چکیده

آنچه که بین تمام علوم مشترک است، حجم وسیعی از اطلاعات و داده به شکل‌های مختلف است، که همراه با پیشرفت و وسعت علوم بزرگتر و ارزشمندتر می‌شود. با رشد سریع علوم کامپیوتر و استفاده از آن در چند دهه اخیر، تقریبا همه سازمان‌ها در پروژه‌های مختلف خود حجم عظیمی داده در پایگاه-های داده ذخیره می‌کنند. این سازمان‌ها و کسانی که به نوعی در پروژه‌ها سهیم هستند به فهم این داده‌ها و بهتر بگوییم کشف دانش نهفته در آن نیازمندند. این نیاز، باعث به‌وجود آمدن حوزه جدید میان رشته‌ای کشف دانش و داده‌کاوی[1] شده است، که حوزه‌های مختلف همچون پایگاه داده، آمار، یادگیری ماشین را برای کشف دانش ارزشمند نهفته در اطلاعات و داده‌ها را با هم تلفیق می‌کند.در این پژوهش به معرفی و بررسی مخازن نرم افزاری پرداخته شده است.که در فصل اول این پژوهش کلیات و تعاریف پژوهش قرارگرفته است. درفصل دوم ادبیات و مستندات پژوهش تشریح گردیده است و در فصل پایانی لزوم استفاده از تکنیکهای داده کاوی برای کشف و رفع خطاهای نرم افزار بر پایه ضریب تشابه معنایی متن و خوشه بندی بیان شده است.

[1]-Knowledge Discovery and Data Mining(KDD)

فهرست مطالب

فصل اول کلیات

فصل دوم ادبیات پژوهش

فصل اول

کلیات

۱-۱- مقدمه

آنچه که بین تمام علوم مشترک است، حجم وسیعی از اطلاعات و داده به شکل‌های مختلف است، که همراه با پیشرفت و وسعت علوم بزرگتر و ارزشمندتر می‌شود. با رشد سریع علوم کامپیوتر و استفاده از آن در چند دهه اخیر، تقریبا همه سازمان‌ها در پروژه‌های مختلف خود حجم عظیمی داده در پایگاه‌های داده ذخیره می‌کنند. این سازمان‌ها و کسانی که به نوعی در پروژه‌ها سهیم هستند به فهم این داده‌ها و بهتر بگوییم کشف دانش نهفته در آن نیازمندند. این نیاز، باعث به‌وجود آمدن حوزه جدید میان رشته‌ای کشف دانش و داده‌کاوی[1] شده است، که حوزه‌های مختلف همچون پایگاه داده، آمار، یادگیری ماشین را برای کشف دانش ارزشمند نهفته در اطلاعات و داده‌ها را با هم تلفیق می‌کند. اصطلاح Mining Data همان‌طور که از ترجمه آن به معنی داده‌کاوی مشخص می‌شود، به مفهوم استخراج اطلاعات نهان، و یا الگوها وروابط مشخص در حجم زیادی از داده‌ها در یک یا چند بانک اطلاعاتی بزرگ است.

هر نرم‌افزار در طول فرآیندتولید و پس از آن، انبوهی از اطلاعات و مستندات دارد که قابل کاوش و استفاده مفید است. این داده‌ها معمولا در پایگاه داده‌هایی به‌نام مخازن نرم‌افزاری[2] ذخیره و نگهداری می‌شوند. مخازن نرم‌افزاری نمایش دقیقی از مسیرتولید یک سیستم نرم‌افزاری ارائه می‌دهند[۱]. هدف ازکاوش مخازن نرم‌افزاری[3]MSR استفاده هوشمند از تحلیل داده‌های نهفته در آن‌ها برای کمک به تصمیم‌گیری های بهتر و سریعتر در پروژه تولید و پشتیبانی آن‌ها است. آن‌چه که در اینجا مورد توجه این تحقیق است، استخراج اطلاعات مهم برای همه ذینفعان پروژه نرم‌افزاری است. این اطلاعات از مجموعه داده‌های مرتبط با خطاهای رخ داده در طول تولید و پشتیبانی پروژه استخراج می شود.

در سال‌های گذشته مدل‌های مختلفی با استفاده از الگوریتم‌های داده‌کاوی، تشابه متن و دسته‌بندی و خوشه‌بندی داده‌ها ارائه شده. اما از آن‌جا که جستجو و استخراج اطلاعات از میان داده‌های متنی نیازمند روشی هوشمند برای تطبیق جنبه‌های معنایی و دستوری است، نیاز به مدل‌هایی که از الگوریتم‌ها معنایی استفاده‌کنند وجود دارد.در تحقیق‌های مورد مطالعه این نیاز حس می‌شود.

[1]-Knowledge Discovery and Data Mining(KDD)
[2]-Software Repositories
[3]-Mining Software Repositories

سعی بر این شده که با استفاده از الگوریتم معنایی برپایه مجموعه تشابه جملات[1] بر پایه LCS[2] [۳] و تشابه کلمات (SOC-PMI[3]) [۳]، روی مستندات ذخیره شده در مخازن خطای نرم‌افزار، مانند راه‌حل‌های ارائه شده برای خطاهای مشابه مدل‌های قبلی را تکمیل‌کرده و جوابی بهینه و سریع‌تر برای خطای پیش آمده پیدا کنیم. همچنین می‌توان زمانی تخمینی نیز برای تصحیح خطا پیش‌بینی کرد تا راهنمای تیم توسعه و ذینفعان دیگر نرم‌افزار باشد. همچنین دید بهتری نسبت به روند پیشرفت و تکامل نرم‌افزار مورد نظر ارائه شود.

۲,۱. تعریف مسئله

یکی از مراحل مهم و اساسی در مهندسی و تولید نرم‌افزار مرحله یافتن و رفع خطاهای موجود در نرم‌افزار است. این مرحله از تولید نرم‌افزار جزء وقت‌گیرترین و پرهزینه ترین مراحل به حساب می‌آید[٤]. سال‌هاست که دانش داده‌کاوی و استخراج دانش به کمک مهندسین نرم‌افزار آمده‌است. رفع خطا در فرآیند تولید بسته به مدل توسعه نرم‌افزار چندین بار انجام می‌گیرد. خطاها و مشکلات برطرف شده معمولا به روش‌های مختلف تحت عنوان مخازن خطای نرم افزار، مستندسازی و ذخیره می‌شود. این مخازن منابع عظیم دانش هستند، که کمک بزرگی در تسریع زمان تولید نرم‌افزار و پایین‌آوردن هزینه‌ها خواهدبود[٥]. روش‌هایی نیازاست که این دانش و اطلاعات مفید استخراج شود. در این تحقیق روشی برای سرعت بخشیدن به رفع‌خطای جدید با استفاده از اطلاعات موجود در مخازن خطای نرم‌افزار، ارائه شده‌است. مدل‌های زیادی تا به‌حال ارائه‌شده که یا مکمل هم بوده یا از الگوریتم‌های جدید استفاده شده‌است. مدل‌های پیشنهادی با استفاده از تشابه متن همگی از الگوریتم‌های معمولی و ساده استفاده کرده اند. در این مدل ها به این نکته مهم کمتر توجه شده که مخازن حجم بالا و پیچیده‌ای از اطلاعات را شامل می‌شوند، که بعضاً تشابه بین کلمات و معانی مختلف یک جمله نتیجه‌گیری را سخت‌تر می‌کند. پس نیاز به الگوریتم‌های معنایی در بررسی تشابه متن احساس می‌شود. همچنین باید به این نکته مهم توجه کرد که الگوریتم معنایی انتخاب شده بهینه است و قادر باشد میان این حجم اطلاعات که از سوی کاربران مختلف ثبت می‌شود، بهترین جواب با تشابه بیشتر را انتخاب کند. اگرچه تا به‌حال روش‌های بسیاری برای تکمیل مخازن خطا و استفاده از دانش نهفته در آن صورت‌گرفته اما می‌توان گفت که ضرورت اهمیت به تشابه معنایی بین داده‌ها در

[1]-String Similarity
[2]-longest common subsequence
[3]-Second-order co-occurrence pointwise mutual information

نظر گرفته نشـده اسـت. در این تحقیق سعی شده که این ضعف در جستجو و بهره‌گیری دانش نهفته در این مخازن داده پوشش داده شود.

مدل ارائه شـده در اینجا ابتدا لیسـتی از خطاهای مشابه خطای جدید با استفاده ازیک الگوریتم تشـابه معنایی مناسـب، با توجه به اطلاعات متنی ذخیره‌شـده ارائه می‌دهد. در مرحله بعد این خطاها براساس چرخه عمر خطا با استفاده از روش خوشه‌بندی K-means، خوشه بندی می‌شوند.

همچنین میانگین تشـابه هر دسـته به خطای جدید، گروه منتخب را مشـخص می کند. خطاهای موجود در این گروه، راه حل‌های پیشـنهادی برای هر کدام، کمکی برای تسهیل و تسریع در رفع خطا است و میانگین طول‌عمر گروه تخمینی بر پیچیدگی و زمان حل‌مشکل خواهد بود.

۳,۱. پیشینه تحقیقاتی

شـاید بتوان گفت که بنیان MSR و نرم‌افزارهای ردیابی‌خطا[1]با تولید GNATS[2] در سـال۱۹۹۲ نهاده شـد. پس از آن نرم‌افزارهای‌متعددی به یاری مهندسی‌نرم‌افزار شتافته‌است. در این مدت سعی و اهتمام محققان بر تکمیل مدل‌ها و موتورهای‌جسـتجو مورد استفاده در این نرم‌افزارها بوده است. در سـال ۲۰۰۰ مدلی با اسـتفاده‌از رگرسیون فازی[3] برای تخمین تعداد خطاهای موجود در نرم‌افزار ارائه شـد[٦]. در[٥] همین کار با اسـتفاده از شبکه‌های‌عصبی انجام‌شد. درسال ۲۰۰۷ LucasD.Panjer داده کاوی را برای تخمین زمان‌خطا با مقایسـه پنج الگوریتم دسـته‌بندی داده‌ها مطرح کرد[۷]. Suma.V و همکارانش به سـراغ روش‌های خوشـه‌بندی رفته و ثابت‌کردند که روش K-mean در میان روش‌های مورد بررسـی آنها بهینه‌تر بوده اسـت[۸]. CathrinWeiß و همکارانش از روش نزدیکترین همسـایه و تشـابه جملات با اسـتفاده از موتورارزیابیLucene[4] اسـتفاده کردند[۹]. Lucene خود از SVM و یک مدل بولی[5] برای جستجو و ارزیابی متون استفاده می‌کند. Nagwani نیزدرمدل‌های مختلفی از الگوریتم های داده‌کاوی متعددی برای اهداف مختلف اسـتفاده‌کرد[۱۰]،[۱۱]،[۱۲]. اما همه این روش‌ها یا به جنبه معنایی تشـابه متون و کلمات توجه نداشـتند. یا برای اسـتخراج ضـریب تشـابه دو متن از الگوریتم‌های مناسب برای این محیط بهره نبردند. الگوریتم‌های استفاده شده بیشتر به تاثیر یک متن و یافتن ضـریب تشابه آنها با متن کاربرتوجه داشتند، معمولا استاتیک بودند و هرگز به بهره‌وری روش

[1] -Bug Tracking
[2] -GNU product’s issue-tracking software
[3] -Fuzzy Regresion
[4] -www.lucene.apache.org
[5] -Boolean model

در محیط با حجم بالا توجه نداشتند. نوآوری کار تحقیقاتی ارائه‌شده نسبت به مدل‌های قبلی استفاده از یک الگوریتم بهینه با دقت بیشتر در اندازه‌گیری تشابه میان دو متن است به‌طوری‌که با حجم زیاد مستندات و کلمه‌های مشابه در مخازن نرم‌افزاری سازگار باشد. این الگوریتم تشابه جملات و تشابه معنایی بین کلمات را با هم ترکیب می‌کند. به‌طوری‌که، مشابه‌ترین جملات را، با توضیحات داده‌شده برای خطا از سوی کاربر، از میان داده‌های قبلی در مخزن استخراج کند.

٤,١. فرضیه های تحقیق

١. مخازن خطای هر نرم افزار منابع ارزشمند دانش هستند.
٢. اطلاعات ثبت شده در مخازن خطا به پیشبرد سریعتر و کم هزینه تر فرآیند تولید نرم افزار کمک می کند
٣. ثبت روش حل مشکل یا خطا در طول فرآیند تولید می تواند به توسعه دهنده پروژه برای حل خطا جدید کمک کند.
٤. برای پیش بینی قسمتی از زمان انجام پروژه و هزینه آن می توان از این اطلاعات استفاده کرد.

٥,١. اهداف تحقیق

از آنجا که در روش ارائه شده در این تحقیق سعی بر آن شده تا کاستی‌های موجود در روش‌های قبلی به نوعی پوشش داده شود ، به‌دنبال بهترین و مناسب ترین الگوریتم برای جستجو و محاسبه تشابه متون و مستندات در یک مخزن با حجم‌عظیم‌داده هستیم. به طوری‌که با بالارفتن این حجم دچار مشکل پیچیدگی‌محاسبه و بالارفتن زمان‌پاسخ‌گویی نشود. همچنین تشابه بین دو متن باید شامل تشابه‌معنایی باشد. به‌جز تشابه بین کلمات تشابه به صورت مجموعه کلمات. یعنی جملات باید بررسی شود. با توجه‌به مطالب ذکرشده یافته‌های اصلی این تحقیق را می‌شود در مطالب زیر خلاصه کرد.

١. ارائه یک روش برای کاوش مخازن خطای نرم‌افزار که کاستی‌های روش‌های قبلی در اندازه‌گیری میزان تشابه بین نمونه و خطا را پوشش‌دهد.
٢. یافتن خطاهای مشابه با خطای جدیدواستفاده از آنها برای تخمین زمان لازم برای رفع‌خطا
٣. تعیین میزان پیچیدگی خطای جدیدبا محاسبه میانگین طول عمر خطاهایی با شباهت بیشتر
٤. استفاده ازراه‌کارهای رفع خطا با جستجوی خطاهایی با ضریب تشابه بالاتر با خطای جدید

هدف نهایی کاهش هزینه‌های مالی پروژه‌های نرم‌افزاری و تسریع درفرآیند مهندسی‌نرم‌افزار است.

٦,١. نوآوری تحقیق

آنچه بیش از همه به‌عنوان نوآوری در کار و بهبود روش‌های گذشته مطرح است:

۱. استفاده از یک الگوریتم معنایی مناسب به‌همراه تشابه خطی در مرحله اول

۲. استفاده از الگوریتم‌های خوشه‌بندی در مرحله دوم.

در این تحقیق برای بالا بردن دقت و سرعت در استفاده از اطلاعات از الگوریتم پیشرفته تر نسبت به کارهای قبلی استفاده شده است. الگوریتم ترکیبی که به جنبه معنایی تشابه نیز توجه داشته باشد. همچنین نتایج نهایی برای بررسی راحت تر و دقیق تر با استفاده از یک روش بهینه خوشه بندی طبقه بندی میشوند، تا دسترسی کاربر به اطلاعات نهایی راحتر و سریعتر باشد. به بیانی ساده تر اطلاعات نهایی خوانا وجامع باشند.

۷,۱. ساختار پایان نامه

این تحقیق شامل ۵ فصل است که ترتیب فصول و موضوعات مطرح شده به‌صورت کلی در زیر بیان شده‌است:

۱,۷,۱. فصل اول: کلیات(همین فصل)

در این فصل یک مقدمه از حوزه این تحقیق ارائه شده و در آن مشخص‌شده که هدف از این تحقیق چیست و خواهان پاسخ‌گویی به چه سوالی هستیم. در ادامه همین فصل یک توضیح مختصر از تحقیقات انجام‌شده در مورد موضوع این تحقیق ارائه شده است. در نهایت نیز یک دید کلی از فصل‌های مختلف این تحقیق ذکر شده‌است.

۲,۷,۱ .فصل دوم: ادبیات تحقیق

در این فصل به بیان مفاهیم و مقدماتی که این پایان‌نامه بر اساس آن مطرح شده‌است، پرداخته می‌شود. مخازن خطای نرم‌افزار و انواع آن‌ها، چرخه عمر یک خطا، همچنین انواع مستندات مرتبط با یک خطا بررسی‌می‌شود. پس‌از آن استراتژی‌های اندازه‌گیری تشابه معنایی بین متون و تفاوت آن‌ها مورد مطالعه قرار می‌گیرد.

۳,۷,۱. فصل سوم : استفاده از تکنیک‌های داده‌کاوی برای کشف و رفع خطاهای نرم‌افزار بر پایه ضریب تشابه معنایی متن و خوشه‌بندی

ضـریب تشـابه رشـته ای یا خطی بر پایه LCS ، روش SOC-PMI به‌عنوان یک روش معنایی و تشـابه جملات بر اسـاس عبارات مشـترک معرفی‌می شـوند. و در نهایت روش اندازه‌گیری شـباهت ترکیبی با ادغام و بهینه‌سـازی این روش‌ها معرفی می‌شـود. روش‌های خوشه‌بندی و محاسبه میانگین داده‌ها مورد نیاز این تحقیق هم بیان می‌شـوند. روش پیشـنهادی با استفاده‌از استراتژی‌ها و روش‌های انتخاب‌شـده که با محیط داده‌ای مخازن سـازگار هسـتند، برای یافتن خطای مشابه‌تر با خطای جدید تشریح می‌شود و توضیح‌می‌دهیم که چگونه با یک خوشه‌بندی و محاسبه می‌توانیم میزان پیچیدگی و تخمینی برای زمان رفع‌خطا ارائه‌کرد.

٤,٧,١.فصل چهارم: اعتبارسنجی و توسعه

پیاده سـازی روش بر روی نمونه داده های واقعی که از مخزن خطا اسـتخراج شده با استفاده از برنامه‌نویسـی و نرم‌افزار Rapid miner[1]، مقایسـه روش از نظر زمان و دقت پاسـخ‌گویی با روش‌های دیگر. همچنین روش ارائه شده را با روشهای قبلی مورد مقایسه و تحلیل موضوعی قرار می‌دهیم.

