۱۰		
			دقت	بازیابی	سکه	دقت	بازیابی	سکه
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



(ب) برای ده پیشنهاد اول



(آ) برای پنج پیشنهاد اول

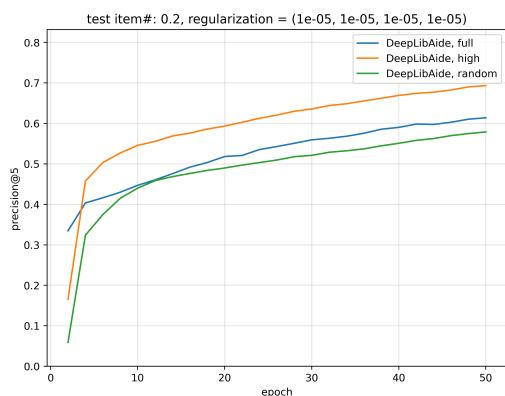
شکل ۹.۵: مقایسه سکه برحسب دوره در روش پیشنهادی شامل نمونه کامل و باکیفیت و تصادفی در شرایط برابر

۵.۵.۵ ضریب تنظیم و کاهش مسئله بیش‌برازش

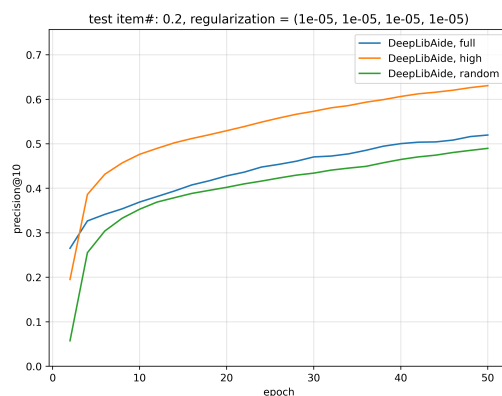
همان‌طور که در شکل‌های ۱۲.۵ دیده می‌شود افزایش ضریب تنظیم به واسطه کاهش مشکل بیش‌برازش، باعث بهبود عملکرد هر دو الگوریتم در زمینه سکه می‌شود ولی افزایش آن به مقدار ۱۰ به توان منفی پنج باعث ساده شدن بیش از حد مدل و افت عملکرد آن می‌شود.

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 داده اعتبارسنجی مستقل از کیفیت داده و حجم دادگان به‌طور مشابهی پیش می‌رود.

