یکپارچه سازی داده های XML فازی ناهمگن بر اساس شباهت ساختاری و معنایی

چکیده

انضمام داده های وب به یک نیاز اساسی برای مدیریت داده های وب تبدیل شده است.تعداد قابل توجهی رویکرد برای یکپارچه سازی داده های زبان تعمیم یافته ی نشانه گذاری XML از منابع اطلاعاتی ناهمگن بدست آمده است.البته این رویکردهای هیجان انگیز هنوز مناسب اتحاد اطلاعات XML فازی نیستند به دلیل مشخصات فازی که دارند.در این مقاله،ما یک چهارچوب برای مقابله با اتحاد اسناد XML فازی آماده کرده ایم. اولا، ما یک مدل درختی جدید برای XML فازی تهیه کردیم. ثانیا، ما الگوریتم موثر بر اساس فاصله ویرایش شده ی درخت برای شناسایی ساختار و شباهت های معنایی بین اسناد فازی حاضر در مدل درختی XML فازی آماده کرده ایم. ثالثا، ما یک استراتژدی اتحاد که برای یکپارچه سازی اسناد فازی از منابع داده های مختلف استفاده شده است معرفی کرده ایم. و در آخر ما آزمایش هایی برای مشخص نمودن رویکردمان که می تواند به طور موثر اسناد XMLفازی را متحد کند انجام داده ایم.

مقدمه

با پیشرفت دائم و سریع اینترنت ، XML نیز در راستای استانداردهای واقعی برای نمایش افزایش مقدار اطلاعات در دسترس بر روی وب، پیشرفت کرده است.مزیت اصلی XML باز بودنش بر اساس استانداردها و ویژگی ها می باشد. منعطف بودنش اجازه می دهد اسناد بسط پیدا کنند برای توصیف محتوا در هر سایزی. خیلی از ارگانها و سازمانها از XML به عنوان یک توصیف اطلاعاتی و یا مدل مخزن مانند در کاربردهای بر اساس وب خود استفاده می کنند. تقاضا برای انضمام اطلاعاتی سیستم های مختلف درخواستی آنلاین به طور سریعی با همکاری موسسات افزایش پیدا کرده است. انضمام داده های XML یک امر حیاتی و سختی است به دلیل پیچیدگی و انعطاف پذیری مشخصاتش. داده ی XML انضمام بخشیده می شود با مستقر شدن در منابع اطلاعاتی ناهمگن و توزیع شده. در نتیجه، اطلاعات در منابع مختلف ممکن است نمایش های مختلف و یا شاید یک مرجع یکسان در دنیای واقعی داشته باشند. از این رو اطلاعات از منابع اطلاعاتی مختلف بالقوه تناقض ها و همپوشانی هایی دارند.

این انضمام داده های XML را سخت تر می کند. انضمام داده های XML به این معنا نیست که مستقیما منابع اطلاعاتی XML را در کنار یکدیگر قرار بدهیم . برای موفقیت در یکپارچه سازی اطلاعات XML نیاز است تا تناقضها و سپس همپوشان ها را از اسناد شناسایی و حذف کنیم.

یکپارچه سازی اطلاعات XML توجه خوبی را از هم جامعه آکادمیک و هم جامعه صنعتی به دلیل کاربردهای زیادش دریافت کرده است. یک رویکرد برای یکپارچه سازی اطلاعات XML این است که یک انبار اطلاعاتی بسازیم که یک ظرف اطلاعاتی از انضمام و یکچارچگی از منابع اطلاعاتی ناهمگن تولید کند.

هدف اصلی آن است که پروسه ی شناسایی و تطبیق تناقض ها و همپوشانی ها ازمقدارهای استخراج شده XML محلی را آسان بنماید. از انجا که ساختار اسناد XML ناهمگن است یک تناقض در ساختار می تواند باعث سختی انضمام سازی شود.