۵,۷,۱. فصل پنجم : نتیجه گیری

در این فصـل به جمع‌بندی، نتیجه‌گیری و تبیین دسـتاوردها و کارهای آینده پرداخته می‌شـود. ویژگی‌های این روش با روش‌های دیگر مقایسـه شـده و زمینه‌های بهبود و نتایج به صـورت موردی بیان می‌شـود. از آنجا که اسـتفاده از تشـابه معنایی در این زمینه در قدم‌های اولیه قرار دارد، در آینده زمینه‌های زیادی برای کار بیشتر وجود دارد که به چند مورد در این فصل اشاره خواهد شد.

[1]-www.apidminer.com

فصل دوم

ادبیات تحقیق

۱,۲. فرآیند مهندسی نرم‌افزار

به زبان ساده فرآیند تولید نرم‌افزار به مراحل مختلف برای تولید و توسعه یک محصول نرم افزاری انجام می‌شود اشاره دارد. به صورت کلی فرآیند مهندسی نرم افزار شامل سه مرحله اصلی است: برنامه ریزی (امکان سنجی)، پیاده سازی آزمایش، تست و مستندسازی، استقرار ونگهداری سیستم. آن‌چه که مد نظر ماست قسمتی از مرحله پیاده‌سازی، آزمایش، تست و مستند سازی است.

۱,۱,۲.پیاده سازی ،آزمایش ،تست و مستند سازی

این مرحله در مهندسی نرم افزار بر حسب نوع مدل فرآیند مهندسی شامل قسمت‌های مختلفی است. در واقع این قسمت تولید نرم افزار است، ساده بگوییم این قسمت برنامه نویسی است. پس از آن نرم افزار باید تست شود. بسیاری از پروژه‌های بزرگ نرم‌افزاری مخصوصا پروژه‌های متن باز در چندین مرحله نسخه‌های یک نرم افزار را کامل تست می‌کنند. این کار در شرایط مختلفی انجام می‌شود که درباره آن توضیح خواهیم داد.

در هر صورت مشکلات نرم‌افزار باید شناسایی و رفع شوند. مستندسازی نیز در تمام مراحل تولید باید انجام شود. طراحی داخلی نرم‌افزار برای تعیین اهداف سیستم، نگهداری آینده و ارتقاء و بهبود سیستم هرچند پروژه پایان یافته باشد انجام می شود. همچنین ممکن است این مستندسازی شامل نوشتن ساختار تکه‌های برنامه، ظاهر برنامه کاربردی داخلی و خارجی هم باشند. این مطلب خیلی مهم است که همه چیز پروژه مستندسازی شود. این مرحله از تولید نرم‌افزار موضوع تحقیق و راه‌کار ارائه‌شده است. بالا بردن بهره‌وری و پایین‌آوردن زمان انجام این مرحله از اهداف اصلی این تحقیق هستند.

اگر اهداف مهندسی نرم‌افزار را موارد زیر در نظر بگیریم. این تحقیق را رسیدن به همه این اهداف را تسهیل می‌کند.

- افزایش کیفیت، قابلیت اطمینان، قابلیت نگهداری
- رضایت کاربران و سهامداران
- کاهش هزینه‌های جانبی و پشتیبانی
- تحویل به موقع
- استفاده از مولفه‌های استاندارد
- استفاده مجدد

مرحله مستندسازی یا به روایتی ثبت تمامی اطلاعات برآمده از پروژه، ارتباط تنگاتنگی با مرحله تست یا رفع خطا دارد. همچنین هر دو این مراحل وابسته به این تحقیق هستند. اینکه چگونه می‌توان از اطلاعات ذخیره شده به شکل‌های مختلف در طول عمر پروژه برای پیشبرد مرحله تست استفاده کرد، نیازمند راه‌کارهایی‌هوشمند در حوزه‌داده‌کاوی است. قبل‌از هرچیز باید بدانیم که در یک پروژه نرم‌افزاری چگونه مستندات و اطلاعات متنی ذخیره می‌شود. منظور از خطا در این تحقیق چیست؟ در ادامه انواع مخازن داده و اطلاعات یک پروژه نرم‌افزاری معرفی می‌شود.

۲,۲. انواع مخازن داده

۱,۲,۲.کد اصلی:

کد اصلی بخش قابل اجرا و رفتار یک توسعه نرم‌افزاری است. که در نهایت به صورت فرمت اجرایی به مشتری تحویل داده می‌شود. که عموما به عنوان مهمترین داده از سوی توسعه‌دهندگان مورد توجه قرار می‌گیرد. مخزن حاوی این اطلاعات شامل تعدادی از منابع کد و اسناد در یک یا چند زبان مختلف برنامه‌نویسی است. این اسناد معمولا به موجودیت‌های منطقی به نام‌های ماژول[1]یا بسته گروه‌بندی می‌شوند. تمام این مجموعه اطلاعات کد اصلی سیستم نامیده می‌شود. برای کاوش این متون تمرکز روی شناسه‌ها (متغیر، نام)، توضیحات و رشته‌های اصلی داخل کد اصلی است. معمولا کلمات کلیدی و نمادها در نظر گرفته نمی‌شوند.

۲,۲,۲. مخازن خطا(سیستم ردیابی خطا BTS[2])

این مخازن برای ذخیره اطلاعات مربوط به ایجاد و حل خطا، مشخصات ارتقاء سیستم و کلیه اقدامات دیگر در مرحله تعمیر و نگهداری استفاده می‌شوند. معمولا هنگامی‌که توسعه‌دهندگان و کاربران به مشکل یا خطایی در یک سیستم نرم‌افزاری مواجه می‌شوند، یادداشتی درباره این خطا در پایگاه داده خطا در موضوع مربوطه ذخیره می‌شود. این اطلاعات شامل: علت و مکان وقوع خطا در برنامه و اینکه چگونه خطا باعث ایجاد اشکال و خلل در روند برنامه شده است. پس از آن یک یا چند متخصص، موضوع ایجاد شده را برای رفع مشکل بررسی می‌کنند. چنانچه خطا برطرف شود موضوع در فرم مربوطه بسته می‌شود. تمام این اطلاعات در مخازن و پایگاه‌های خطا ذخیره می‌شوند. عمومی‌ترین سیستم‌های مخازن خطا Bugzilla ، Trac هستند.

[1]-Module
[2] -bug-tracking system

اگرچه تا به امروز سیستم‌های متعددی ساخته شده‌اند. در حالت عادی بین خطا[1]، نقص[2]، عیب[3] تفاوت قائل می‌شویم، اما در این تحقیق همه را با لفظ خطا و هم معنی در نظر می‌گیریم.

۳,۲,۲. لیست نامه‌ها و گفتگوهای ثبت شده

لیست ایمیل‌ها (یا آرشیو بحث‌ها) همراه با گفتگوهای ثبت شده بین افراد دخیل در یک پروژه آرشیوی از ارتباطات متنی توسعه‌دهندگان ، مدیران و ذینفعان آن پروژه هستند. لیست متنی متشکل از بسته‌های الکترونیکی که شامل سه قسمت:

۱. سرآیند (فرستنده ،گیرنده و زمان ارسال)
۲. بدنه پیغام(متن داخل ایمیل)
۳. مجموعه‌ای‌از فایل‌های پیوست‌شده(مستندات اضافی که همراه ایمیل فرستاده می‌شود) می‌باشد.

شرح گفتگوها شامل ثبت مکالمات فوری بین ذینفعان پروژه، که بر حسب زمان یا نویسنده دسته‌بندی شده‌اند، می‌باشد.

٤,٢,٢. پایگاه داده کنترل منبع (پایگاه داده کنترل ویرایش ها)

سیستمی برای ثبت تاریخ تغییرات (ویرایش‌ها) به‌همراه خود ویرایش و اطلاعات دیگر به صورت اسناد و اطلاعات متنی است. توسعه‌دهندگان معمولا تاریخ و زمان ویرایش یک کد اصلی را در پایگاه داده‌هایی ذخیره می‌کنند. پایگاه داده‌های کنترل کد رایج مانند cvs [۱۳] و svn [١٤] ، به توسعه دهندگان اجازه می‌دهند به یک کپی از مخزن سراسری و جهانی، در سیستم فایل‌های محلی خود، دسترسی داشته باشند. اسناد موجود را ویرایش کند، یا اطلاعاتی اضافه یا کم کند و یا ساختار دایرکتوری این مخازن را تغییر دهند. همچنین می‌تواند در مخزن اصلی سند یا اطلاعات جدید محلی ایجاد کند.

بنابراین کنترل بازبینی‌ها دو نتیجه مهم در بر خواهد داشت:

- اول اینکه به توسعه‌دهندگان اجازه می‌دهد، مستقل از کسانی که به مخازن دسترسی دارند، فایل‌های روی سیستم‌های خود را تغییر دهند . پس از آن که تغییرات تایید شده ایجاد شد بقیه می‌توانند این تغیرات را بررسی‌کنند. این استقلال کاری اجازه می‌دهد که یک چرخه کار موازی بدون نیاز به ارسال ایمیل و گفتگو و نیز بدون تغیرات ورژن برنامه به عقب و جلو تشکیل شود.

[1] -bug
[2] -defect
[3] -fault

- دوم اینکه زمان و تاریخ همه اطلاعات و مستندات به صورت خودکار ثبت و نگهداری می‌شود. اگر نسخه‌های قبل نرم‌افزار نیاز بود، توسعه‌دهندگان به‌راحتی می‌توانند به نسخه‌های قبل سیستم دسترسی داشته باشند و سیستم را به نسخه قبلی برگردانند.

۵,۲,۲. اطلاعات طراحی و نیازمندی‌های سیستم

مستندات نیازمندی‌ها، معمولا در ارتباط با مشتری و یا با تاییدهای او تنظیم می‌شود. این اسناد لیستی از نیازهای مشتری است که خواهان انجام آن توسط سیستم است. این نیازها به دو صورت دسته‌بندی می‌شوند. اینکه چه نیازهایی را سیستم باید برطرف کند و چگونه و با چه کیفیتی موردانتظار مشتری است. اطلاعات طراحی نیز شامل تمام اطلاعات مربوط به طراحی‌معماری و الگوریتم‌های مهم و مورد استفاده[1] سیستم است. طراحی سیستم می‌تواند به شکل نمودار(مانند UML) و یا به‌صورت متون جریان کار نمایش داده شوند.

٦,۲,۲. مخازن شرح اجرا

اطلاعات مربوط به خروجی یک برنامه در حین یک یا چند بار اجرا، بر حسب آن چه که در قسمت تست مشخص شده ثبت می‌شود. این اطلاعات شامل لیستی از زمان و دلیل اجرا شدن متدهای یک سیستم، مقادیر قطعی متغیرها و جزئیات مربوط به شرایط اجراست. این لیست هنگامی مفید است که فرآیند اشکال‌زدایی در مقیاس وسیع همزمان با هزاران یا میلیون‌ها فرآیند دیگر از سوی افراد مختلف انجام شود. در این هنگام پیدا کردن اشکالات فردی لازم اما کاری دشوار و زمان بر است. این مخازن کمک می‌کنند به تک تک این اجراها با جزئیات دسترسی داشته باشیم.

۷,۲,۲. مخازن سیستم نرم افزار

مخازن سیستم نرم‌افزار شامل مجموعه‌ای از سیستم‌های مختلف به همراه کد اصلی آنها که افراد علاقه‌مند می‌توانند به‌راحتی در آن جستجو کرده و از آنها استفاده نمایند. از مخازن مهم رایج در این حوزه می‌توان به Source Forge [۱۵] وGoogle Code [۱٦] اشاره کرد.

این مخازن شامل آرایه وسیعی از اطلاعات در حوزه‌های مختلف پروژه‌های نرم‌افزاری است که می‌توان اطلاعات غنی را در فازهای مختلف آن استخراج کرد. آن‌چه به‌عنوان منابع داده در این تحقیق مورد استفاده قرار می‌گیرد همان مخازن خطا است. که با استفاده از یک سیستم ردیابی خطا نمونه‌های آماری لازم استخراج می شود[۱۷].

[1] -Use Case

۳,۲. خطای نرم‌افزاری

اشـکال نرم‌افزاری، خطا، نقص و یا شـکسـت ایجاد شـده در برنامه کامپیوتری است، که باعث تولید یک نتیجه نادرسـت یا غیرمنتظره می‌شود. یا باعث بروز رفتار ناخواسته از سیستم می‌گردد. رفع این مشـکلات در پروژه‌های نرم‌افزاری از سخت‌ترین و پر هزینه‌ترین مراحل مهندسی نرم‌افزار است. این خطاها می‌تواند در پروژه‌های بزرگ ومخصـوصـا پروژه‌های متن باز[1] از سـوی همه کاربران و توسـعـه‌دهنـدگـان گزارش شـوند. از این رو نرم افزارهای ردیابی خطا به کمک آمده تا بتوان این گزارشـات را ثبـت و پی‌گیری کرد. خاطر نشـان می‌کنم که در این تحقیق همان‌طور که قبلا گفتیم مشکلات اعم از نقص، شکست یا خطا با عنوان خطای نرم‌افزاری معرفی می‌شود.

۱,۳,۲.سیستم ردیابی خطا

سیستم ردیابی خطا یک برنامه نرم‌افزاری است که برای کمک در سهولت پی‌گیری خطاهای نرم افزاری در طول عمر پروژه طراحی و ارائه شـده‌اسـت. این سـیسـتم یک نرم‌افزار کاربردی برای تیم توسـعه است به‌طوری که به آن‌ها در تضمین کیفیت و کاهش هزینه ها و پیشرفت سریع پروژه کمک می‌کند. این سیستم‌ها به کلیه افراد دخیل در یک پروژه اعم از کاربران، توسعه‌دهندگان و ذینفعان دیگر اجازه می‌دهد که خطای رویت شده در نرم‌افزار را در هر مرحله از روند تولید که باشد گزارش داده و پیگیری کننـد. بسـیاری از پروژه‌های بزرگ از این نرم‌افزارها اسـتفاده‌کرده و پروژه‌های خود را در نسـخه‌های مختلف ارائه می‌دهند. تا در مراحل مختلف از ارائه نسـخه جدید خطاهای نسخه قبلی که دیگران گزارش داده‌اند پیگیری و رفع کنند. پروژه در هر حالتی ارائه شـود، چه به‌صورت مرحله‌ای یا کامل، متن‌باز یا بسـته و با هر متد اسـتفاده شـده در مهندسـی نرم‌افزار ، یک سـیستم ردیابی خطا در پیشبرد بهتر و سریع‌تر فرآیند مهندسی نرم‌افزار بسیار ارزشمند است.

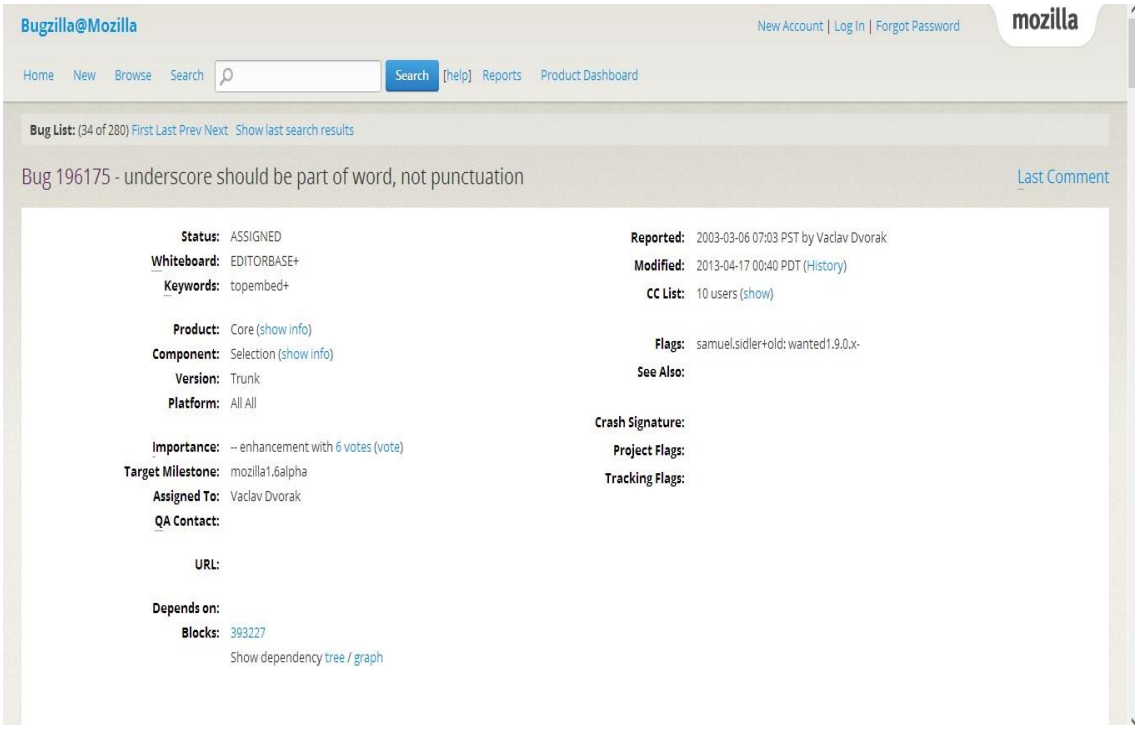
بسیاری از این نرم‌افزارها برای استفاده عموم طراحی شده و بقیه تنها برای استفاده در یک پروژه طراحی می‌شـوند. معمولا سـیسـتم‌های ردیابی خطا با دیگر اپلیکیشن‌های مدیریت پروژه یکپارچه و مجتمع می‌شـوند[۱۸]. مهمترین قسـمت این سـیسـتم‌ها مخازن اطلاعاتی آن‌هاست که کلیه اطلاعات مربوط بـه خطا را در خود ذخیره می‌کنند. باید دید در یک سـیسـتم ردیابی خطا ایده‌ال چه اتفاقی می‌افتـد و یـک خطـا چه چرخه‌ای را در طول عمر خود در آن طی می‌کند. به‌عنوان مثال باگزیلا[2]یک سیستم ردیابی خطا است که در سال ۱۹۹۸ توسط تری ویسمن[3] برای شرکت موزیلا[4] نوشته شد.