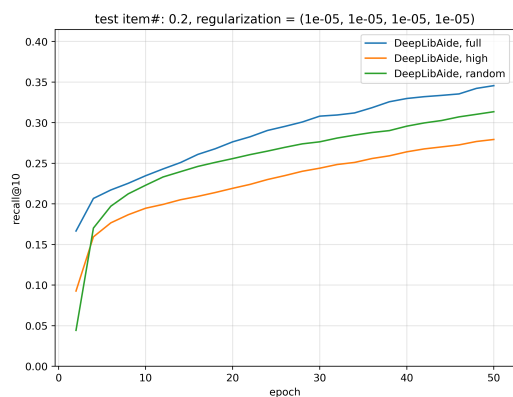


(ب) برای پنج پیشنهاد اول

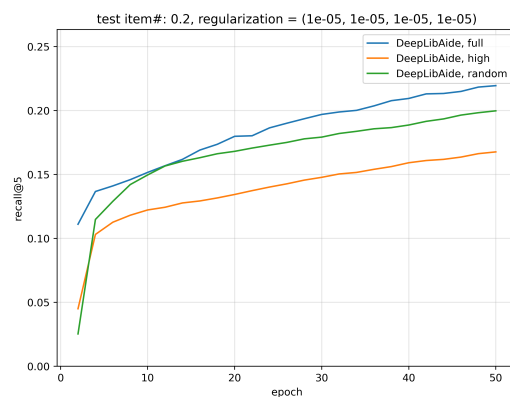


(آ) برای ده پیشنهاد اول

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

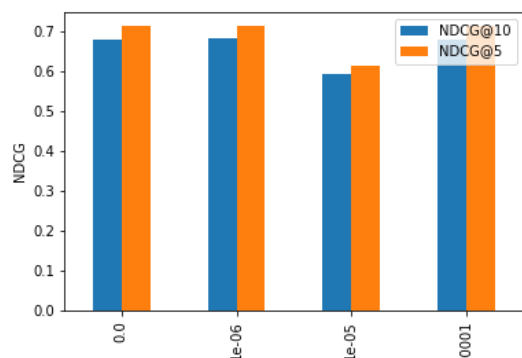


(ب) برای ده پیشنهاد اول

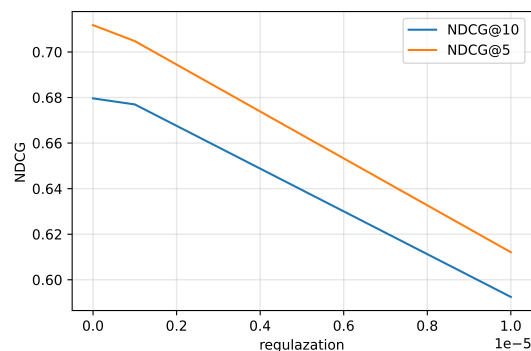


(آ) برای پنج پیشنهاد اول

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

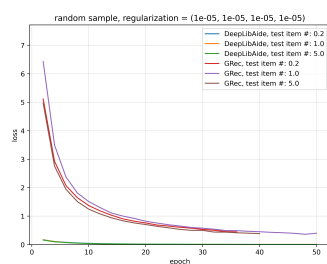


(ب) نمودار میله‌ای

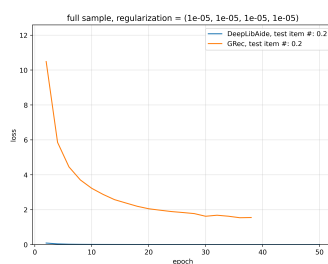


(آ) نمودار معمولی

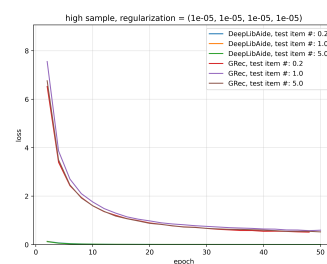
شکل ۱۲.۵: مقایسه سود انباشته کاهش یافته هنجار شده بر حسب ضریب تنظیم در روش پیشنهادی و روش پایه در شرایط برابر



(ج) برای نمونه پروژهای تصادفی



(ب) برای نمونه کامل



(آ) برای نمونه پروژهای با کیفیت

شکل ۱۳.۵: مقایسه ضرر بر حسب دوره در روش پیشنهادی و روش پایه در شرایط برابر

فصل ۶

نتیجه‌گیری و کارهای آینده

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

۱.۶ جمع‌بندی

فناوری‌های پیشنهادی توسط سامانه 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.

نواوری‌های این پژوهش به‌طور خلاصه شامل موارد زیر است:

• نحوه استخراج و ساخت دادگان ورودی

- استفاده از روشی متفاوت برای ساخت دادگان ورودی: استفاده از الگوهای عبارات منظم در پرمس‌های اس‌کیوال، باعث امکان اجرای کل خط‌لوله استخراج و به‌روزرسانی مجموعه داده در کمتر از ۳ دقیقه بر اساس آخرین دادگان رسمی گیت‌هاب می‌شود.
- استخراج داده مورد نیاز در توصیه‌گر مبتنی بر محتوا: موضوعات برچسب‌خورده به پروژه‌ها از طریق رابط کاربری برنامه‌نویسی گیت‌هاب استخراج و با استفاده از پردازش زبان طبیعی روی متن توضیحات پروژه‌ها تکمیل شد.
- محاسبه یک عدد برای سابقه و کیفیت پروژه‌ها: اطلاعاتی درباره 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^۱ پروژه‌ها می‌توان نوشتن متن شناسه مصنوع و شناسه سازمان، همه کتابخانه‌های اشتباه استخراج شده از گیت‌هاب از دادگان حذف می‌شوند.