رویکردهای فعلی یکپارچه سازی فرض می کنند که اطلاعات XML اشیاء مشخص در جهان واقعی را مشخص می کنند. در واقع در خیلی از کاربرد های عملی اطلاعات فازی هستند. در پروسه ی اطلاعات XML فازی تلاش هایی شده است. مدل هایی از XML فازی آماده برای نمایش اطلاعات فازی شده اند. بگذارید به چندتا مثال نگاهی بیندازیم. فرض کنید ما از موقعیت آکادمیک فردی به نام جورج اطلاعاتی نداریم ممکن است موقعیتش استادیار باشد با درجه عضویت 1.0 و استاد با درجه عضویت 0.7 . سپس ما میتوانیم از XML فازی زیر استفاده کنیم:

<name>George</name>

<Dist Type = “disjunctive”>

<Val Poss = 1.0>

<position>Associate Professor</position>

</Val>

<Val Poss = 0.7>

<position>Lecturer</position>

</Val>

</Dist>

در این مثال مدرک XML فازی جورج اطلاعات فازی آکادمیکش را استفاده از عنصر ها و صفت های جدیدی را به نمایش گذاشت. با افزایش XML نمایش فازی بر روی وب کاربردهای انضمام XML فازی محققان را بررسی و آماده کردن روش های مختلف برای ادغام اطلاعات XML فازی، تشویق می کند.

یک مشکل ریشه ای که نیاز به حل شدن در انضمام اطلاعات XML فازی ناهمگن دارد، شناسایی همان و یا شباهت های عنصرها و مقادیر است.

براین اساس تناقض ها و همپوشانی های عناصر حذف می شوند. از این رو، پیدا کردن تناقض ها و همپوشانی های عناصر و مقادیرشان در داده های XML فازی برای یکی کردن چندین اسناد XML فازی به یک سند، نیاز است.

هرچند وجود شناسایی محتوای اطلاعات XML موجی به طور گسترده مورد بررسی قرار می گیرد. رویکرد های حال حاضر نمی توانند برای تسخیر ساختار و اطلاعات معنایی از داده XML فازی به کار گرفته شوند زیرا کمبود مدل ساختار XML فازی موثر است.

برای یکپارچه سازی اطلاعات XML فازی از منابع اطلاعاتی ناهمگن، ما یک رویکرد موثر و مفید برای تقویت یکپارچگی سند XML فازی آماده کردیم. اصلی ترین بخش این کار در مدیریت میدان اطلاعاتی XML فازی که در ادامه به طور خلاصه بیان شده است، می باشد.

ما اولین قدم را در راستای ساختن مدل جدید درختی XML فازی برمی داریم. این کار توصیف داده XMLفازی را آسان می سازد و تسخیر ساختار و اطلاعات معنایی را نیز. بر این اساس ما رویکرد موثری را برای تشخیص شباهت ها بین گره ها درخت XML فازی آماده کرده ایم. ما بر روی ساختار و مقایسه شباهت های معنایی اسناد ناهمگن XML فازی که از منبع های مختلف جمع آوری شده اند، تمرکز کرده ایم. ما پاسخ های اطلاعاتی در گره درخت ها شناسایی می کنیم وسپس و در مورد شباهت های گره ها از درخت های XML فازی مختلف تصمیم می گیریم ارتباط بین دو سند XML فازی می تواند از گره هایی پاسخ هایشان تشخیص داده شود در این راه ما می توانیم مستقیما تناقض ها و همپوشانی ها بین اطلاعات XML فازی که از منابع مختلف جمع آوری شده اند مشخص کنیم. نتایج تجربی نشان می دهد رویکرد ما می تواند به طور موثر اسناد XML فازی را یکپارچه بنماییم.

کل مقاله در ادامه به طور منظم آماده است. بعد از نشان دادن کارهای مربوطه انجام شده در بخش2، ما مدل درختی XML فازی را در بخش سه معرفی کرده ایم، در رویه آماده سازی مدل درختی XML فازی ما یک رویکرد برای اندازه گیری ساختار و شباهت معنایی اطلاعات XML فازی در بخش 4 طراحی کرده ایم. استراژدی انضمام اطلاعات XML فازی در بخش 5 توسعه داده شده است. ارزیابی های تجربی در بخش 6 آمده اند. در آخر و در بخش 7 نتایج مقاله آمده است.

2. کار مرتبط

2.1. XML ناقص

اطلاعات ناقص به طور گسترده در بسیاری از کارها و کاربردهای دنیای واقعی وجود دارند. خواص ناقص اطلاعات ممکن است شامل ناقصی، بی دقتی، ابهام، شک و نامفهومی باشد. در محتوای XML انواع مختلف اطلاعات ناقص بررسی شده اند. یک روش ساده برای نمایش و تحقیق در مورد XML با اطلاعات ناقص در [5] جایی که چهارچوب ساده برای دستیابی نگهداری و تحقیق اسناد XML با اطلاعات ناقص ارائه شده است.