---

[1] -open source
[2]-Bugzilla
[3]-Terry Weissman
[4]-www.mozilla.org

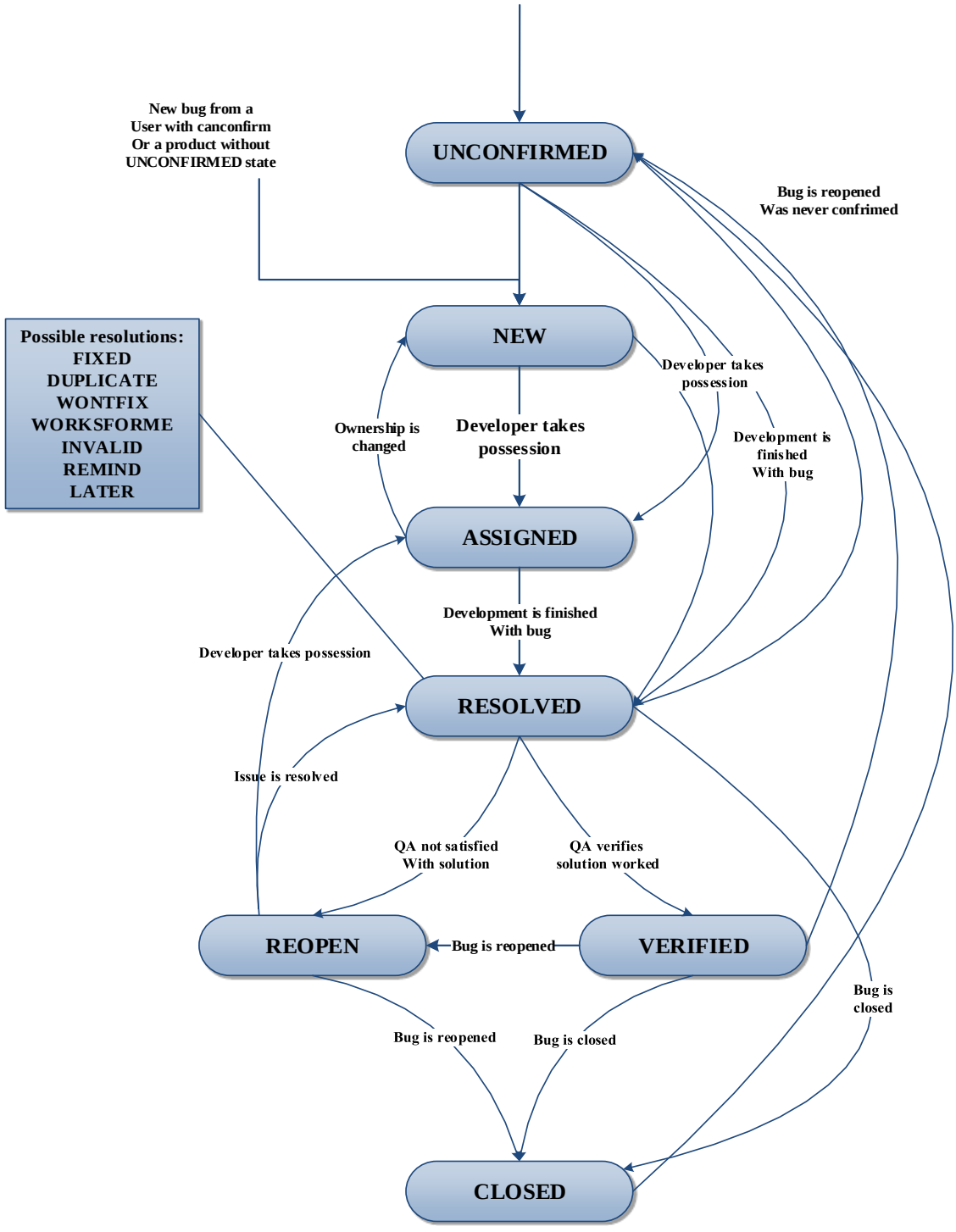
در ابتدا باگزیلا به‌عنوان یک سیستم ردیابی خطا متن‌باز از داخل خانه برای کاربران مرتبط با نت اسـکیپ[1] طراحی شد. ابتدا به زبان TCL[2] و سپس به Perl برای راحتی و سهولت استفاده و دخالت همگان در پیشرفت کار برگردانده شد. این سیستم برای استفاده در پروژه‌های متن باز بهینه‌سازی شد. به‌طوری که شرکت‌ها و پروژه‌های بزرگی چون یاهو، ناسا،Gnome ، KDE ، Apache ، RedHat، ویکی مدیا و خود موزیلا از آن استفاده کردند هر خطا یک شناسه(ID)، عنوان، تاریخ، وضعیت، نام محصول و مولفه، میزان شدت سختی و پیچیدگی، اولویت و غیره را در خصوصیات خود دارد[۹] (شکل۲-۱).

شکل۲- ۱-مشخصات یک موضوع (خطا)در نرم افزار Bugzilla

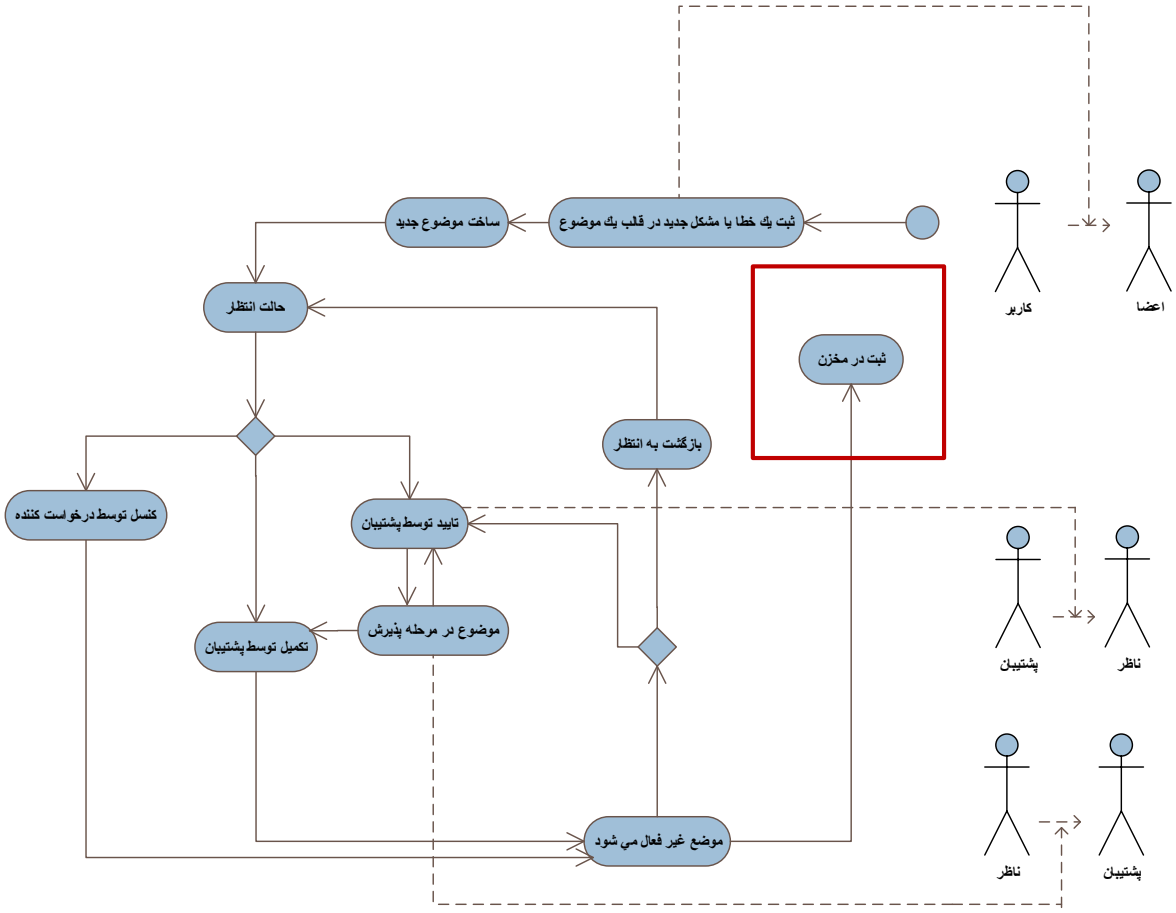


یک خطا از زمان ایجاد تا زمان بسـته‌شـدن(حل مشکل) چرخه‌ای را در سیستم دارد(شکل۲-۲) نمودار فعالیت برای یک مخزن خطا در شکل۲-۳ نشان داده شده‌است.

[1]-Net Scape
[2]-Tool Command Language



شکل۲- ۲ - چرخه یک خطا در یک سیستم مدیریت خطا[۱۹]



شکل۲- ۳ – نمودار جریان کار[1] یک سیستم ردیابی خطا

۱- خطا جدید (NEW) یا تایید شده وارد سیستم می‌شود(unconfirmed) اگر اعتبار خطا توسط یکی از اعضای تیم یا هسته مدیریتی تایید شد، به‌عنوان یک گروه جدید ثبت می‌شود.

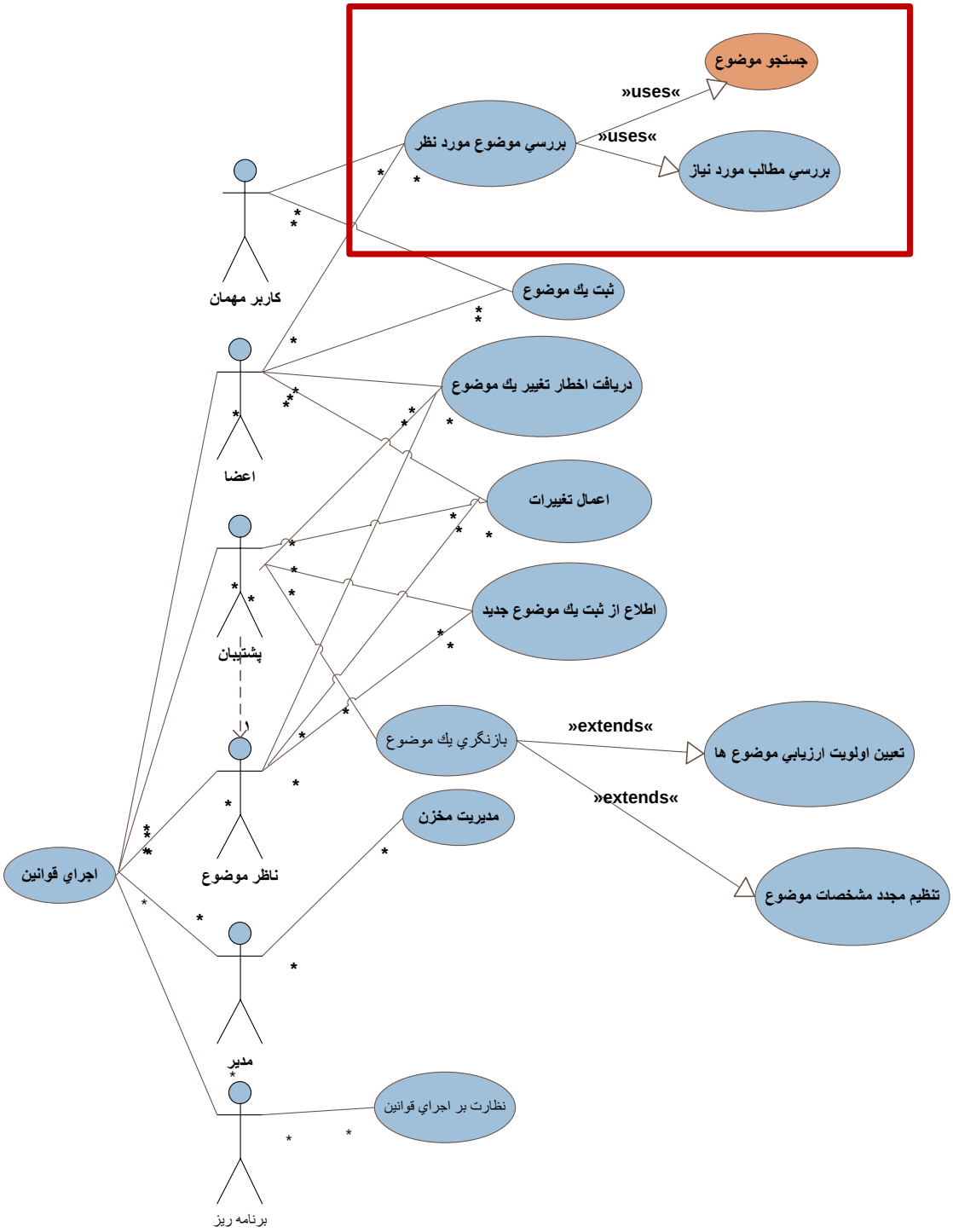
۲- مدیران وکسانی که مسئول ارزیابی خطاها هستند باید وجود خطا را تایید کنند. اعتبار و غیرتکراری بودن آن را نیز بررسی‌کنند. اگر خطا تایید شد به قسمت تایید شده‌ها (Assigned) رفته، تا در اختیار تیم توسعه برای حل و فصل و رفع خطا قرار گیرد. در غیر این صورت به قسمت تایید نشده (unconfirmed) برای بررسی بیشتر می‌رود.

۳- اگر خطا رفع شد برچسب حل شده (Resolved) خورده و آن را به وضعیت حل شده انتقال می‌دهند.

[1] -Work Flow Diagram

٤- اعضای تیم ممکن است یک خطا حل‌شده را به یکی از حالت‌های سازگار(Verified) و سپس بسته (Closed) اتصال دهند. یا دوباره به شکلی در رابطه با این خطا مواجه شدند و آن را به حالت دوباره باز یا تایید شده بفرستند.

خطاهایی که به مرحله Resolved بازگردانده می‌شوند یا نامعتبرند یا تکراری هستند و نیاز به بررسی مجدد دارند و یا به‌طور کامل رفع شده‌اند. چرخه عمر یک خطا از مرحله‌ی جدید تا مرحله خروج از حل تعریف می‌شود. شکل۲-٤ وظایف هر یک از اشخاص در ارتباط با مخزن خطا را نشان می‌دهد. دو کادر قرمز دردو نمودار شکل۲-٤ و شکل۲-۳ محدوده کاری این تحقیق را نشان می‌دهند.



شکل۲-٤- نمودار مورد کاربرد[1] یک سیستم ردیابی خطا

همان‌طور که در شـکل۲-۵ می‌بینید، یک خطا دارای مشخصات متنی است. برچسب، وضعیت خلاصه و در قسمت جزئیات توضیحات و راه‌حل‌های ارائه شده با ذکر نام نویسنده و تاریخ برای هر خطا درج شـده اسـت. همین اطلاعات و متون ثبت شـده دامنه کار تحقیقاتی ماست. این داده‌ها منبع غنی اطلاعات و دانش سودمند است. اینکه چگونه این داده‌ها را کاوش کنیم و چه اطلاعاتی در آن‌ها

[1] - Use Case Diagram

نهفته اســت، گام‌های تحقیق ما را می‌سازند. از آنجایی که این داده‌ها توسط افراد مختلف و با استفاده از قوانین نحوی صــحیح و ناصــحیح و گاه کلمات مشــابه دریک حوزه اســتفاده شــده‌اند نیاز به یک جستجو معنایی به شدت احساس می‌شوند.

ID	Status	Version	Component	Severity	Summary
1026187	NEW	unspecified	General	normal	Warnings displayed when running tests
1026149	NEW	unspecified	General	normal	combining multiple ui buttons
1025939	NEW	unspecified	General	normal	TEST-UNEXPECTED-FAIL | tests/test-native-options.testSimplePrefs | [Exception... "Component (NS_ERROR_UNEXPECTED) [nsIPrefBranch.setBoolPref]
1025356	UNCONFIRMED	unspecified	General	normal	Firefox 30 extension with Add-on SDK throwing NS_ERROR_FAILURE in sdk/timers.js
1025315	NEW	unspecified	General	normal	only run sdk/preferences/native-options.enable() if optionsURL is data:text/xml,<placeholder/>

شکل۲- ۵-مشخصات یک خطا

پایه و اســاس این تحقیق جستجو جملات مشابه معنایی است. به‌صورتی که آن‌دسته از جملات که دارای تشــابه معنایی بیشــتری با جمله یا کلمه کاندیدای ما هستند، جداسازی شوند. چگونگی این کار در سه مرحله خلاصه می شود: ابتدا با استفاده از تکنیک‌هایی که در ادامه پیشنهاد می‌شود از میان داده‌های آماری‌که از یک مخزن خطا انتخاب شده، میزان تشابه هر خطا (بر اساس داده‌های متنی مانند توضــیحات، ســرآمد، راه‌حل و غیره) با خطای جدید اندازه‌گیری می‌شــود. در مرحله بعد این خطا با استفاده از یک الگوریتم خوشه‌بندی که سعی شده یک الگوریتم بهینه باشد به دسته‌های جدا بر اساس ضریب تشابه محاسبه شده تقسیم می‌شوند.

این کار به کاربر کمک می‌کند که نمونه‌های مشابه گزینش شده را در خوشه‌های طبقه‌بندی شده از نظر ســختی و پیچیدگی کار در اختیار داشــته‌باشــد.گاهی ممکن اســت کاربر خود بر اساس تجربه حدس بزند که مشکل زیاد پیچیده نیست. یا برعکس مشکل به زمان زیادی برای حل نیاز داشته باشد، در این حالت این خوشــه‌بندی می‌تواند در کاســتن زمان یافتن راه‌حل کمک قابل توجهی داشته باشد. آن‌چه این کار را از کارهای انجام شــده قبلی متمایز می‌کند توجه به بهره‌وری و بهبود کارایی اســت. یک مخزن خطا مجموعه‌ای گســترده و بزرگ از داده‌هاست که هر روز بزرگتر می‌شود. با بزرگ شدن نمونه آماری و نیاز به دقت در کار ابعاد پیچیدگی این عمل روشــن خواهد شــد. هدف اصلی این کار اســتفاده از تکنیکی اســت که دقت بالاتری داشــته باشد. جستجو در یک بانک داده که متون در آن از قانون خاص برای رعایت اصول تحریر و قواعد نحوی استفاده نشده است، و کسانی که داده‌ها را ثبت کرده‌اند ممکن است از کلمات مشابه زیادی استفاده کرده باشند، نیاز بیشتری به یک جستجوی معنایی دارد.