¹conventions

– ساخت یک داشبورد تعاملی برای اکتشاف داده‌های ورودی.

• الگوریتم سامانه توصیه‌گر و بهبود عملکرد آن

– استفاده از یک الگوریتم جدیدتر مبتنی بر یادگیری عمیق: الگوریتم مورد استفاده در توصیه‌گر

DeepLibAide برای ارائه فهرست پیشنهادها، به جای چسباندن بردارهای جاساز به دست‌آمده 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 سکه می‌شود. از این‌رو این پژوهش در زمینه رویکرد

حل مسئله نوآوری دارد.

– ارائه یک سامانه توصیه‌گر تلفیقی: سامانه 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 واسطه

استفاده از یک دادگان متفاوت از دادگان استفاده کتابخانه‌ها در پروژه‌ها، باعث بهبود مسئله شروع سرد

می‌شود.

– مطالعه 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 سابقه پروژه بیانگر اندازه 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 پایه‌و اساس 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 زمینه 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^۱ 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 درباره تصمیم‌ها و اطلاعاتی پیرامون تصمیم‌ها است؛ مثلاً می‌توان خود تغییر انجام گرفته در فناوری‌های مورد استفاده در پروژه در هر کامیت را بررسی کرد و به این ترتیب دانش سطح بالای معماری مانند «مجموعه انتخاب‌های بدیل» برای هر تصمیم انتخاب فناوری را مشخص کرد زیرا براساس یک یافته تجربی که روی پروژه آپاچی اسپارک^۲ و آپاچی هدوپ^۳ 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 نمونه دیگر با بررسی متن کامیت‌ها یا متن ایراد در سامانه پیگیری وظایف و اتصال آن به کامیت‌هایی که در آن‌ها کتابخانه‌ها تغییر کردند و حاشیه‌نویسی واژگان مرتبط با معماری به کار رفته در این متن‌ها از طریق هستان‌شناسی دی‌بی‌پدیا^۴ می‌توان پایه‌و‌اساس تصمیم‌های معماری که به حذف یا اضافه یا تغییر یک فناوری در پروژه مرتبط هستند را استخراج کرد. در یک نمونه دیگر با مطالعه 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 زمینه پایه‌و‌اساس معماری قابل استخراج نیست ولی با در نظر گرفتن اطلاعات پیرامون هر

¹interpretability

²Apache Spark

³Apache Hadoop

⁴DBpedia

تصمیم در کنار خود تصمیم‌ها، بخشی از دانش معماری، با قابلیت استفاده 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 استخراج شده و ارتباط آن‌ها با ویژگی‌هایی از پروژه‌های نرم‌افزار قابل بازیابی است. در صورت استخراج دانش سطح بالای معماری، بازنمایی آن در قالب یک هستان‌شناسی و در ادامه استفاده از آن برای استدلال درباره 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 مطالعه روندها و تغییرات در تصمیمات انتخاب فناوری

نحوه 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 نسخه 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^۱ دستاوردها استفاده شده‌است ولی برای ارزیابی می‌توان از داده به‌دست آمده از یک منبع ثانویه (مثلاً stackoverflow) به عنوان داده معیار استفاده و تصمیم‌های معماری پیشنهادشده را با آن مقایسه کرد. روش‌های ارزیابی برخط مانند مصاحبه^۲، آزمایش^۳، مطالعه میدانی یا مشاهده مستقیم^۴ و پرسشنامه^۵ به ترتیب بیان‌شده در پژوهش‌های حوزه

¹offline

²interview

³experiment

⁴observation

⁵questionnaire

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

۴.۲.۶ مطالعه سایر انواع تصمیم‌های معماری

علاوه بر کتابخانه‌های مورد استفاده، می‌توان نسخه مناسب هر کتابخانه را نیز پیشنهاد داد. همچنین با بررسی مستندات نرم‌افزار، سامانه پیگیری وظایف^۱ و ... می‌توان دریافت که برای مدیریت پیکربندی از چه فناوری‌ای استفاده شده یا با بررسی کدمنبع، زبان برنامه‌نویسی و الگوهای معماری مورد استفاده را نیز می‌توان دریافت.