در 5-13 برای حوزه XML احتمالی دستیابی هایی صورت گرفته است. مدل XML احتمالی برای اطلاعات احتمالی در [6] معرفی شده اند که در آن یک پیاده سازی از اپراتورهای جستوجو XML طراحی و اثر این پیاده سازی به خوبی مدیریت اطلاعات XML احتمالی نمایش داده شده است با استفاده از یک کیس استادی. در [7]، با شمارش تمام دنیاهای احتمالی در زیر درختهای مختلف یک مدل اطلاعاتی XML نامعلوم به کار گرفته شده است تا حالت های ممکن دیتابیس را به جای حالت واقعی ذخیره کند.

تعریف رسمی از ساختار xml به صورت نامشخص ارائه شده است و معنا پشت مدل اطلاعاتی بحث شده است. یک مدل برای اطلاعات نیمه ساختاری اطلاعات احتمالی آورده شده است و این دو معنادارد که احتمالا پیوسته نمایش مدل های XML احتمالی در [11-9] بررسی شده است که در آن یک چهارچوب چکیده از نمایش XML احتمالی توصیف شده است. به عنوان یک درخت برچسب شده با گره های عادی و گره های توزیعی، گره های عادی اطلاعات واقعی را مشخص می نمایند و گره های توزیعی پروسه تولید اسناد XML رندوم را تولید می کنند. در [12] اطلاعات XML نامعلوم با 3 مدل XML احتمالی نمایش داده شده است. اولین مدل، یک استقلال بین اتصال های احتمالی فرض می کند و دومین مدل وابستگی هایی احتمالی را رمزگذاری می کند سومین مدل این دو مدل قبلی را ترکیب می کند و از این رو از دو مدل قبلی قوی تر است در [13-6] یک مدل بهم چسبیده از اسناد P به اسم ProTDB ساخته شده است تا اطلاعات XML احتمالی را مدیریت کند. برای نمایش اطلاعات XML فازی چندین تلاش شده است در [21-14] برای مدل کردن اطلاعات فازی در اسناد XML یک رویکرد برای نمایش اطلاعات فازی در اسناد XML به وسیله ای به وسیله استاده در تئوری احتمال در روابط مشابه در [14] معرفی شده است. در [14] نقشه اطلاعات فازی از یک دیتابیس فازی به یک سند XML فازی نمایش داده شده است در [15] یک رویکرد برای سروکار داشتن XML فازی پیشنهاد شده است که در آن تعاریف انواع مناسب سند [DTDs] تعریف شده است و یک ترکیب مناسب برای انواع اطلاعات فازی معرفی شده است در [15] اینکه چطور اطلاعات فازی اینکه چطور اطلاعات فازی بین تعاریف انواع اسناد XML فازی در سیستم های کاربردی نمایش یک تعریف از شکل XML برای نمایش اطلاعات فازی در [16] پیشنهاد شده است.

در [17] مدل سازی فازی شی گرا (FOOM) بر اساس XML برای مدل سازی توسعه داده شده است برای مدل کردن مشخصات فنی مورد نیاز و ترکیب کردن ایده های کلیشه ای برای آسان نمودن مدل سازی از الزمات بی دقت با XML، طرح (FOOM) بر اساس امکانات کلیدی فرموله شده است: تنظیمات فازی، توصیفات فازی، قوانین فازی و پیوستگی فازی.

فازی بودن در اسناد XML، طرح ها در [10-18] مورد بررسی واقع شده است. مدل های نمایشی XML فازی توسعه داده شده اند. دو سطح از فازی بودن در یک سند XML [18] شناسایی شده که در آنها درجه عضویت همکاری با عنصرها و تنظیمات فازی مربوط به مقادیر عناصر.

در [21-20] یک بسط از XML ساخته شده است که نامعلوم XML و نامحدود در ساختار XML را به یک بسط XML فازی تبدیل می کند و سپس یک ابزار برای تعاریف توابع عضویت دلخواه توسعه داده شدند. برای کار با XML، XSD و DTD یک رجوع به [55] برای بررسی مدل کردن اطلاعات فازی در مدل اطلاعات XML وجود دارد.

2.2 تطبیق موجودیت XML

یک کار ریشه ای در اسنجام بخشی اطلاعات XML این است که تمام موجودیت هایی که به یک شیء در دنیای واقعی سند XML مرجع یافته اند را پیدا کنید. بعضی روش های تطبیق موجودیت در اسناد XML در [32-22] پیشنهاد شده است در [22] برای حل تناقص های ساختاری در طرح اسنجام XML یک مدل اطلاعاتی بنام ORA-SS به کار گرفته شده است تا معناهای مجازی را بگیرد. چهارچوب برای تطابق موجودیت [23] یک کار تطابقی با

استفاده از یا بدون استفاده از آموزش دیتا انجام شده است. بر پایه ی ماتریس بسط داده شده یک روش برای محسابه ساختار و شباهت معنایی اسناد XML در [24] پیشنهاد شده است. بر اساس یک XML مطلع فاصله ها بین درخت های برچسب دار شده پیشنهادی، ویرایش داده شده اند.