٤,٢.تحقیقات پیشین در حوزه داده‌کاوی در مخازن خطا

در سال ۱۹۹۲ اولین نرم‌افزار ردیابی خطا GNATS[1] شروعی برای MSR بود. با ارائه اولین نرم‌افزار ردیابی خطا، مدیریت و استفاده از دانش نهفته در خطاها اولین قدم‌های خود را برداشت. در این مدت محققان سعی‌کردند برای استخراج بهینه و مفیدتر دانش، مدل‌ها و روش‌های جدیدی با استفاده از الگوریتم‌های مختلف ارائه کنند. بیشتر این روش‌ها از الگوریتم‌های دسته‌بندی برای دسته‌بندی اطلاعات و استفاده از دانش آن‌ها به‌صورت طبقه‌بندی شده استفاده کردند. ابتدا مروری کوتاه بر تحقیقات قبلی و روش های بررسی شده خواهیم داشت. نواقص و کمبودهایی که به موجب آن ها این تحقیق انجام شده و روش ارائه شده‌است نیز بررسی خواهد شد.

سال ۲۰۰۰، JunzoWatada به منظور برآورد تعداد خطاها در یک پروژه نرم‌افزاری از رگرسیون فازی استفاده‌کرد[٦]. وی مجموعه سئوالاتی را برای استفاده در سیستم فازی در مراحل مختلف از یک پروژه مطرح کرد. در این روش تمام داده‌ها برای جستجو هدف قرار نمی‌گیرد و این خود ضعف بزرگی است.

Lucas D.Panger در سال ۲۰۰۷ مقایسه‌ای برای دسته‌بندی مخازن خطا با استفاده از پنج الگوریتم دسته بندی 0-R، 1-R، درخت تصمیم‌گیری C4.5، Naïve Bayes و رگرسیون لجستیک، در داده‌های گرفته شده از Bugzilla انجام داده‌است[٥]. این کار با محاسبه درصد خطاهایی که دسته‌بندی شده‌اند و ضریب Kappa برای هر کدام با استفاده از نرم‌افزار Weka انجام داده است. داده‌ها بر اساس طول عمروتوضیحات متنی دسته‌بندی شده‌اند. این تحقیق تنها مقایسه‌ای بین الگوریتم‌های مختلف در یک دسته‌بندی ساده برای داده‌های موجود در یک مخزن است، که کمتر به شباهت بین یک خطا جدید و خطاهای دیگر پرداخته شده است.

در همان سال CathrinWeib و همکارانش با استفاده از داده‌های گرفته شده از JBoss در سه مرحله طول عمر خطای جدید را تخمین زدند[٩]. ابتدا به‌کمک الگوریتم نزدیک‌ترین همسایهα-KNN گزارش‌های مشابه را از منبع استخراج می‌کند و در مرحله بعد با استفاده از موتور جستجوLucene داده‌ها با شباهت متنی بیشتر استخراج می‌شود. Lucene ازSVM و یک مدل بولی[2] برای جستجو و ارزیابی متون استفاده می‌کند. در این تحقیق اگر موضوع با شباهت با توضیحات خطای جدید پیدا نشد، باز الگوریتم نزدیک‌ترین همسایه α–KNN را برای استخراج داده استفاده می‌شود پس از این مراحل داده‌های فیلترشده می‌تواند در تخمین طول عمر خطا جدید و حل آن کمک می‌کند. این روش جزء روش‌های ابتدایی با به‌کارگیری اندازه‌گیری تشابه بین متون بود. اما موتور و الگوریتم آن نه‌تنها به معنای کلمات توجه نمی‌کرد بلکه جزه روش‌های ابتدایی جستجو در متون متشابه بود.

[1]-GNU product’s issue-tracking software
[2] -Boolean model

Nagwani در سال ۲۰۰۹ روشی را با دامنه خطاهای حوزه GUI ارائه کرد. در این تحقیق خطاهای مربوط به GUI را از پروژه مختلف را به عنوان داده در نظر گرفت. از نظر او خطا در طول فرآیندی رخ می دهد که این فرآیند شامل چند رویداد متوالی است[۱]. مدل وی در سه مرحله داده ها را فیلتر می کند. در مرحله اول داده ها بر اساس رویداد، در مرحله بعد برای قسمت‌های مختلف هر رویداد مقایسه می‌شوند. در مرحله آخر براساس ملزومات هر قسمت تفکیک و با هم مقایسه می‌شوند. در این روش خطاهایی که از نظر مرحله و رویدادهای موجود در فرآیند دارای تشابه بیشتر با خطای جدید هستند از منابع استخراج می‌شود. این روش اگرچه به مفاهیم و معانی توضیحات هر موضوع خطا بیشتر توجه کرده، اما در پیاده‌سازی موانع زیادی وجود دارد. مراحل توسط نیروی انسانی مجزا و مشخص شوند و تمام این رویدادها در طول یک فرآیند خطا دار باید مشخص باشند. همچنین محدود به خطاهایی از که در یک فرآیند مشخص در GUI است.

یک سال بعد Nagwani وBhansali با استفاده از الگوریتمK-means مجموعه‌ای از داده‌ها شامل طول عمر و شماره داده را به خوشه‌هایی تقسیم کردند، تا بتوانند میزان پیچیدگی‌یک خطا را با تخمین ذهنی طول عمر یک خطای جدید را مشخص کند. به این ترتیب ابتدا کاربر برای طول عمر خطا جدید تخمینی می‌زند و این تخمین با مراکز خوشه‌های ایجاد شده بر اساس طول عمر داده‌ها، پیچیدگی خطا جدید را مشخص می‌کند[۱۱]. این روش دو مورد مهم توجه نشده اول اینکه تخمین ذهنی طول عمر یک خطا زیاد اصولی و دقیق نیست. دوم اینکه تعیین میزان پیچیدگی یک خطا با یک پس‌زمینه ذهنی کمک چندانی در پیشرفت پروژه و بهره‌وری در هزینه و زمان آن ندارد.

در تحقیقی دیگر در همان سال، وی با استفاده از سه الگوریتم تشابه جاکارد، کسینوسی و TF-IDF میانگینی از تشابه موجود بین قسمت‌های متنی مختلف از خطای جدید و خطاهای دیگر در منبع خطا را محاسبه می‌کند[۱۰]. نتایج را برای هر الگوریتم با هم مقایسه کرده است. این کار با محاسبه تشابه بین بخش موضوع، توضیحات و دیگر بخش‌های متنی از هر موضوع موجود در منبع خطا را با خطای‌جدید، به‌صورت جداگانه محاسبه می‌کند. این نتایج با ضرایب وزنی مشابه با هم جمع می‌شوند و به عنوان میزان شباهت خطای‌جدید با خطا داخل مخزن در نظر گرفته می‌شود. برنامه‌ای که Nagwani ارائه کرده است این کار را با استفاده از هرسه الگوریتم انجام می‌دهد.

برای نمونه در این تحقیق، با استفاده از ضریب کسینوسی برای یک نمونه جدید، نتایج محاسبه و چهارخطا با شباهت بیشتر، به‌عنوان کمک برای حل خطا و محاسبه میانگین طول عمر آنها، به‌عنوان تخمینی برای طول عمر خطا استخراج شده‌است. در این روش برای بررسی میزان تشابه بین متون خطا جدید و داخل مخزن از روش‌های خطی استفاده شده که برای شروع مناسب است، اما کافی نیست. این الگوریتم‌ها معایبی دارند که الگوریتم‌های بهتری را می‌توان استفاده‌کرد.

به‌عنوان مثال الگوریتم‌های معنایی که در آنها علاوه بر تشابه لغوی و نحوی تشابه معنایی بین جملات و متون را در نظر گرفته می‌شود.

در سال ۲۰۱۱، Nagwani تحقیق خود را روی نویسندگان و توسعه‌دهندگان ثبت شده در مخزن خطا وموضوعاتی که جوابگوی آنها بوده‌اند یا نظرات موثری در رفع خطا داشته‌اند، متمرکز کردند[۱۲]. به‌این‌صورت که روی تعدادی داده که از Bugzilla استخراج شده بود کلمات تکراری مهم در این موضوع‌ها را به‌عنوان داده استخراج می‌کند. لیستی ازتوسعه‌دهندگان و کاربران ثبت کننده به تفکیک کلمات اصلی تکرار شده لیست می‌شود. در این حالت می‌توان کسانی را که در حوزه موضوع خطای جدید کار کرده‌اند، برای کمک به حل مشکل شناسایی کرد. در این روش تشخیص موضوع اصلی و کلمات مهم باید مطابق با مجموعه و در مقایسه با آن استخراج شود. انتخاب و یا تشکیل مجموعه بهینه خود نیازمند ارائه روشی مناسب است.

Suma.V و همکارانش در سال ۲۰۱۲ به استفاده ازچند روش خوشه‌بندی و مقایسه آن‌ها اکتفا کرد.آنها هشت الگوریتم را برای دسته‌بندی داده‌ها در مخازن خطا استفاده کردند که در میان آنها K-means در برابر افزایش داده‌ها نتایج بهینه‌تری در اختیار کاربر می‌گذارد[۸]. الگوریتم‌ها با استفاده از نرم‌افزار Weka پیاده‌سازی شده‌اند. این روش نیز تنها به دسته‌بندی ساده اکتفا شده و به نوع داده‌ها که متنی هستند توجه نشده است.

Nagwani در جدیدترین تحقیق خود در ۲۰۱۳ الگوریتم‌هایNB ، J84، SVM ، CC و CLUBAS را روی داده‌های گرفته شده از پروژه Mozilla برای دسته‌بندی استفاده کرد و نتایج را تحلیل کرده‌است[۲۰]. به‌عنوان نتیجه نشان داده‌است که NB و J84 با زیاد شدن حجم داده‌ها افت دقت ندارد.

پس از مطالعه‌ای اجمالی روی BTS ها و روش‌هایی که تاکنون برای جستجو و استفاده دانش موجود در این سیستم‌ها نشان می‌دهد، که سیر پیشرفت آن مدتی است که متوقف شده. در واقع پیشرفتی چشم‌گیر در آن‌ها دیده نمی‌شود. آن‌چه نبودش بیش از همه درکاوش در داده‌های مهم موجود در این مخازن، به چشم می آید، هوشمند سازی این کاوش هاست . داده های موجود در این مخازن مجموعه ای از متون و ارقام است. پس جستجو و کاوش پیشرفته با در نظر گرفتن جنبه های معنایی موجود در آنها، مطمئنا نتایج بهتری را در اختیار کاربران قرار خواهد داد .

در این تحقیق سعی شده به این جنبه از داده کاوی مخازن، یعنی استفاده از الگوریتم هایی که به مفهوم و معنای داده ها و استخراج دانشی بهینه و مختصرتر توجه شود.

۵,۲. اندازه گیری شباهت بین متون

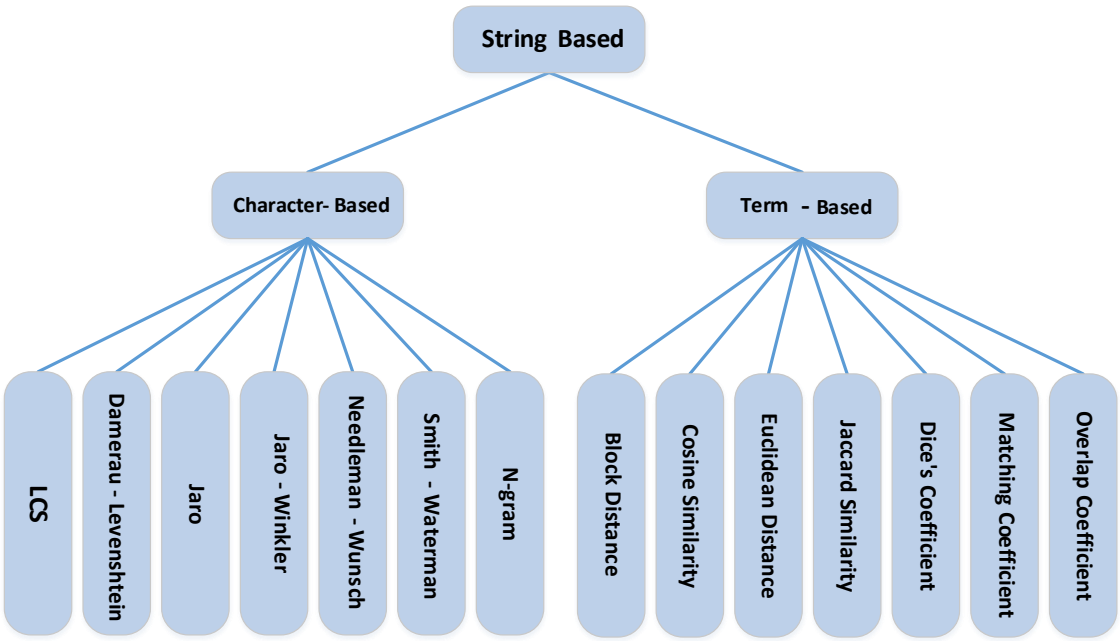
پیدا کردن شباهت بین دو متن و جمله با پیدا کردن شباهت بین کلمات آغاز می شود. کلمات به صورت معنایی یا لغوی می توانند به هم شبیه باشند.شباهت لغوی به این معنی است که حروف کلمات با هم مشابه باشند. و شباهت معنایی حالتی است که دو کلمه در یک زمینه و به یک نوع و معنی استفاده شوند.

شباهت معنایی در روش ها مقوله ای جدا از شباهت خطی است. شباهت معنایی به صورت شباهت بر پایه مجموعه و شباهت بر پایه دانش معرفی می شود. شباهت خطی کاملا متریک بوده و تشابه دو جمله را بر اساس توالی دستهای و ترکیب کاراکترها و کلمات داخل رشته بررسی می کند. شباهت معنایی، شباهت بین کلمات دو جمله براساس اطلاعات مجموعه ای که شامل آن می باشد، را مشخص می کند.ابتدا متداول ترین روش های هر دو معیار را به اختصار معرفی کرده و سپس روش مورد استفاده این تحقیق را با شرح کامل دلایل انتخاب شرح می دهیم.

۱,۵,۲.شباهت خطی

معیار شباهت خطی بر پایه توالی و موقعیت کاراکتر ها بنا شده است. در اندازه گیری این شباهت میزان (فاصله) شباهت دو جمله بر اساس همین دو معیار یعنی توالی خطی و موقعیت کلمه در جمله اندازه گیری می شود. از چهارده روش و معیار اندازه گیری ارائه شده در طول تحقیقات عمومی، در این گروه هفت معیار مربوط به گروه توالی خطی[1]و هفت معیار و روش به گروه موقعیت کلمه[2] تعلق دارند.(شکل۲-٦)

[1]-Term-based
[2] -Character-based



شکل۲- ٦ – انواع الگوریتم های تشابه خطی

۱,۱,۵,۲.اندازه گیری شباهت بر پایه کاراکتر[1]

- بزرگترین زیر دنباله مشترک[2] (LCS)

روشـــی اســـت برای پیدا کردن بزرگترین زیر دنباله در مجموعه ای از دنباله ها. معیار شـباهت بزرگترین رشته پیوسته از کاراکترهاست که در هر دو جمله موجود باشند. هدف در این روش مقایسـه دو رشـته S_1 و S_2 است به طوری که رشته S_3 با حفظ ترتیب، و نه لزوما متوالی در هر رشته موجود باشد.

- Dameran-LevelShtien

تعریف فاصله بین دو رشته با شمارش حداقل تعداد عملیات مورد نیاز برای تبدیل یک رشته به دیگری که در آن عملیات باید درج، حذف و یا جایگزینی یک واحد یا جابجایی در حرف مجاور باشد.

- Jaro و Jaro-Winkler

بر پایه تعداد و ترتیب کاراکترهای مشـترک بین دو رشته تعریف می شود به طوری که فاصـله دو کلمه مشـترک با توالی یکسـان دردو خط هر چه کوتاه تر باشد دو جمله شباهت بیشتری دارند.

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

[1] -Character-Based
[2]-Longest Common Subsequence

$$d_w = \begin{cases} d_j & \text{if } d_j < b_t \\ d_j + (l_p(1-d_j)) & \text{otherwise} \end{cases}$$

m تعداد کل کاراکترهای مشترک و t تعداد کاراکترهایی است که جا به جا شده اند. در فرمول دوم که ضـریب Jaro-Winkler اسـت ، d_j ضریب Jaro و ℓ تعداد کاراکترهای ثابت قبل اولین کاراکترجابه جا شده و p ضریب ثابت تفکیک است که باید بیشتر از ۰,۲۵ باشد.

- نیدلمن-رانچ[1]

یک همترازی برابری بر دو ترتیب متوالی (مانند A وB) انجام می دهد. این الگوریتم نمونه ای از برنامه نویسـی پویاسـت. این معیار برای جملات و رشـته های هم اندازه دو شباهت قابل توجه کاربردی است.

- اسمیت واترمن[2]

این الگوریتم، الگوریتم نیدلمن رانچ را بهینه می کند به این صورت که به جای در نظر گرفتن تمام توالی کاراکتری، این الگوریتم سعی می کند با در نظر گرفتن بخشهای مختلف با همه طولهای ممکن میزان شـباهت را بهینه کند. این الگوریتم برای متونی که به نظر شـبیه نیستند اما ممکن است قسمت های کوچکی از آنها به هم شبیه باشد مناسب است.

- n-gram

زیر دنباله ای n موردی از دو رشـته یا متن را با هم مقایسه می کند. مثلا 1-gram کلمه به کلمه جملات را در نظر می گیرد یا 2-gram دو کلمه با هم از هر رشته، و به همین ترتیب فاصله و شباهت n-gram از هر رشته را تا حداکثر n اندازه گیری می کند.