¹issue tracking system

- [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 stack-overflow," 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. Elshqeir, 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 Sal–ford, 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. McCreddie, 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

Layer Size	اندازه لایه	آ
Measurement	اندازه گیری	آپاچی اسپارک Apache Spark
Android Arsenal	اندروید آرسنال	آپاچی هدوپ Apache Hadoop
Fork	انشعاب	

1

Reasoning.....	استنتاج
Spring.....	اسپرینگ
Cross Validation.....	اعتبارسنجی متقابل
Confidence.....	اعتماد
Fold.....	افراز
Genetic Algorithm.....	الگوریتم ژنتیک
Paradigm.....	الگواره
Alternatives.....	انتخاب‌های بدیل
Embedding Vector Size.....	اندازه بردار جاساز
Batch Size.....	اندازه دسته
Code Review.....	بازبینی کد
Explicit Feedback.....	بازخورد صریح
Implicit Feedback.....	بازخورد ضمنی
Recall.....	بازیابی
Byte.....	بایت
Input Feature Vector.....	بردار ویژگی‌های ورودی
Offline.....	برون خطی
Visualization.....	بصری‌سازی
Adam Optimizer.....	بهینه‌ساز ادم

د

Validation Data داده اعتبارسنجی

Expert Knowledge دانش خبره

Pull Request درخواست واکنشی

Jupyter Notebook دفتر جویپتر

Precision دقت

Epoch دوره

Architectural Viewpoints دیدگاه‌های معماری

Dbpedia دی‌بی‌پدیا

ت

Linear Activation Function.. تابع فعال‌سازی خطی

Feature Transformation تبدیل ویژگی

Interpretability تفسیرپذیری

Hybrid..... تلفیق

Hybrid Recommender..... توصیه‌گر تلفیقی

Early Stopping توقف زودهنگام

Software Evolution..... تکامل نرم‌افزار

ز

Ecosystem زیست‌بوم

Java جاوا

ج

س

Software Construction ساخت نرم‌افزار

Software Structure And ساختار و معماری نرم‌افزار

Architecture

Recommender System سامانه توصیه‌گر

Bug Tracker System سامانه پیگیری ایرادها

Issue Tracking System سامانه پیگیری وظایف

Normalized سود انباشته کاهش یافته هنجار شده

Discounted Cumulative Gain

ح

Message Dropout حذف تصادفی پیام

خ

Families Of خانواده‌های برنامه‌ها و چارچوب‌ها

Programs And Frameworks

Crisp..... خشک

Hierarchical Clustering.. خوشه‌بندی سلسله‌مراتبی

Social Science	علوم اجتماعی	Cognitive Bias	سوگیری‌های شناختی
Behavioral Science	علوم رفتاری	Sigmoid	سیگموید
Design Science	علوم طراحی		

ش

		Cosine Similarity	شباهت کسینوسی
Binary File	فایل دودویی	Social Network	شبکه اجتماعی
Parameter	فراسنج	Light Graph	شبکه پیچشی گراف سبک
		Convolution Network Filtering	

ق

		Light Graph	شبکه پیچشی گراف سبک
Rule	قانون	Convolutional Network	
Conventions	قراردادها	Cold Start	شروع سرد
Item	قلم		

ص

		Formal	صوری
Lift	لیفت		

ض

		Regularization Coefficient	ضریب تنظیم
--	--	----------------------------------	------------

م

Content Based	مبتهی بر محتوا
Matlab	متلب
Scope	محدوده
Maven	مدیر بسته‌های میون
Knowledge Management	مدیریت دانش

ط

Software Design	طراحی نرم‌افزار
-----------------------	-----------------

ع

Collaborative Filtering	پالایش همکارانه	Misleading	مشتبه
Neural Graph	پالایش همکارانه گراف عصبی	Artifact	مصنوع
	Collaborative Filtering	Case Study	مطالعه موردی
Pandas	پانداس	Software Architecture	معماری نرم‌افزار
Pytorch	پایتورچ	Topics	موضوعات
Rationale	پایه‌واساس		
Natural Language Processing .	پردازش زبان طبیعی		
Support	پشتیبانی		
Null	پوچ		
Unpersonalized	پیشنهادهای غیرشخصی		
	Recommendation		

ن

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

چ

ه

Software Development . .	چرخه عمر ایجاد نرم‌افزار		
	Life Cycle		
		Hibernate	هایبرنت
		Ontology	هستان‌شناسی

ک

و

Mining Software . .	کاویدن مخازن پروژه‌های نرم‌افزار		
	Repositories		
Libraries	کتابخانه‌ها		
Quantification	کمی‌سازی		
		Api	واسط برنامه‌نویسی کاربردی
		Quality Attribute	ویژگی‌های کیفی

پ

گیت‌هاب Github

گ

گرادیان کاهشی Gradient Descent

گراف شبکه پیچشی Graph Convolutional

ی

یادگیری عمیق Deep Learning Network

یادگیری قانون انجمنی . . Association Rule Learning Group گروه

گوگل پلی Google Play

واژه‌نامه انگلیسی به فارسی

A	
Binary File فایل دودویی	Adam Optimizer بهینه‌ساز ادم
Bug Tracker System سامانهٔ پیگیری ایرادها	Alternatives انتخاب‌های بدیل
Byte بایت	Android Arsenal اندروید آرسنال
C	
Case Study مطالعهٔ موردی	Apache Hadoop آپاچی هَدوپ
Code Review بازبینی کد	Apache Spark آپاچی اسپارک
Cognitive Bias سوگیری‌های شناختی	Api واسط برنامه‌نویسی کاربردی
Cold Start شروع سرد	Architectural Viewpoints دیدگاه‌های معماری
Collaborative Filtering پالایش همکارانه	Artifact مصنوع
Confidence اعتماد	Association Rule Learning .. یادگیری قانون انجمنی
Content Based مبتنی بر محتوا	
Conventions قراردادهای	B
Cosine Similarity شباهت کسینوسی	Batch Size اندازهٔ دسته
Crisp خشک	Behavioral Science علوم رفتاری