۲,۱,۵,۲. شباهت بر پایه توالی[3]

- فاصله بلوک یا فاصله منهتن

این الگوریتم فاصله بین دو نقطه روی یک مسیر مشبک را محاسبه می کند. در شباهت بین دو جمله این فاصله با اندازه گیری مجموع تفاوت بین اجزاء مترادف نظیر به نظیر بدست می آید.

$$d_1(p,q) = \|p-q\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

- شباهت کسینوسی

ا ندازه گیری فاصله داخلی دو بردار متقاطع در فضا با استفاده از اندازه گیری کسینوس زاویه بین دو بردار. در اندازه گیری شـباهت بین دو جمله یا رشـته هر جمله نماد یک بردار

[1]-Needleman-Wunsch
[2]-Smith- Waterman
[3] -term-based

اســـت و ارزش هر یـک، برای نمونه تعداد دفعات، تکرار یک توالی، کاراکترها یا یک کلمه خاص در جمله، در نظر گرفته می شود.

$$similarity = \cos(\theta) = \frac{A.\mathrm{B}}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(\mathrm{B}_i)^2}}$$

- ضریب تاس[1]

این ضـریب به صـورت دو برابر عبارات مشترک در رشته های مورد مقایسه تقسیم بر تعداد کل عبارت های در نظر گرفته شده در هر دو رشته محاسبه می شود.

$$Qs = \frac{2C}{A+B} = \frac{2|A \cap B|}{|A|+|B|}$$

- فاصله اقلیدسی[2]

یا فاصـله L2 که همان فاصله بین دو نقطه در مختصات دکارتی است. برای شباهت بین دو جمله این فاصـله جذر مجموع تفاوت بین عبارات متناظر در دو رشته در نظر گرفته می شود. $d(p,q) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\ldots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$

- ضریب تشابه جاکارد[3]

نسبت مجموع عبارات مشابه و مشترک در دو رشته به عبارات غیر مشترک

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- ضریب تطبیق[4]

یک ضریب تشابه بسیار ساده. اندازه عبارات مشترک به کل عبارات.

- ضریب همپوشانی[5]

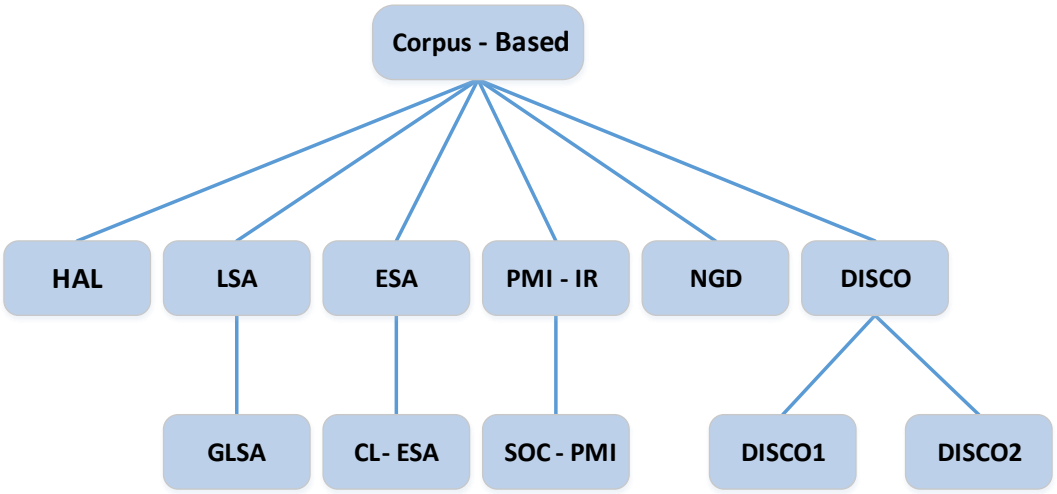
شبیه به ضریب تاس با این تفاوت که شباهت بین دو رشته را در حالی اندازه می گیرد که کلمه ای از یکی زیرمجموعه ای از دیگری باشد.

$$overlap = \frac{|X \cap Y|}{\min(|X|,|Y|)}$$

[1]-Dice's Coefficient
[2] -Euclidean distance
[3] -Jaccard similarity
[4] -Matching Coefficient
[5] -Overlap coefficient

۲,۵,۲. تشابه بر پایه مجموعه

سـنجش تشابه دو متن بر پایه مجموعه یک معیار سنجش تشابه معنایی است، که شباهت بین کلمات را بر اســـاس اطلاعات مفید موجود در یک مجموعه بدســـت می آورد. این مجموعه حجم انبوهی از دست نوشته ها و متون گفتاری است که برای تحقیقات زبانی مورد استفاده قرار می گیرند. می توان گفت که میزان تشابه وابسته به مجموعه ای است که کلمات و جملات از آنها استخراج شده اسـت. شـکل۲-۷ انواع روش های سـنجش تشـابه بر پایه مجموعه را نشان می دهد، که در ادامه به اختصار معرفی می شوند.



شکل۲- ۷- انواع الگوریتم های تشابه بر پایه مجموع

- Hyperspace Analoue to Language (HAL)

در این روش یک فضـای معنایی از کلمات موجود در مجموعه ساخته می شود به این شـکل که یک ماتریس کلمه در کلمه تشـکیل می شـود. به طوری که مقادیر آن نشان دهنده میزان ارتباط بین کلمه سـطر و ستون متناظر با آن مقدار است. سپس آنتروپی پایین در ستون ماتریس توسـط کاربر اسـتخراج می شود. در تجزیه تحلیل متن برای استخراج ماتریس یک قالب n کلمه ای را در نظر می گیریم.

(مثلا n=1) سـپس فاصـله وقوع هر کلمه با کلمه اولی شمارش می شود. وزن مقادیر مـاتریس رابطه معکوس با فاصـــله وقوع هر کلمه با کلمه ابتدایی دارد .کلمه ای که به کلمه ابتدایی نزدیک تر باشـد از نظر معنایی رابطه نزدیکتری و در نتیجه وزن بیشتری دارد. HAL همچنین بـه ترتیب قرارگیری نیز توجه می کند. یعنی اینکه همســـایه قبل از کلمه مورد نظر است یا بعد از آن به این منظور حالت همسایه مختلف را ثبت می کند.

- Latent Semantic Analysis (LSA)

تحلیـل معنایی نهفته متداول ترین روش در میان روش های مبتنی بر مجموعه اصـــلی اســـت. به ویژه برای تحلیل رابطه معنایی بین دســتهای از متون مورد اســتفاده قرارمی گیرد. روش کار به اختصـار اینگونه اسـت ،که ابتدا ماتریس کلمه و سطر یا پاراگراف تشکیل می-شـود، که ردیف ها نشـان دهنده کلمات منحصر به فرد و ستون ها نشان دهنده سطرهاست. پس از روش تجزیه مقدار دهی منفرد (SVD)[1]به منظور کاهش ســطر و ســتون ماتریس در حالی که ساختار شباهت بین ردیف ها حفظ می شود، استفاده می کند.د پس از آن کلمات با محاسـبه کسـینوس رابطه بین دو بردار تشـکیل شـده توسط هر دو ردیف مقایسه می شوند. مقادیر هر چه به یک نزدیک تر باشد، دو کلمه به هم شبیه تر بوده و هر چه به صفر نزدیک-تر باشد، شباهت کمتری دارند.

- تعمیم تحلیل معنایی نهفته GLSA[2]

یک چارچوب برای محاسـبه میزان اهمیت عبارات و سند حاوی آن . در این روش که بر پایه LSA است، توجه بر ارزش سند حاوی عبارت به جای روش ارائه شده در LSA شده که به تعداد کلمات موجود در سـطرها و مقایسه آنها توجه کرد. GLSA به این منظور نیاز به محاسبه میزان تشابه معنایی بین عبارات و نیز استفاده از روش های کاهش ابعاد مورد مقایسه دارد. برای رسیدن به نتیجه مطلوب بهترین و کارآمدترین روش ها را با هم ترکیب می کند و در آخر ماتریس عبارات موجود در اسـناد که در LSA معرفی شـده، برای محاسـبه وزن در ترکیب بردارهای خطی از عبارات مورد استفاده قرار می گیرد.

- تحلیل معنایی صریح ESA[3]

معیاری برای محاسـبه ارتباط معنایی بین دو متن دلخواه اسـت. ESA به عنوان روشـی برای بهبود طبقه بندی متون معرفی شـده است. در این روش یک ماتریس tf-idf تشکیل می شـود که مقادیر آن نشـان دهنده رابطه معنایی میان دو متن اسـت که توسـط اندازه گیری کسینوسی محاسبه می شود.

[1]-Singular Value Decomposition
[2]-Generalized Latent Semantic Analysis
[3]- Explicit Semantic Analysis

- تحلیل صریح معنایی متقابل زبانی CL-ESA[1]

یک تعمیم چند زبانه از ESA است. که از یک متن چند زبانه به عنوان مرجع در مرکز چندین متن دیگر که از نظر زبانی مستقل هستند استفاده می کند. ارتباط این اسناد و میزان وابستگی آن ها به سن مرجع از طریق اندازه گیری کسینوسی محاسبه می شود. ویکیپدیا از این روش برای دسته بندی و اتصال اسناد متنی خود استفاده می کند.

- بازیابی اطلاعات متقابل نقطه به نقطه PMI-IR[2]

روشی است برای محاسبه شباهت بین کلمات است. موتور جستجو آستاویستا از این روش برای محاسبه احتمال در جستجو پیشرفته استفاده می کند. دو کلمه که در یک صفحه وب نزدیک ترین موقعیت را نسبت به هم داشته باشد نمره تشابه بالاتری در PMI-IR خواهد داشت.

- بازیابی اطلاعات متقابل نقطه به نقطه نوع دوم SCO-PMI[3]

در این روش شباهت معنایی در واژه، با لیست کردن کلمات همسایه با آن در واژه در متن کلی انجام می شود. مزیت این روش در این است که می توان شباهت بین دو کلمه را که در همسایگی هم نیستند اما رابطه معنایی دارند را نیز اندازه گیری کرد.

- فاصله نرمال گوگل NGD [4]

اندازه گیری شباهت معنایی که از بازدیدهای صورت گرفته به وسیله موتور جستجو گوگل برای یک مجموعه از کلمات کلیدی بدست آمده است. در موتور جستجو فاصله دو کلمه کلیدی با معنی یکسان یا نزدیک به هم در زبان طبیعی رابطه نزدیکتری نسبت به دو کلمه غیر هم معنی دارند و در اصلاح به هم نزدیکتر هستند.

$$NGD(x,y)=\frac{\max\{\log f(x),\log f(y)\}-\log f(x,y)}{\log N-\min\{\log f(x),\log f(y)\}}$$

M در اینجا تعداد صفحات وب جستجوشده به وسیله Google، f(x) و f(y) تعداد موقعیت ها، در جستجو عبارتهای x و y هر دو وجود داشتند. اگر این دو کلمه به هم نزدیک نباشد وجدا از هم در صفحه ظاهر شوند، ضریب NGD بی نهایت است و چنانچه هر دو دائما کنار هم ظاهرشوند NGD آنها صفریا برابر با مقداری بین مربع x و مربع y خواهد بود.

[1]-Cross Language-Explicit Semantic Analysis
[2]-Pointwise Mutual Information-Information Retrieval
[3]-Pointwise Mutual InformationSecond Order Co-Occurrence
[4]-Normalized Google Distance

- استخراج توزیعی کلمات مشابه با استفاده از تکرار وقوع کلمات DISCO[1]

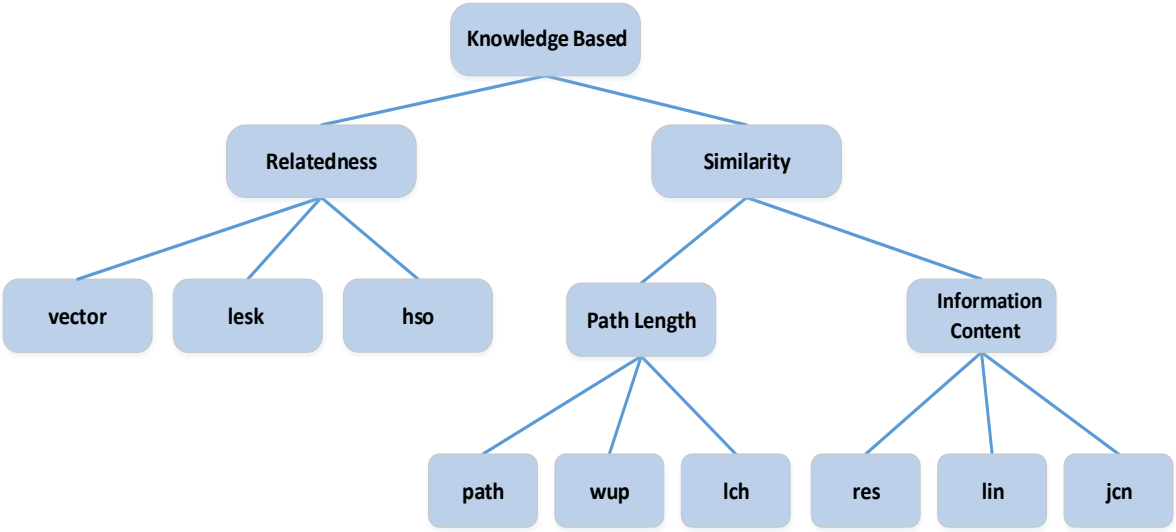
در این روش فرض بر این است که کلمات مشابه در زمینه های مشابه استفاده می شوند. بر همین اساس مجموعه های بزرگ متن مورد تجزیه و تحلیل آماری قرارمی گیرند تا شباهت توزیعی بین کلمات استخراج شود. درDISCO شباهت توزیعی بین کلمات با استفاده از یک قاب متحرک با اندازه۳± برای اندازه گیری دفعات پدیدار شدن کلمه، اندازه گیری می شود .دو معیار DISCO1 و DISCO2 نیز توسعه ای از معیار اصلی هستند به طوری که DISCO1 شباهت دو کلمه را بر اساس ترتیب مجموعه که کلمه در آن است محاسبه می کند و DISCO2 میزان تشابه را بر اساس مجموعه کلمات توزیع شده مشابه با کلمات مورد نظر محاسبه می کند.

۳,۵,۲.تشابه بر پایه دانش

روش های مبتنی بر دانش بر اساس شناسایی میزان شباهت بین کلمات با استفاده از اطلاعات بدست آمده از شبکه های معنایی عمل می کنند. Wordnet[2] متداول ترین شبکه معنایی است. Wordnet یک پایگاه بزرگ واژه ای در زبان انگلیسی است. اسمها، فعل ها، صفت ها و قیدها به صورت مجموعه ای از مترادف های مشابه دسته بندی شده اند(synsets) که هر کدام بیان کننده یک مفهوم مجزا هستند .

synsets ها با استفاده از روابط معنایی و مفهومی و روابط لغوی به هم متصل هستند. روش های اندازه گیری تشابه بر پایه دانش به دو گروه تقسیم می شوند: اندازه گیری شباهت معنایی و اندازه گیری ارتباط معنایی. اندازه گیری شباهت معنایی همان گونه که از اسمش بر می آید هنگامی قابل اندازه گیری است، که دو کلمه در رابطه معنایی بر پایه شباهت با یکدیگر باشند، دو کلمه همانند و متشابه یکدیگر باشند اما اندازه گیری بر پایه رابطه معنایی دو کلمه می پردازد. به عنوان مثال اینکه کلمه ای نوع خاصی از دیگری باشد، یا دو کلمه مخالف هم باشند، یکی بخشی از دیگری باشد یا غیره.

[1] -Extracting DIStributionally similar words using CO-occurrences
[2] -www.wordnet.princeton.edu



شکل۲- ۸- انواع الگوریتم های تشابه برپایه دانش

از میان نه روش ارائه شـده اندازه گیری شـباهت معنایی(شـکل ۲-۸) سه روش بر پایه محتوا اطلاعات و سـه روش بر پایه طول مسـیر اندازه گیری می شـوند سـه روش بر پایه رابطه مفهومی به اندازه گیری شـباهت می پردازند. معیار path عددی را به عنوان میزان شباهت دو کلمه بر می گرداند که بر کوتاه ترین مسیر مفهومی که دو کلمه را به هم متصل می کند ، دلالت دارد به عنوان مثال father و parent در شـبکه جزء یک طبقه بندی متصل هستند یا نه و فاصله مفهومی بین این دو کلمه چقدر است. در روش HSO زنجیره ی لغوی بین دو کلمه پیدا شده و بر اساس آن رابطه دو کلمه در یکی از سـه رشته ارتباطی تعریف می شود فوق العاده قوی- قوی- متوسط که حداکثر نتیجه برای یک رابطه عدد ۱٦ است.[۲۱]

٤,٥,۲. اندازه گیری شباهت ترکیبی

همانگونه که از اسـم این روش ها بر می آید این روش ها با ترکیب روش های قبل سعی به از بین بردن نواقص و کاستی های هر روش با روش دیگر کرده اند تا معیارهای بهینه تری را ارائه دهند. بسـیاری از تحقیق ها نیز در این حوزه صـورت گرفته تا به حال هشت روش آزمایش شده ارائه شده که دوتای آنها بر پایه اندازه گیری بر اسـاس مجموعه، شـش تای دیگر بر اساس اندازه گیری بر پایه دانش مطرح شـده اند. روش ارائه شـده در [۲۲] ابتدا شباهت معنایی بین کلمات از یک پایگاه دانش لغوی و مجموعه استخراج می شود و سپس در مرحله دوم تاثیر ترتیب و جای کلمه در معنای جمله را در نظر می گیرد.

در روش STS تشـــابه متن معنایی میزان شـــباهت بین کلمات را با ترکیب اطلاعات معنایی و نحوی اندازه گیری می کند. STS از دو روش شباهت خطی و شباهت معنایی به همراه روش انتخابی common word order بهره می گیرد.

STS در[۲۳] روی سی جفت کلمه روش جدید را آزمایش کرده و با محاسبه ضریب همبستگی پیرسون[1] نتایج را بهبود بخشید. در روش [۲٤] نیز از اندازه گیری معنایی بر اساس مجموعه همراه با میزان شباهت معنایی بر اساس دانش برای کلمات هم نقش در جملات مختلف استفاده شده است. مهمترین خصوصیات این روش استفاده از مدل های یادگیری ماشین مانند رگرسیون خطی و مدل bugging برای بدست اوردن یک درجه شباهت موثر بین جملات است.

در [۲۵] دو روش اندازه گیری مفهومی با استفاده از wordnet وNgram را با هم برای ارتباط دادن بین دو روش دستی و اتوماتیک انتخاب شده است.