اعتبارسنجی متقابل Cross Validation

G

الگوریتم ژنتیک Genetic Algorithm

گیت‌هاب Github

گوگل پلی Google Play

گرایان کاهشی Gradient Descent

گراف شبکه پیچشی Graph Convolutional

Network

گروه Group

H

هایبرنت Hibernate

خوشه‌بندی سلسله مراتبی .. Hierarchical Clustering

تلفیق Hybrid

توصیه گر تلفیقی Hybrid Recommender

I

بازخورد ضمنی Implicit Feedback

بردار ویژگی‌های ورودی Input Feature Vector

تفسیرپذیری Interpretability

سامانه پیگیری وظایف Issue Tracking System

قلم Item

D

دی‌بی‌پدیا Dbpedia

یادگیری عمیق Deep Learning

علوم طراحی Design Science

E

توقف زودهنگام Early Stopping

زیست‌بوم Ecosystem

اندازه 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

Message Dropout حذف تصادفی پیام

Mining Software .. کاویدن مخازن پروژه‌های نرم‌افزار

J

Repositories Java جاوا

Misleading مشتبه Jupyter Notebook دفتر جوپیتر

N

Natural Language Processing . پردازش زبان طبیعی

Neural Graph پالایش همکارانه گراف عصبی

Collaborative Filtering

Normalized سود انباشته کاهش یافته هنجار شده

Discounted Cumulative Gain

Novelty نو ظهوری

Null بوج

Numpy نامپای

K

Knowledge Management مدیریت دانش

L

Layer Size اندازه لایه

Libraries کتابخانه‌ها

Lift لیفت

Light Graph شبکه پیچشی گراف سبک

Convolution Network Filtering

Light Graph شبکه پیچشی گراف سبک

Convolutional Network

Linear Activation Function .. تابع فعال‌سازی خطی

O

Object Relation Mapping ... نگاشت رابطه به شی

Offline برون‌خطی

Ontology هستان‌شناسی

M

Matlab متلب

Maven مدیر بسته‌های میون

Measurement اندازه‌گیری

P

Social Network شبکه اجتماعی	Pandas پانداس
Social Science علوم اجتماعی	Paradigm الگوواره
Software Architecture معماری نرم‌افزار	Parameter فراسنج
Software Construction ساخت نرم‌افزار	Precision دقت
Software Design طراحی نرم‌افزار	Pull Request درخواست واکشی
Software Development چرخه عمر ایجاد نرم‌افزار	Pytorch پایتورچ
Life Cycle	

Q

Software Evolution تکامل نرم‌افزار	Quality Attribute ویژگی‌های کیفی
Software Structure And ساختار و معماری نرم‌افزار	Quantification کمی‌سازی
Architecture	
Spring اسپرینگ	
Support پشتیبانی	

R

Rationale پایه‌و‌اساس
Reasoning استنتاج
Recall بازیابی
Recommender System سامانه توصیه‌گر
Regularization Coefficient ضریب تنظیم
Rule قانون

T

Topic موضوعات

U

Non-personalized پیشنهاد‌های غیرشخصی
--

Recommendations

S

Scope محدوده
Sigmoid سیگموید

V

Validation Data داده اعتبارسنجی

بصری‌سازی Visualization

نسخه Version

ضمیمه ۱: پرسمان استخراج وابستگی‌ها

```

        _java_dependency` AS WITH `
        pom_path` AS (
15 SELECT
16   `id`,
17   `repo_name`,
18   `path`
19 FROM
20   `bigquery-public-data.`
      github_repos.files` `files`
21 WHERE
22   `files`.`path` LIKE "%pom.xml"
23 ),
24 `pom_content` AS (
25 SELECT
26   `pom_path`.`repo_name`,
27   ARRAY_AGG(
28     STRUCT(
29       `pom_path`.`path`,
30       `contents`.`content`,
31       ExtractDependencyBlock(`
          contents`.`content`) AS `
          dependency_block`

```

```

1 CREATE TEMP FUNCTION
      ExtractVersion(version STRING
      ) AS (
2   REGEXP_EXTRACT(`version`, r "\$
      {(.*)}")
3 );
4 CREATE TEMP FUNCTION
      ExtractDependencyBlock(
      content STRING) AS (
5   REGEXP_EXTRACT_ALL(
6     REGEXP_REPLACE(
7       `content`, r "<exclusions>[\s
          \S]*?</exclusions>",
8       ""
9     ),
10    r "<dependency>[\s\S]*?</
      dependency>"
11  )
12 );
13 CREATE
14 OR REPLACE TABLE `ace-resolver
      -359805.github.1