همان طور که در قبل بیان شد، نمونه ها و بانک دادهای مورد استفاده در این تحقیق و به طور کل در مخازن خطا نرم افزار توسط یک گروه یا کاربر خاص تنظیم و ثبت نمی شوند . این داده ها توسط تمام کسانی که به نوعی در ارتباط با نرم افزار و پروژه هستند ثبت می شوند. پس طبیعی است که این متون از نظر نوع نگارش و دیکته لغات با الگو ویزه و اصولی نباشد. از سوی دیگر ممکن است این متون حاوی کلمات مشابه و هم معنا و حتی گاهی هم معنی اما غیر مربوط به هم باشند. در این شرایط برای استخراج دانش و فیلتر کردن داده ها نیاز به روشی است که این مشکلات تاثیر چندانی در نتایج آن نداشته باشد. همچنین در تشخیص جملات مشابه هم نیاز به دقت محاسباتی در روش های خطی و هم نیاز به دقت در معنا و نحو جملات و کلمات لازم است .

کلمات تخصصی در این متون پر اهمیت تر از کلمات متداول و اضافه هستند، پس نیاز است که اهمیت آنها به مراتب بیشتر از کلمات رایج در همه جملات است. روش مورد نظر پاسخگوی همه این نیاز ها خواهد بود همچنین در روش [۲۳] نه تنها شباهت ظاهری و معنایی و نحوی کلمات در نظر گرفته می شود بلکه به کلمات مجاور و حتی غیر مجاور آنها که در جمله یا متن ظاهر می شوند و روابط معنایی آنها با کلمه مورد نظر توجه می شود.

دلایل فوق باعث انتخاب روش اندازه گیری تشابه معنایی با استفاده از تشابه خطی وتشابه معنایی بر پایه مجموعه، در این تحقیق شده است. در ادامه به طور کامل این روش را توضیح می دهیم. روش مورد نظر شباهت بین دو متن را از نظر معنایی و اطلاعات نحوی (نظم متداول کلمات در زبان) مورد بررسی قرار می دهد. برای این کار از سه تابع شباهت استفاده می شود.

اول، شباهت رشته ها و شباهت معنایی کلمات محاسبه می شود. سپس برای ترکیب محاسبات با شباهت نحوی از تابع شباهت کلمات متداول انتخابی استفاده می کنیم. در نهایت ضریب شباهت دو متن با ترکیب شباهت رشته، شباهت معنایی بین کلمات و شباهت کلمات رایج و نرمال سازی محاسبات بدست می آید این روش STS[2]نام دارد.

[1] -Pearson correlation coefficient
2 -Semantic Text Similarity

فرض کنیـد کلمـه ای در دو جملـه بـه کار رفته باشـــد که این دو جمله معنا و حوزه مفهومی یکسانی داشته باشد اما در یک جمله کلمه مورد نظر اشتباه نوشته شده باشد.

به عنوان مثال دو جمله زیر را در نظر بگیرییید

1.Einstein was a German-born theoretical physieist.
2.Einstain was the scientist of physics at 19 century.

در دو جمله بالا Einstein در نقش اســم به کار رفته و هر دو جمله مفهومی نزدیک به هم دارند اما در جمله دوم این کلمه از نظر املایی درســت نوشــته نشده است. مگر از معیارهای تشابه مبنی بر فرهنگ لغات دیگر برای بررســی میزان تشــابه دو جمله بالا استفاده کنیم به خاطر تفاوت دو کلمه، یا کلمات موجود در جمله میزان تشابه کمی را نشان می دهند در حالی که این دو جمله بر اساس رشته کلمات و مجموعه ای که کلمه مورد نظر در آن است شباهت را اندازه گیری کند. در ادامه به توضیح بخش های مختلف مورد استفاده در روش برگزیده خود می پردازیم.

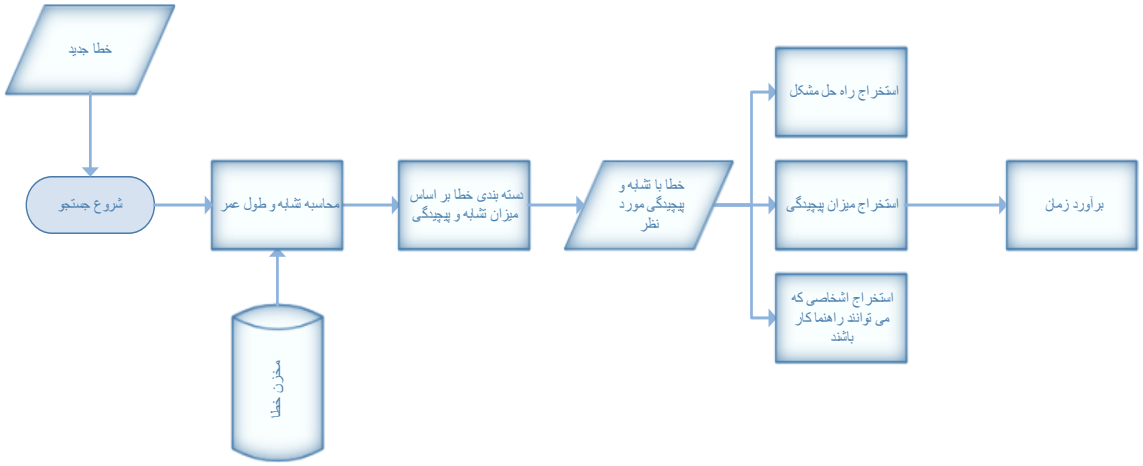
فصل سوم

استفاده از تکنیکهای داده کاوی برای کشف و رفع خطاهای نرم افزار بر پایه ضریب تشابه معنایی متن و خوشه بندی

۱,۳. مقدمه

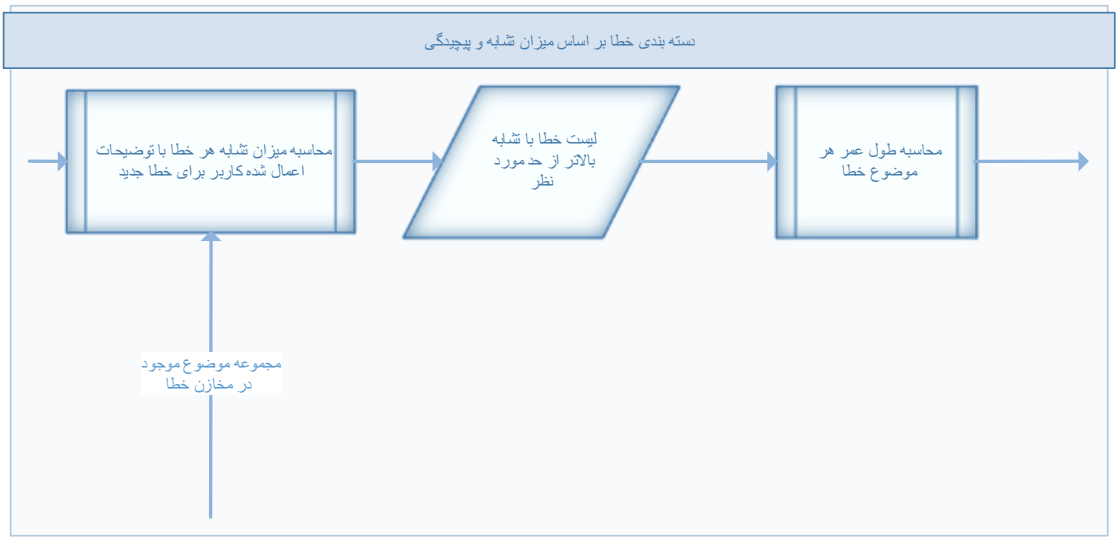
مخازن خطای نرم افزار سالهاست که به کمک مهندسین نرم افزار و مدیران برای پیاده سازی پروژه ها کوچک و بزرگ نرم افزاری ، متن باز یا بسته آمده است. هر خطا در این مخازن به صورت فرآیندی جداگانه برای ثبت تمام رویدادها و اعمال نظرهای منطقی وموثر، درطول رفع مشکل و خطا از روی پروژه، از سوی افراد دخیل در پروژه ثبت می شود. این کار علاوه بر مزایای مربوط به پی-گیری روند رفع خطا ، مجموعه ای از دانش نهفته در این اطلاعات را در اختیار ذینفعان قرار می دهد. اگر این اطلاعات با حساسیت لازم و دقت کافی استخراج شود در روند تکمیل پروژه اثر به سزایی خواهد داشت. از آنجا که این داده ها بیشتر حاوی متون هستند نیاز به الگوریتم ها و معیارهایی برای بررسی متن احساس می شود. بیشتر این مخازن و کارهای قبلی انجام شده از روش های ساده و یا از موتورهای جستجوگر معمول برای استخراج دانش مفید استفاده کرده اند.

یکی از نکاتی که پایه این تحقیق و روش قرار گرفته است، تشابه بین متن توضیحی درباره خطای جدید و متون موجود در موضوع، توضیحات و دیگر متون ثبت شده مربوط به هر خطا است. این تشابه قابل اندازه گیری است و به یقین متن با تشابه بالاتر مربوط به خطا با تشابه بالاتر به خطای جدید است. این خطا شانس برطرف کردن خطای جدید را بالاتر برده و زمان رفع آن و در نتیجه هزینه های مربوط به آن را کاهش می دهد. به خاطر دلایلی که در فصل قبل بیان شد، انتخاب یک روش بر پایه تشابه معنایی ضروری است. در این فصل از روش محاسبه تشابه معنایی بر پایه مجموعه و تشابه خطی برای استفاده در یک بانک خطای نرم افزاری نمونه برای یافتن راه حل خطا و در نهایت محاسبه پیچیدگی خطا با استفاده از محاسبه طول عمر خطا استفاده می شود. شکل ۳-۱ مراحل مختلف روش ارائه شده در این تحقیق را نشان می دهد .



شکل ۳-۱- مراحل مختلف روش ارائه شده

در مرحلـه اول تعـدادی خطـا بـه عنوان نمونـه کوچکی از یـک بانک داده از داده های موجود در Bugzilla انتخاب شـده. خلاصـه، زمان ایجاد و زمان خاتمه (پاسـخ قابل قبول) هر کدام استخراج می شـود. در مرحله بعد میزان تشابه بین نمونه ها با جمله نمونه خطای جدید در پنج مرحله محاسبه و لیست این تشابهات که اعدادی بین ۰ و ۱ هستند استخراج می شود. در مرحله سوم طول عمر خطا ها محاسبه می شود این عدد پیچیدگی آنها را نشان می دهد.(شکل ۳-۲)



شکل ۳-۲- مراحل دسته بندی خطا

و در مرحله نهایی این دو لیسـت داده عددی خوشه بندی می شوند تا بتوان نمونه های منتخب را برای کمک به حل خطا و نیز پیش بینی مقادیر لازم مورد استفاده قرار می گیرند.

۲,۳ . محاسبه تشابه معنایی بر پایه مجموعه و تشابه خطی

Islam در روش خود۳ روش تشـابه را برای برطرف کردن ضـعف آنها ترکیب کرد. این روش روشی متداول در رفع نواقص الگوریتم هاست. برای محاسبه تشابه بین دو جمله یا متن در این روش سـه مرحله کلی وجود دارد. ۱. محاسـبه تشـابه خطی بین کلمات ۲. محاسـبه تشـابه معنایی بر پایه مجموعه ۳. محاسبه تشابه بر اساس کلمات متداول ٤ . محاسبه تشابه مجموع(محاسبه نهایی).

۱,۲,۳ . شباهت خطی بین کلمات

یکی از روش هـای مورد انتظار در این نوع اندازه گیری روش طولانی ترین زیردنباله مشـــترک (LCS)[1]با کمی تغییرات و نرمال سـازی اسـت. سه نسخه اصلاح شده متفاوت از LCS به کار رفته و سـپس وزن های بدست آمده با هم جمع می شوند. به این نکته باید توجه کرد که نسخه های اصلاح شده LCS نتایج بهتری (دقت و جامعیت بالاتر) را نسبت به LCS و یا معیارهای شباهت دیگر خواهند داشت.

$$LCS(X_i,Y_j)=\begin{cases}0 & \text{if } i=0 \text{ or } j=0\\ LCS(X_{i-1},Y_j)+1 & \text{if } x_i=y_j\\ longest(LCS(X_i,Y_{j-1}),LCS(X_{i-1},Y_j)) & \text{if } x_i \neq y_j\end{cases}$$

```
LCS − Length(X ,Y )
1  m ← Length[X ]
2  n ← Length[Y ]
3  for i ← 1 to m
4        do c[i ,0] ← 0
5  for j ← 0 to n
6        do c[0, j] ← 0
7   for i ← 1 to m
8        do for j ← 1to n
9              do if xi = yj
10                 then c[i , j ] ← c[i − 1, j − 1] + 1
11                      b[i , j ] ← "□  "
12                 else if c[i − 1, j ] ≥ c[i , j − 1]
13                         then c[i , j ] ← c[i − 1, j ]
14                              b[i , j ] ← "↑ "
15                         else c[i , j ] ← c[i , j − 1]
16                              b[i , j ] ← "← "
17  return c and b
```

[1] -Allison and Dix 1986

Melamed با تقسیم طولانی ترین زیر رشته مشترک به طول رشته طولانی دیگر سعی کرد که LCS را نرمال سازی کند. وی روش را LCSR نامید. اما در این روش طول رشته های کوتاهتر که گاهی اوقات در محاسبه شباهت دارای ارزش بالاتری بودند، در نظر گرفته نمی شود و این خود ضعف محسوب می شود.

اما در این روش LCS با در نظر گرفتن توامان طول رشته کوتاهتر و بلندتر نرمالسازی می شود(NLCS)[1].

$$v_1 = NLCS(r_i, s_j) = \frac{length(LCS(r_i, s_j))^2}{length(r_i) \times length(s_j)}$$

در حالی که در LCS توالی کاراکترها مهم و ضروری نیست، اما این توالی برای داشتن درجه بالایی از تطبیق در رشته لازم است. این نکته در روش MCLS[2] مورد توجه قرار گرفته است. در این روش ابتدا با توالی حداقل کاراکتر $MCLS_1$ شروع و تا حداکثر کاراکتر MCLSn پیش می رود. الگوریتم ۱ $MCLS_1$ در زیر نشان داده است. همچنین الگوریتم۲ MCLSn ، روند کار را مشخص کرده است.

الگوریتم ۱:MCLS1(LCS باحداکثرتوالی n=1)

input : r_i , s_j , λ /* where $|r_i| = \tau$, $|s_j| = \eta$ *and* $\tau \le \eta$ */

output : r_i

1 $\tau \leftarrow |r_i|$, $\eta \leftarrow |s_j|$

2 *while* $|r_i| \ge 0$ do

3 *if* $r_i \bigcap s_j$ *then* /* i.e., $r_i \subset s_j = r_i$ */

4 *return* r_i

5 *else*

6 $r_i \leftarrow r_i \setminus c_\tau$ / * i.e., remove the right most character from r_i */

7 *end*

9 *end*

[1]-normalized longest common subsequence

[2]-maximal consecutive longest common subsequence starting at character

الگوریتم۲ :MCLSn (LCS با حداکثر توالی n=n)

input : r_i , s_j , λ /* where $|r_i| = \tau$, $|s_j| = \eta$ *and* $\tau \le \eta$ */

output : x /* x is the maximal consecutive LCS starting at any character n */

1 $\tau \leftarrow |r_i|$, $\eta \leftarrow |s_j|$

2 *while* $|r_i| \ge 0$ do

3 determine all n-grams from r_i where $n = 1 \ldots |r_i|$ and

4 $\bar{r}_i$ is the set of n-grams

5 *if* $x \in s_j$ where $\{x | x \in \bar{r}_i$,x=max($\bar{r}_i$)$\}$ *then*

6 *return* x

7 *else*

8 $r_i \leftarrow r_i \setminus x$ / * remove x from $\bar{r}_i$ */

9 *end*

10 *end*

در الگوریتم اول دو رشته به عنوان ورودی گرفته حداکثر رشته متوالی مشترک در هر دو رشته که با کاراکتر اول رشته کوتاه تر آغاز شده باشد به عنوان خروجی برگردانده می شود. اگر الگوریتم ۲ همین کار را برای تک تک کاراکترها تکرار کرده و بزرگترین رشته متوالی از هر دو رشته را به عنوان خروجی باز می گرداند. این دو الگوریتم نیز نرمالسازی می شوند.

$$v_2 = NMCLCS(r_i, s_j) = \frac{length(MCLCS(r_i, s_j))^2}{length(r_i) \times length(s_j)}$$ فرمول ۳-۱

$$v_3 = NMCLCS_n(r_i, s_j) = \frac{length(MCLCS_n(r_i, s_j))^2}{length(r_i) \times length(s_j)}$$ فرمول ۳-۲

مقادیر وزنی بدست آمده از سه قبلی v_1 و v_2 و v_3 با ضرایب وزنی w_1 و w_2 و w_3 که $w_1+w_2+w_3=1$ با هم جمع شده و میزان شباهت دو جمله بدست می آید.

$$\alpha = w_1v_1 + w_2v_2 + w_3v_3$$

در این جا برای W ها ارزش برابر در نظر گرفته می شـــود. به عنوان مثال فرض کنید که روش را روی دو رشته زیر پیاده کنیم.

$ri = fcmfkzyw$
$sj = fcfmfkzny$
$LCS(ri, sj) = fcmfkz$
$MCLCS1(ri, sj) = fc$
$MCLCSn(ri, sj) = mfkz$
$NLCS(ri, sj) = 7^2/(8\times 9) = 0.68$
$NMCLCS1(ri, sj) = 2^2/(8\times 9) = 0.056$
$NMCLCSn(ri, sj) = 4^2/(8\times 9) = 0.22$
$\alpha = w_1v_1 + w_2v_2 + w_3v_3 = 0.33\times 0.68 + 0.33\times 0.056 + 0.33\times 0.2\ 2 = 0.32$ میزان شباهت

۲,۲,۳ .تشابه معنایی بین کلمات

در روش مورد بررســی توجه بر روش های اندازه گیری بر پایه مجموعه اســت، زیرا این روش ها انواع گســترده ای از متون را پوشش می دهند متونی که در دنیای واقعی استفاده می شود در حالی که در روش های بر پایه دانش به این نکته توجه نشده است.

PMI-IR یک روش ســاده برای محاســبه شباهت متنی مجموعه ای از کلمات است که با استفاده از اطلاعات نقطه به نقطه متقابل به صورت زیر تعریف می شود

$$PMI(w_1, w_2) = \log p(w_1 \text{ AND } w_2)/p(w_1)p(w_2)$$

w_1 و w_2 کلمات و $p(w_1 \text{AND} w_2)$ احتمال رخ داد توامان دو واژه است. اگر دو کلمه از نظرآماری مستقل باشند $p(w_1 \text{ AND } w_2) = p(w_1).p(w_2)$ و در غیراینصورت امکان پدیدار شدن همزمان را با هم در متن دارند و $p(w_1 \text{AND } w_2) > p(w_1).p(w_2)$.

PMI-IR توســط موتور جســتجوگر آلتاویستا[1] برای جستجو نحوی و محاسبه احتمالات استفاده می شــود. در ساده ترین حالت دو کلمه هنگامی که در یک سند یافت شوند، احتمال رخ داد همزمان را دارند. این احتمال با بازیابی اسناد قابل اندازه گیری به صورت تقریبی است.

$$PMI-IR(w_1, w_2) = hits(w_1 AND w_2)/hits(w_1)hits(w_2)$$

$hits(x)$ تعداد متونی است که حاوی کلمه x بوده اند.

روش رخ داد همزمان مرتبه دوم PMI(SOC-PMI)[2] لیســتی از کلمات همســایه با دو کلمه مورد نظر در یک مجموعه بزرگ را، بر اساس اطلاعات متقابل نقطه به نقطه مرتب می کند.

مزیت دیگر این روش این اســت که شــباهت میان دو کلمه در هر جای متن نیز می توان اندازه گیری کرد. حتی اگر در کنار هم نباشد. تنها کافی است کلمات همسایه مشابه داشته باشند.

---

[1] -www.altavista.com
[2]- Second Order Co-occurrence PMI (SOC-PMI)

مزیت دیگر این روش (PMI-IR) این است که شباهت میان دو کلمه در هرجای متن نیز می توان اندازه گیری کرد. حتی اگر در کنار هم نباشند. تنها کافی است که کلمات همسایه مشابه داشته باشند.

در این روش از مجموعه ملی بریتانیا (BNC) به عنوان یک منبع برای متون استفاده می شود. کلماتی که در هر دو فهرست مشترک است لیست شده و جمع ارزش PMI (از لیست مقابل) برای محاسبه برای بدست آوردن نسبت ارزش معنایی محاسبه می شود. تابع نقطه به نقطه اطلاعات متقابل برای کلماتی که دارند، تعریف می شود.

$$f^{pmi}(t_i,w)=\log_2\frac{f^{b}(t_i,w)\times m}{f^{t}(t_i)f^{t}(w)}$$ فرمول ۳-۳

$f^{t}(t_i)$ تعداد دفعاتی است که t_i در مجموعه ظاهر می شود. $f^{b}(ti,w)>0$ تعداد دفعاتی را که t_i و w در یک قاب[1] ظاهر می شود را نشان می دهد. m تعداد کل توکن ها در مجموعه است. برای کلمه w مجموعه کلمات X^{w} به صورت زیر تعریف می شود.

$$f^{pmi}(t_1,w)\geq f^{pmi}(t_2,w)\geq\ldots\geq f^{pmi}(t_{\beta-1},w)\geq f^{pmi}(t_\beta,w)$$

قاعده کلی برای β به صورت زیر است

$$\beta=\left(\log\left(f^{t}(w)\right)\right)^{2\frac{\log_2^{(n)}}{\mu}}$$ فرمول ۳-٤

μ یک مقدار ثابت است که در این روش ٦/٥ در نظر گرفته شود. μ با اندازه مجموعه رابطه مستقیم دارد. هر چه μ کوچکتر باشد کلمات مهمتر ممکن است نادیده گرفته شود.

تابع مجموع PMI، β کلمه $w1$ نسبت به کلمات دیگر به صورت زیرتعریف می شود.

$$f(w_1,w_2,\beta)=\sum_{i=1}^{\beta}(f^{Pmi}(X_i^{w_1},w_2))^{\gamma}$$ فرمول ۳-۵

اگر $f^{pmi}(X_i^{w_1},w_2)>0$ تمام مقادیر مثبت PMI برای کلمات مجموعه X^{w_2} در X^{w_1} نیز هستند. به عبارت دیگرکلماتی که از نظر معنایی به w_2 نزدیک باشند. مطمئنا در لیست کلمات هم معنی $sim(w_1,w_2)=\frac{f^{\beta}(w_1)}{\beta_1}+\frac{f^{\beta}(w_2)}{\beta_2}=\frac{44.255}{23}+\frac{22.364}{19}=3.10$ نیز هستند.

مقدار γ[2] باید مقداری بزرگ تر ازیک باشد. پس تابع مجموع $\beta-PMI$ برای w_1 نسبت به w_2 با $\beta=\beta_1$ برابرو برای w_2 نسبت به w_1 با ۳ برابر $\beta=\beta_2$. در نهایت تابع شباهت معنایی PMI بین دو کلمه w_1 و w_2 به صورت زیر تعریف می شود[۲۳].

---

[1]-در اینجا قاب ۱۱ کلمه فرض شده است

[2]-هر چه این مقدار بیشتر باشد توجه به کلمات با ارزش بیشتر است. در آزمایشهای متوالی در این روش به این نتیجه رسیده اند که $\gamma=3$ مقدار مناسب است. $\gamma\geq4$ میزان تاکید بیهوده و تاخیر زیاد را به همراه داشت و کلمات کم

$$sim(w_1,w_2)=\frac{f(w_1,w_2,\beta_1)}{\beta_1}+\frac{f(w_1,w_2,\beta_2)}{\beta_2}$$ فرمول ۳-۶

الگوریتم ۳ نحوه نرمال سازی مقادیر بدست آمده از $sim()$ را نشان می دهد.

الگوریتم ۳. نرمالسازی محاسبات شباهت معنایی

input : set D, distance measure dist, number k of cluster
output : A partitioning P of the set D of documents
1 choose randomly k data points from D as starting centroids $t_{\vec{p}_1} \ldots t_{\vec{p}_k}$
2 repeat
3 Assign each point of P to the closest centroid with respect to dist
4 (Re-) calculate the cluster centroids $t_{\vec{p}_1} \ldots t_{\vec{p}_k}$ of clusters $P_1 \ldots P_k$
5 until cluster centroids $t_{\vec{p}_1} \ldots t_{\vec{p}_k}$ are stable
6 return set $\mathbf{P} := \{P_1 \ldots P_k\}$ of clusters.

مثال زیر روند کار را توضیح می دهد.

با استفاده از روش توضیح داده در بالا میزان تشابه معنایی دو کلمه car و automobile در ۱۲ جمله جدول۳-۱ محاسبه می شود.

جدول ۳-۱- ۱۲ جمله به عنوان نمونه داده

1	pursuit accident claim car driver exclude
2	soak motorist company car driver risky
3	company car driver tend travel farther
4	job engineer disappear fall mechanical engineer car industry worst affect
5	Sign recession car industry
6	brightest engineer moment car industry
7	yugoslavia benefit direct investment automobile industry
8	acreage expand emergence automobile industry
9	automobile industry among hardest hit recession
10	automobile industry largely male force
11	component supplier automobile industry expand

ارزش تر نادیده گرفته می شود Islam و Inkpen در بخش کوچکی از BNC پس از تکرار آزمایش تعداد مناسب β ، μ و γ را بدست آوردند.

12	client industry manufacturer component automobile industry

$f^t(t_i)$ تعداد تکرار کلمات اصلی محاسبه می شود.(جدول۳– ۲)

جدول ۳– ۲ - تعداد تکرار کلمات اصلی

t_i	$f^t(t_i)$	t_i	$f^t(t_i)$
disappear	1	worst	1
yugoslavia	1	soak	1
Pursuit	1	fall	1
brightest	1	supplier	1
travel	1	company	2
Benefit	1	recession	2
risky	1	farther	1
Sign	1	car	6
male	1	investment	1
accident	1	industry	10
affect	1	force	1
mechanical	1	job	1
claim	1	client	1
among	1	Tend	1
moment	1	hardest	1
engineer	3	component	2
automobile	6	manufacturer	1
emergence	1	expand	2
direct	1	driver	3
hit	1	exclude	1
Largely	1		

در مرحله بعد $f^b(ti,w)$ برای همه t_iها برای دو کلمه مورد نظر محاسبه می شود. قاب با طول ۱۱ کلمه(±۵) برای این در نظر گرفته شده است.(جدول۳– ۳)

جدول ۳-۳ - محاسبه $f^b(ti,w)$ برای همه کلمات جدول ۳-۲

t_i	$f^b(t_i, W_1)$	t_i	$f^b(t_i, W_2)$
brightest	1	acreage	1
accident	1	Emergence	1
affect	1	direct	1
motorist	1	hit	1
disappear	1	largely	1
worst	1	yugoslavia	1
pursuit	1	supplier	1
soak	1	benefit	1
fall	1	recession	1
travel	1	male	1
risky	1	Investment	1
recession	1	industry	7
company	2	force	1
sign	1	client	1
farther	1	hardest	1
mechanical	1	Component	2
claim	1	expand	2
tend	1	manufacturer	1
industry	3	Among	1
moment	1		
engineer	3		
exclude	1		
driver	3		

$f^b(\text{‘engineer’},w_1) = 3$ $f^b(\text{‘engineer’},w_2) = 7$

تابع اطلاعات متقابل نقطه به نقطه (PMI) برای t_i ها یی که $0 < f^b(t_i,w)$ بر اســاس فرمول۳-۳ محاسبه می شود. (جدول۳-٤)

جدول ۳- ٤ -مفدار تابع اطلاعات متقابل نقطه به نقطه برای تمامی مقادیر با $0 < f^{b}(t_i, w)$

X_i (also t_i)	$f^{pmi}(t_i, W_1)$	Y_i (also t_i)	$f^{pmi}(t_i, W_2)$
motorist	3.544	emergence	3.544
disappear	3.544	direct	3.544
worst	3.544	acreage	3.544
pursuit	3.544	hit	3.544
soak	3.544	largely	3.544
travel	3.544	yugoslavia	3.544
brightest	3.544	supplier	3.544
fall	3.544	benefit	3.544
risky	3.544	male	3.544
company	3.544	investment	3.544
sign	3.544	among	3.544
farther	3.544	force	3.544
accident	3.544	client	3.544
affect	3.544	hardest	3.544
mechanical	3.544	component	3.544
tend	3.544	manufacturer	3.544
claim	3.544	expand	3.544
engineer	3.544	industry	3.029
moment	3.544	recession	2.544
driver	3.544		
exclude	3.544		
recession	2.544		
industry	1.807		

β_1 و β_2 طبق فرمول ۳-۳ برای محاسب نهایی محاسبه می شود.

$\beta_1 = 24.88 \quad \beta_2 = 24.88$

μ در اینجا ۰,۷ در نظر گرفته شده است. γ نیز برای محاسبات همان مقدار بهینه ۳ در نظر گرفته می شـود. دو کلمه recession و industry هر دو جز مجموعه Y_iو X_i هسـتند پس طبق فرمول۳-٤ و ۵-۳

$$f^{\beta}(W_1) = \left(f^{pmi}(\text{“recession”},W_2)\right)^{\gamma} + \left(f^{pmi}(\text{“industry”},W_2)\right)^{\gamma} = (2.544)^3 + (3.029)^3 = 44.255$$

$$f^{\beta}(W_2) = \left(f^{pmi}(\text{“industry”},W_1)\right)^{\gamma} + \left(f^{pmi}(\text{“recession”},W_1)\right)^{\gamma} = (1.807)^3 + (2.544)^3 = 22.364$$

$$\text{sim}(w_1,w_2) = \frac{f^{\beta}(w_1)}{\beta_1} + \frac{f^{\beta}(w_2)}{\beta_2} = \frac{44.255}{23} + \frac{22.364}{19} = 3.10$$

۳,۲,۳ . تشابه جملات بر اساس عبارات مشترک

اگر دو متن کلمات مشـابه با هم داشـته باشند می توان میزان مشابهت آنها را بر اساس این کلمات مشـترک بدسـت آورد (این کلمات یا همگی در یک موقعیت مکانی هسـتند، و یا در موقعیت های مختلف در جملات ظاهر می شوند). هستینگز[۲٦] معتقد بود که این کلمات ارزش چندانی در اندازه گیری شـباهت معنایی جملات کوتاه ندارد. در این روش برای آنکه از اهمیت کم این معیار چشـم پوشی نشود فاکتور وزنی آن را کمتر از ۰/۵ در نظر می گیریم .

روش کار به این صـورت اسـت که ، فرض کنیم P و R دو جمله با کلمات مشابه هستند و|R|<|P|، تمام کلمات مشـابه در دو جمله اسـتخراج می شـوند. اگر X مجموع کلمات مشابه درP و Y مجموع کلمات مشابه در R باشد هر کدام از این مجموعه ها ترتیب مخصوص به خود دارند. از آنجا که تعداد کلمات P بیشـتر ازR اسـت و در اصـطلاحP بزرگتر ازR اسـت. به کلمات موجود در مجموعه X به ترتیب موقعیتشـان در جمله وزنی از۱ تا δ اختصـاص می دهیم و همین وزن به کلمات مجموعه Y الحاق می شود. سپس میزان شباهت این جمله بر اساس دستور زیر محاسبه می شود.

$$\mathbf{x} = \{x_1, x_2, \ldots, x_\delta\}$$

$$\mathbf{Y} = \{y_1, y_2, \ldots, y_\delta\}$$

$$S_o = \frac{1 - |x_1 - y_1| + |x_2 - y_2| + \ldots + |x_\delta - y_\delta|}{|1 - x_\delta| + |x_2 - x_{\delta-1}| + \ldots + |x_\delta - x_1|}$$

$$S_o = \begin{cases} 1 - \dfrac{2\sum_{i=1}^{\delta} |x_i - y_j|}{\delta^2} & \text{if } \delta \text{is even} \\ 1 - \dfrac{2\sum_{i=1}^{\delta} |x_i - y_j|}{\delta^2 - 1} & \text{if } \delta \text{is and } \delta > 1 \\ 1 & \text{if } \delta \text{is and } \delta = 1 \end{cases}$$

فرمول ۳-۷

به عنوان مثال دو جمله را در نظر بگیرید

P: Einstein was a German-born theoretical physicist

R: The theoretical physicist – Einstein lived at 19 century

X = {Einstein, theoretical, physicist} X= {1, 2, 3}

Y= {theoretical, physicist, Einstein} Y= {2, 1, 3}

با توجه به فرمول ۳-۷ $S_o = 1 - \frac{2 \times 2}{9} = \frac{5}{9} = 0.55$

٤,٢,٣ . شباهت کلی جملات

Islam روش خود را در٦ مرحله ارائه خلاصه کرده است:

۱- ابتدا تمام کلمات اضافه در جملات برای پیدا کردن کلمات کلیدی پاک می شود اگر P و R دو جمله مورد نظر باشند m کلمه از P و n کلمه از R کلمات اصلی ما را تشکیل می دهند. (حذف حروف و کلمات اضافه)

$$n \geq m \qquad P = \{p_1, p_2, ..., p_\delta\} \qquad R = \{r_1, r_2, ..., r_\delta\}$$

۲- در این مرحله کلمات مشابه در این مجموعه علامت گذاری می شوند. δ کلمه مشابه درR وP کنار گذاشته شده و بقیه برای بررسی نگه داشته می شود. اگر m=δ بود به مرحله ٦ میرویم. در غیراین صورت δ<m به ترتیب ادامه میدهیم.

$$P = \{p_1, p_2, ..., p_{m-\delta}\} \qquad R = \{r_1, r_2, ..., r_{n-\delta}\}$$

۳- ماتریس تشابه خطی دو مجموعه $a(m-\delta)\times(n-\delta)$ به صورت زیرتشکیل می شود. هر α_{ij} در a به صورت زیر محاسبه می شود :

اگر $p_i \in P$، τ کاراکتر داشته باشد $p_i = \{c_1, c_2, ..., c_\tau\}$ و هر $r_j \in R$، η کاراکتر $r_j = \{r_1, r_2, ..., r_\eta\}$ و $\tau \leq \eta$ به صورتی که τ طول کوتاه ترین کلمه و η بلندترین کلمه مشترک است[۲۳] .

$$v_1 \leftarrow \mathrm{NLCS}(p_i, r_j)$$

$$v_3 \leftarrow \mathrm{NMCLCSn}(p_i, r_j)$$

$$v_2 \leftarrow \mathrm{NMCLCS1}(p_i, r_j)$$

فرمول ۳- $\alpha_{ij} \leftarrow w_1 v_1 + w_2 v_2 + w_3 v_3$

۸

$$w^1 + w^2 + w^3 = 1$$

$$M_1 = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1j} & \dots & \alpha_{(n-\delta)} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2j} & \dots & \alpha_{2(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_{i1} & \alpha_{i2} & \dots & \alpha_{ij} & \dots & \alpha_{i(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_{(m-\delta)1} & \alpha_{(m-\delta)2} & \dots & \alpha_{(m-\delta)j} & \dots & \alpha_{(m-\delta)(n-\delta)} \end{pmatrix}$$

مثال زیر این مرحله را توضیح می دهد

r_j ="allmileage_make_maxkm"

p_i ="make_minmile_distance_possible_take"

اجرا مرحله اول

r_j = {all, mileage, make, max, km} m=5

p_i = {make, min, mile, distance, possible, take} n=6

اجرا مرحله دوم

r_j = {all, mileage, max, km}

p_i = {min, mile, distance, possible, take}

	Min	*mile*	*distance*	*possible*	*take*
all	0	0.055	0.041	0.027	0.082
mileage	0.188	0.565	0.058	0.076	0.058
max	0.11	0.082	0.027	0	0.055
km	0.11	0.082	0	0	0.123

به عنوان نمونه میزان شباهت بین possible وmileage اینگونه محاسبه شده:

$v_1 = 3^2 / (8\times 7) = 0.16$

$v_2 = 0$

$v_3 = 2^2 / (8\times 7) = 0.071$

مطابق فرمول ۳-۸ $\alpha_{24} = 0.33 \times v_1 + 0.33 \times v_2 + 0.33 \times v_3 = 0.076$

٤- ماتریس $M_2 = (\beta_{ij})_{(m-\delta)\times(n-\delta)}$ ، ماتریس تشـــابه معنایی از روی الگوریتم ۳ تشـــکیل می شود.

$$M_2 = \begin{pmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1j} & \dots & \beta_{(n-\delta)} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2j} & \dots & \beta_{2(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \beta_{i1} & \beta_{i2} & \dots & \beta_{ij} & \dots & \beta_{i(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \beta_{(m-\delta)1} & \beta_{(m-\delta)2} & \dots & \beta_{(m-\delta)j} & \dots & \beta_{(m-\delta)(n-\delta)} \end{pmatrix}$$

۵- ماتریسM از طریق دو ماتریس بالابه صورت زیرتشکیل می شود. $\mathrm{M} \leftarrow \psi \mathrm{M}_1 + \varphi \mathrm{M}_2$

ψ فاکتور وزنی شباهت خطی و φفاکتور وزنی شباهت معنایی است. که $\varphi + \psi = 1$. هر کدام از این ضرایب اگر صفر در نظر گرفته شوند ماتریس مربوطه در محاسبه ماتریس مجموع M لحاظ نمی شود. اگر این ضرایب هر دو ۰,۵ باشند یعنی شباهت خطی و معنایی هر دو به یک میزان اهمیت دارند. عنصر با ماکزیمال ارزش γ_{ij}، در ماتریسM استخراج می شود.