```

```

50     ARRAY(
51         SELECT
52             AS STRUCT REGEXP_EXTRACT(
53                 `dependency_block`, r "<
                    groupId>(.*?)</groupId
                    >"
54             ) `group`,
55         REGEXP_EXTRACT(
56             `dependency_block`, r "<
                    artifactId>(.*?)</
                    artifactId>"
57             ) `artifact`,
58         (
59             SELECT
60                 IF (
61                     REGEXP_CONTAINS(`
                        version`, r '\${[\w
                        .-]*?}'),
62                 (
63                     SELECT
64                         REGEXP_EXTRACT(
65                             `content`,
66                             CONCAT(
67
32     )
33     ) `pom`
34 FROM
35     `pom_path`
36 JOIN `bigquery-public-data.
    github_repos.contents` `
    contents` ON `contents`.`id
    ` = `pom_path`.`id`
37 WHERE
38     `contents`.`binary` = FALSE
39 GROUP BY
40     1
41 ),
42 `pom_dependency` AS (
43     SELECT
44         `repo_name`,
45         ARRAY_LENGTH(`pom`) `pom_count`
46         ,
47         ARRAY_AGG(
48             STRUCT(
49                 `path`,
                    ARRAY_LENGTH(`
                        dependency_block`) AS `
                        dependency_count`,

```

```

87     FROM
88         UNNEST(`dependency_block`)
89         `dependency_block`
90     ORDER BY
91         `version` DESC,
92         `group` DESC,
93         `artifact` DESC
94     ) AS `dependency`
95 ) `pom`
96 FROM
97     `pom_content`,
98     UNNEST(`pom`)
99 GROUP BY
100     1,
101     2
102 )
103 SELECT
104     `repo_name`,
105     `pom_count`,
106     SUM(`dependency_count`) `
107         total_dependency_count`,
108     ARRAY_AGG(
109         STRUCT(

```

```

68         ExtractVersion(`
69             version`),
70         ExtractVersion(`
71             version`)
72     )
73     ),
74     `version`
75 )
76 FROM
77     (
78     SELECT
79         REGEXP_EXTRACT(
80             `dependency_block`, r
81             "<version>(.*?)</
82             version>"
83         ) `version`
84     )
85     `version`,
86     REGEXP_EXTRACT(
87         `dependency_block`, r "<
88         scope>(.*?)</scope>"
89     ) `scope`

```

```
115 GROUP BY
116     1,
117     2
118 ORDER BY
119     `total_dependency_count` DESC,
120     `pom_count` DESC;
```

```
109     `path`, `dependency_count`, `
110         dependency`
111 )
112 ) `pom`
113 FROM
114     `pom_dependency`,
115     UNNEST(`pom`)
```

```

18      SELECT
19          `repo_name`,
20          `repo`.`id`,
21          `created_at`
22      FROM
23          `ace-resolver-359805`.
24              github.1
25              _java_dependency`
26      LEFT JOIN `githubarchive
27          .month.*` ON `
28              repo_name` = `repo`.`
29              name`
30      ORDER BY
31          `created_at` DESC
32      )
33      )
34      WHERE
35          `id` IS NOT NULL
36      GROUP BY
37          `id`
38      )
39      GROUP BY

```

```

1      CREATE
2      OR REPLACE TABLE `ace-resolver
3          -359805.github.2
4          _java_repository` AS WITH `
5          dependency` AS (
6      SELECT
7          `repo_name`,
8          ARRAY_AGG(`id`) [ OFFSET (0) ]
9          `id`
10     FROM
11     (
12     SELECT
13         ARRAY_AGG(`repo_name`) [
14             OFFSET (0) ] `repo_name`,
15         `id`
16     FROM
17     (

```

```

54 FROM
55     `bigquery-public-data.
        github_repos.languages`
        `t2`,
56 UNNEST(`language`)
57 WHERE
58     `t1`.`repo_name` = `t2`.`
        repo_name`
59 )
60 GROUP BY
61     1
62 ) `first_language` ON `
        languages`.`repo_name` = `
        first_language`.`repo_name`
63 ),
64 `repo_stars` AS (
65 SELECT
66     `repo`.`name` `repo_name`,
67     COUNT(DISTINCT `actor`.`id`) `
        star_count`,
68 FROM
69     `githubarchive.month.*`
70 WHERE
71     `type` = "WatchEvent"