اگر این مقدار بزرگتراز صفربود به لیست ρ اضافه می شود، $\rho \bigcup \gamma_{ij} \leftarrow \rho$. سطر و ستون مربوط به آن از ماتریس حذف می شود. این کار تا زمانی که $\gamma_{ij} = 0$ و یا $m - \delta - |\rho| = 0$ ادامه پیدامی کند.

٦- تمام مقادیر ρ و $\delta(1-w_f + w_f S_o)$ هم جمع می شوند . طبق فرمول $S(P,R)$ میزان شباهت چند گانه با ضرب در مجموع m و n و تقسیم بر دو برابر ضریب آنها به مقداری بین صفر و یک بالانس می شوند. در این روش می توان از هرکدام از روش ها و مقادیر شباهت صرفه نظر کرد. مثلا از معیار شباهت کلمات متداول با صفر اعلام کردن S_o صرفه نظر کرد و یا اینکه با صفر در نظر گرفتن w_f اهمیت نحوی را نادیده گرفت[۲۳] .

۵,۲,۳ . نمونه توضیح روند کار

دو تکه متن مورد نظر برای مقایسه و محاسبه ضریب شباهت به روش فوق با R وP مشخص شده است.

$$R = \text{“}allmileage\_make\_maxkm\text{”}$$

$$P = \text{“}make\_minmile\_distance\_possible\_take\text{”}$$

مرحله ۱: جدا کردن کلمات اصلی و مشخص

m=5 $R = \{all, mileage, make, max, km\}$

n=6 $P = \{make, min, mile, distance, possible, take\}$

مرحله ۲: کلمه هایی که در هر دو مجموعه مشترک است حذف می شوند.

$$R = \{all, mileage, make, max, km\}$$

$$P = \{make, min, mile, distance, possible, take\}$$

مرحله ۳ : ماتریس تشابه خطی با محاسبه تشابه خطی تک تک کلمات بر اساس فرمول ۳-۸ تشکیل می شود.

$$\mathrm{M}_1 = \begin{array}{c} \\ \text{all} \\ \text{mileage} \\ \text{max} \\ \text{km} \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{Min} & \textit{mile} & \textit{distance} & \textit{possible} & \textit{take} \end{array} \\ \begin{bmatrix} 0 & 0.055 & 0.041 & 0.027 & 0.082 \\ 0.188 & 0.565 & 0.058 & 0.076 & 0.058 \\ 0.11 & 0.082 & 0.027 & 0 & 0.055 \\ 0.11 & 0.082 & 0 & 0 & 0.123 \end{bmatrix} \end{array}$$

مرحله ٤ :ماتریس تشابه معنایی نیز تشکیل می شود.

$$\mathrm{M}_2 = \begin{array}{c} \\ \text{all} \\ \text{mileage} \\ \text{max} \\ \text{km} \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{Min} & \textit{mile} & \textit{distance} & \textit{possible} & \textit{take} \end{array} \\ \begin{bmatrix} 0.172 & 0.233 & 0.48 & 0 & 0.813 \\ 0.587 & 0.976 & 0.826 & 0 & 0.558 \\ 0.199 & 0.194 & 0.141 & 0 & 0.243 \\ 0.67 & 0.962 & 0.89 & 0 & 0.408 \end{bmatrix} \end{array}$$

مرحله ٥ : ماتریس M از طریق دو ماتریس قبل به کمک فرمول زیر تشکیل می شود.

$$\mathrm{M} \leftarrow \psi \mathrm{M}_1 + \varphi \mathrm{M}_2$$

$$\mathrm{M} = \begin{array}{c} \\ \text{all} \\ \text{mileage} \\ \text{max} \\ \text{km} \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{Min} & \textit{mile} & \textit{distance} & \textit{possible} & \textit{take} \end{array} \\ \begin{bmatrix} 0.086 & 0.144 & 0.26 & 0.013 & 0.447 \\ 0.388 & 0.771 & 0.442 & 0.038 & 0.308 \\ 0.154 & 0.138 & 0.084 & 0 & 0.194 \\ 0.39 & 0.522 & 0.445 & 0 & 0.266 \end{bmatrix} \end{array}$$

سپس مقدار ماکزیمم به مجموعه ρ اضافه می شود و سطر و ستون آن حذف می شود. این کار تا زمانی که $\gamma_{ij} = 0$ و یا $m - \delta - |\rho| = 0$ ادامه پیدا می کند.

$$\mathrm{M} = \begin{array}{c} \\ \text{all} \\ \text{mileage} \\ \text{max} \\ \text{km} \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{Min} & \textit{mile} & \textit{distance} & \textit{possible} & \textit{take} \end{array} \\ \begin{bmatrix} 0.086 & 0.144 & 0.26 & 0.013 & 0.447 \\ 0.388 & 0.771 & 0.442 & 0.038 & 0.308 \\ 0.154 & 0.138 & 0.084 & 0 & 0.194 \\ 0.39 & 0.522 & 0.445 & 0 & 0.266 \end{bmatrix} \end{array}$$

$$\rho = \{0.771\}$$

$$\mathrm{M} = \begin{array}{c} \\ \text{all} \\ \text{max} \\ \text{km} \end{array} \begin{array}{c} \begin{array}{cccc} \textit{Min} & \textit{distance} & \textit{possible} & \textit{take} \end{array} \\ \begin{bmatrix} 0.086 & 0.26 & 0.013 & 0.447 \\ 0.154 & 0.084 & 0 & 0.194 \\ 0.39 & 0.445 & 0 & 0.266 \end{bmatrix} \end{array}$$

$$\rho = \{0.771, 0.447\}$$

$$\mathrm{M} = \begin{array}{c} \\ \text{max} \\ \text{km} \end{array} \begin{array}{c} \begin{array}{ccc} \textit{Min} & \textit{distance} & \textit{possible} \end{array} \\ \begin{bmatrix} 0.154 & 0.084 & 0 \\ 0.39 & 0.445 & 0 \end{bmatrix} \end{array}$$

$$\rho = \{0.771, 0.447, 0.445\}$$

$$Min \quad distance$$
$$M = \max[0.154 \quad 0]$$
$$\rho = \{0.771, 0.447, 0.445, 0.154\}$$

مرحله ٦: و در نهایت محاسبه ضریب تشابه ترکیبی برای دو جمله R و S

$$SimilarityScore(R_i, S_j) = \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) \times (m+n)}{2mn}$$ فرمول۳-۹

$$= (1 + 1.817) \times 11/60 = 0.516$$

۳,۳. خوشه بندی داده ها

۱,۵,۲. الگوریتم K-means

با توجه به [۲۷] الگوریتم K-means یک الگوریتم ساده تکرار شونده است، که مجموعه داده ها را به K خوشه تقسیم می کند. این الگوریتم در سالهای مختلف در زمینه های گوناگون توسط افرادی مانند LIoyd (۱۹۸۲،۱۹۵۷)، Forgey (۱۹٦٥)، Friedman و Rubin (۱۹٦۷) و McQueen (۱۹٦۷) توسعه یافته است. الگوریتم رویبک مجموعه از بردار های d بعدی تعریف می شود، که $D = \{\mathbf{xi} \mid i = 1, ..., N\}$ و $\mathbf{xi} \in |R^d$ نشان دهنده i امین نقطه داده است. تکنیک مورد استفاده برای انتخاب نمونه های اولیه برای تشکیل K دسته، شامل نمونه گیری تصادفی از مجموعه داده هاست. سپس الگوریتم دو قدم زیر را به صورت مکرر انجام می دهد.

- **گام اول، تخصیص داده:**

هر نقطه داده ها به نزدیک ترین مرکز جرم اختصاص داده می شود. اختصاص داده با استفاده از یک سری روابط قراردادی و دلخواه انجام می گیرد. این فرآیند منجر به دسته بندی داده ها می شود.

- **گام دوم، جابجایی دسته ها:**

نمایندگی در دسته به مرکزی ترین نقطه در دسته اختصاص داده می شود. اگر میزان احتمال داده ها یک اندازه باشد. پس جابجایی ها به اندازه انتظار یعنی به اندازه میانگین وزنی خواهد بود. با توجه به این مطلب الگوریتم اگر انتساب داده ها ارزش Cj در معادله تغییر زیادی نداشته باشد، همگرا خواهد بود.

$$\sum_{i=1}^{n} \left(\underset{j}{argmin} \left\| \mathbf{xi} - \mathbf{cj} \right\|_2^{2|} \right)$$

اصول الگوریتم در روند زیر دنبال شده است.

الگوریتم ٤ : الگوریتم K-means

input : set D, distance measure dist, number k of cluster
output : A partitioning P of the set D of documents
1 choose randomly k data points from D as starting centroids $t_{\vec{p}_1} \ldots t_{\vec{p}_k}$
2 repeat
3 Assign each point of P to the closest centroid with respect to dist
4 (Re-) calculate the cluster centroids $t_{\vec{p}_1} \ldots t_{\vec{p}_k}$ of clusters $P_1 \ldots P_k$
5 until cluster centroids $t_{\vec{p}_1} \ldots t_{\vec{p}_k}$ are stable
6 return set $\mathbf{P} := \{P_1 \ldots P_k\}$ of clusters.

یک مسـئله مهم در اسـتفاده از الگوریتم، مسـئله تعیین و اندازه گیری کمیت میزان نزدیکی، در مرحله واگذاری داده به دسته ها است. به طور پیش فرض اندازه گیری میزان نزدیکی، فاصله اقلیدسی در نظر گرفته می شـود. الگوریتم معمولا بسیار حساس به مرکز جرم است. انتخاب مرکز جرم مجدد برای گروه، مشکل واگرایی را برای داده های پراکنده حل می کند.

منابع و مآخذ

١. Naresh Kumar Nagwani, Pradeep Singh/"Bug Mining Model Based on Event-ComponentSimilarity to Discover Similar and Duplicate GUIBugs"/ 2009 IEEE International Advance Computing Conference (IACC 2009)/Patiala, India/6-7 March 2009.

٢. Bergroth, L. Dept. of Comput. Sci., Turku Univ., Finland ; Hakonen, H. ; Raita,T/"A survey of longest common subsequence algorithms"/String Processing and Information Retrieval,Proceedings/ Seventh International Symposium on/ SPIRE 2000.

٣. Islam, A. and Inkpen, D. /" Second Order Co-occurrence PMI for Determining the Semantic Similarity of Words"/ in Proceedings of the International Conference on Language Resources and Evaluation (LREC)/ Genoa, Italy/ pp. 1033–1038/ 2006.

٤. Naresh Kumar Nagwani, ShrishVerma/ "Predicting Expert Developers for Newly Reported Bugs Using Frequent Terms Similarities of Bug Attributes"/ 2011 Ninth International Conference on ICT and Knowledge Engineering / 2011 IEEE.

٥. Hui Zeng, David Rine/ " Estimation of Software Defects Fix Effort Using Neural Networks"/ Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)/ 2004 IEEE.

٦. JunzoWatada/ "Analysis of Software Reliability by Fuzzy Regression Model"/ TENCON 2000

٧. Lucas D. Panjer/ " Predicting Eclipse Bug Lifetimes"/ Fourth International Workshop on Mining Software Repositories (MSR'07)/ 2007 IEEE.

٨. Suma.V, Pushpavathi T.P, and Ramaswamy.V/"An Approach to Predict Software Project Success by Data Mining Clustering"/ International Conference on Data Mining and Computer Engineering (ICDMCE'2012)/ Bangkok,(Thailand)/ December 21-22, 2012.

٩. CathrinWeiß ,Thomas Zimmermann, Rahul Premraj , Andreas Zeller ,Saarland University/ " How Long will it Take to Fix This Bug?"/ Fourth International Workshop on Mining Software Repositories (MSR'07)/ 0-7695-2950-X/07 $20.00 © /2007 IEEE

١٠. Naresh Kumar Nagwani, ShrishVerma/" Predictive Data Mining Model for Software Bug Estimation Using Average Weighted Similarity"/ Advance Computing Conference (IACC)/ 2010 IEEE 2nd International.

١١. Naresh Kumar Nagwani, Ashok Bhansali/"A Data Mining Model to Predict Software Bug Complexity Using Bug Estimation and Clustering"/ 2010 International Conference on Recent Trends in Information/ Telecommunication and Computing/ 2010 IEEE.

١٢. Naresh Kumar Nagwani, ShrishVerma/ "Predicting Expert Developers for Newly Reported Bugs Using Frequent Terms Similarities of Bug Attributes"/ 2011 Ninth International Conference on ICT and Knowledge Engineering / 2011 IEEE.

١٣. B. Berliner/ “Parallelizing software development In Proceedings of the USENIX”/ Winter 1990Technical Conference/ volume 341, page 352, 1990

١٤. C. M. Pilato, B. Collins-Sussman, and B. W. Fitzpatrick/ “Version Control with Subversion”/ O’ReillyMedia/ 2008

١٥. www.sourceforge.net

١٦. www.code.google.com

١٧. Stephen W. Thomas/” Mining Software Repositories with Topic Models”/ School of Computing Queen’s University Kingston, Ontario, Canada. Technical Report 2012-586. IEEE.

١٨. Golnoosh Abaee, Department of Studies in Computer Science,Islamic Azad University, Roodehen Branch, Iran. D.S.Guru, Department of Studies in Computer Science, University of Mysore, Manasagangothri, ,Mysore,570006, India/ ”Enhancement of Bug Tracking Tools; the Debugger”/ 2nd International Conference on Software Technology and Engineering(ICSTE)/ 2010.

١٩. Bug life cycle / www.softwaretestinghelp.com/bug-life-cycle/

٢٠. Naresh Kumar Nagwani, Dr. Shrish Verma1Assistant Professor, Computer Science & Engineering; 2Associate Professor & Head, Electronics & Tel. Communication Engg. Mm National Institute of Technology Raipur,1nknagwani.cs@nitrr.ac.in, 2shrishverma@nitrr.ac.in/ On Studying the Effect of Sample Size in Evaluation of Bug Classifiers/ ISSN: 0974-6846, Vol: 6 Issue: 1/ January 2013

٢١. Wael H. Gomaa,Computer Science Department Modern Academy for Computer ,Science & Management Technology Cairo, Egypt.Aly A. Fahmy,Computer Science Department Faculty of Computers and Information, Cairo University Cairo, Egypt/ ”A Survey of Text Similarity Approaches”/ International Journal of Computer Applications/ (0975 – 8887) Volume 68– No.13/ April 2013.

٢٢. Li, Y., McLean, D., Bandar, Z., O’Shea, J., & Crockett, K/ ”Sentence similarity based on semantic nets and corpus statistics”/ IEEE Transactions on Knowledge and Data Engineering/ 18(8), 1138–1149 / 2006.

٢٣. Islam, A., &Inkpen, D/ ”Semantic text similarity using corpus-based word similarity and string similarity”/ ACM Transactions on Knowledge Discovery from Data/ 2(2), 1–25/ 2008.

٢٤. Nitish, A., Kartik, A. & Paul, B. DERI&UPM/” Pushing Corpus Based Relatedness to Similarity: Shared Task System Description”/ First Joint Conference on Lexical and Computational Semantics (*SEM)/ pages 643–647/ Montreal, Canada, Association for Computational Linguistics/ June 7-8, 2012.

٢٥. Davide, B., Ronan, T. Nathalie A.&Josiane, M. (2012), IRIT/ ”Textual Similarity Combining Conceptual Similarity with an N-Gram Comparison Method”/ First Joint Conference on Lexical and Computational Semantics (*SEM)/ pages 552–556/ Montreal, Canada/June 7-8, 2012 Association for Computational Linguistics.

٢٦. Naresh Kumar Nagwani, Dr. Shrish Verma1Assistant Professor, Computer Science & Engineering; 2Associate Professor & Head, Electronics & Tel. Communication Engg. Mm National Institute of Technology Raipur,1nknagwani.cs@nitrr.ac.in, 2shrishverma@nitrr.ac.in/ On Studying the Effect of Sample Size in Evaluation of Bug Classifiers/ ISSN: 0974-6846, Vol: 6 Issue: 1/ January 2013

٢٧. XindongWu , Vipin Kumar , J. Ross Quinlan , Joydeep Ghosh , Qiang Yang , Hiroshi Motoda , Geoffrey J .McLachlan , Angus Ng , Bing Liu , Philip S. Yu , Zhi-Hua Zhou , Michael Steinbach , David J.Hand , Dan Steinberg / “Top 10 algorithms in data mining/ Springer-Verlag London Limited ”/ Published online: 4 December 2007.

٢٨. Osama Abu Abbas ,Computer Science Departmentm/”*Comparisons Between Data Clustering Algorithms”/*The International Arab Journal of Information Technology/ Vol. 5, No. 3,/July 2008.

٢٩. Rui Xu, Student Member, IEEE and Donald Wunsch II/”Survey of Clustering Algorithms”/ IEEE Transactions on neural networks, VOL. 16, NO. 3, May 2005