```

```

35     `repo_name`
36 ),
37 `languages` AS (
38 SELECT
39     `languages`.`repo_name`,
40     `first_language`.`language` [
        OFFSET (0) ] `language`
41 FROM
42     `bigquery-public-data.
        github_repos.languages` `
        languages`
43 LEFT JOIN (
44 SELECT
45     `repo_name`,
46     ARRAY_AGG(`name`) `language`
47 FROM
48     `bigquery-public-data.
        github_repos.languages` `
        t1`,
49 UNNEST(`language`)
50 WHERE
51     `bytes` = (
52 SELECT
53     MAX(`bytes`)

```

```

92     MAX(`author`.`time_sec`) `
          last_commit_sec`,
93     MAX(`author`.`time_sec`) - MIN
          (`author`.`time_sec`) `
          commit_duration_sec`
94 FROM
95     `bigquery-public-data.
          github_repos.commits`,
96     UNNEST(`repo_name`) `repo_name`
          `
97 WHERE
98     `repo_name` IN (
99         SELECT
100             `repo_name`
101         FROM
102             `ace-resolver-359805.github
                  .1_java_dependency`
103     )
104 GROUP BY
105     1
106 ),
107 `repo_recent_commits` AS (
108     SELECT
109         `repo_name`,

```

```

72 GROUP BY
73     `repo`.`name`
74 ),
75 `repo_forks` AS (
76     SELECT
77         `repo`.`name` `repo_name`,
78         COUNT(DISTINCT `actor`.`id`) `
          fork_count`,
79 FROM
80     `githubarchive.month.*`
81 WHERE
82     `type` = "ForkEvent"
83 GROUP BY
84     `repo`.`name`
85 ),
86 `repo_commits` AS (
87     SELECT
88         `repo_name`,
89         COUNT(`commit`) `commit_count`
          `,
90         COUNT(DISTINCT `author`.`email`
          `) `author_count`,
91         MIN(`author`.`time_sec`) `
          first_commit_sec`,

```

```

127 )
128 SELECT
129   `languages`.`repo_name`,
130   `languages`.`language`,
131   IFNULL(`repo_stars`.`star_count`
132     `, 0) `star_count`,
133   IFNULL(`repo_forks`.`fork_count`
134     `, 0) `fork_count`,
135   `commit_count`,
136   IFNULL(`recent_commit_count`,
137     0) `recent_commit_count`,
138   `author_count`,
139   `recent_author_count`,
140   `first_commit_sec`,
141   `last_commit_sec`,
142   `commit_duration_sec`
143 FROM
144   `dependency`
145 LEFT JOIN `languages` ON `
146   dependency`.`repo_name` = `
147   languages`.`repo_name`
148 LEFT JOIN `repo_stars` ON `
149   dependency`.`repo_name` = `
150   repo_stars`.`repo_name`

```

```

110 COUNT(`commit`) `
111   recent_commit_count`,
112 COUNT(DISTINCT `author`.`email`
113   `) `recent_author_count`
114 FROM
115   `bigquery-public-data.
116   github_repos.commits`,
117 UNNEST(`repo_name`) `repo_name`
118 `
119 WHERE
120   `repo_name` IN (
121     SELECT
122       `repo_name`
123     FROM
124       `ace-resolver-359805.github
125       .1_java_dependency`
126   )
127 AND author.time_sec >=
128   UNIX_SECONDS(
129     TIMESTAMP "2020-01-01
130       00:00:00+00"
131   )
132 GROUP BY
133   1

```

```
147         ON `dependency`.`repo_name`  
148         = `repo_recent_commits`.`  
149         repo_name`  
ORDER BY  
`star_count` DESC,  
`language`;
```

```
144 LEFT JOIN `repo_forks` ON `  
dependency`.`repo_name` = `  
repo_forks`.`repo_name`  
145 JOIN `repo_commits` ON `  
dependency`.`repo_name` = `  
repo_commits`.`repo_name`  
146 LEFT JOIN `repo_recent_commits`
```

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