یک ساختار اندازی گیری مشابه برای اسناد XML توسعه داده شده اند. بر اساس طرح موثر جهانی XML ، یک چهارچوب برای سیستم های انسجام اطلاعات XML که تابع تطبیق را معرفی می کند که هدفش شناسایی همان گره ها از منابع مختلف است در [26] پیشنهاد شده است. در [27] بر اساس گره های صفحه ها در هر جفت زیر درخت از اسناد XML درختی، یک الگوریتم خوشه بندی شده برای شناسایی شباهت ها پیشنهاد شده است. یکی رویکرد برای تطبیق موجودیت استراتژی برای حل تناقص های اساسی در اسنجام طرح XML پیشنهاد داده شده است که در آن تناقض ها از بین می روند و متناقض ها منظم می شوند. رویکر تطبیق XML در [30] یک الگو به نام " الگو تطبیق دهنده ی XML" که عنصرهای اصلی تطبیق دهنده ی XML را توصیف می کند، معرفی می کند. به علاوه، بعضی رویکردها [232-31] سعی می کنند برای اسنجام ارزیابی معنایی و نحوی به یک محاسبه گر فاصله تبدیل شوند.

این توانایی این امکان را می دهد که یک این به شما امکان می دهد ساختار اسناد XML را دقیق تر تجزیه و تحلیل کرده و نتایج واقعی تری به دست آورد.

بیشتر رویکردهای موجود برای تطبیق موجودیت ها در یک سند XML، تکنیک هایی برای پیدا کردن فاصله بین ساختارهای درختی را به کار می برندند.

2.3 انسجام اطلاعات XML

بعضی روش های برای یکپارچگی XML در [35-33 ,26] پیشنهاد شده است بر این اساس است که از توابع اسکولم (Skolem) و Xpath استفاده می کنند تا یک پیشنهاد برای تطبیق سند های XML و حل تناقض مقادیرشان بین اسناد XML در [33] پیشنهاد شده است.

یک مدل اطلاعاتی در [34] برای آسان نمودن رزولوشن تناقض های مقادیر با استفاده از نمایش با وضوح تناقض ها در طرح انجام برای اسنجام اسناد XML که با منابع گسسته یا مجزا در یک محیطی دینامیکی با ادغام شده اند. رویکردی که در [35] که اطلاعات مشابه را از محتوای عنصرها با اطلاعات از ساختار اسناد جدا می کند وجود دارد.

تلاش هایی برای جداسازی نامفهومی از اسناد XML وجود داشته است. در [7]، یک رویکرد انسجامی در محتوای مدل نامعلوم که برای گرفتن اسنجام های احتمالی باقی مانده است معرفی شده است. بر اساس، ایده ی درخت احتمال XML یک رویکرد برای تعیین یک معنای منطقی به منظور پرس و جو بر روی اطلاعات XML احتمالی در [36] آورده شده است.

اطلاعات ناپیوسته در انسجام داده XML اطلاعاتی در [37] بازسازی شده اند. در [38]، یک رویکرد برای نمایش و یکپارچه سازی اطلاعات XML نامعلوم پیشنهاد شده با تئوری شواهدهایی (سند) که نامعلومی مقادیر ناسازگار برای عناصری است که همان دنیای واقعی را نشان می دهند.

شکل انسجام چندین منبع اطلاعاتی وب تحت نامعلومی و وابستگی در [39] تشخیص شده است. یک چهارچوب مبتدی برای تشخیص و بازسازی تناقض ها و هم پوشانی ها در اسناد XML فازی که از منابع اطلاعاتی مختلف جمع آوری شده است و در [40] توسعه داده شده است.

انعطاف XML ، موقعیت هایی برای حل کردن بعضی سختی ها در نمایش اطلاعات فازی فراهم می کند. البته که این انعطاف پذیری همچنین چالش هایی در انسجام منابع اطلاعاتی چندین XML فازی به وجود می آورد.

3. مدل درختی XML فازی

3.1 اسناد XML فازی

برای نمایش اطلاعات فازی بر روی وب، یک مدل XML فازی در [18, 41] معرفی شده است. در این مدل نمایشی فازی بودن اسناد XML در دو سطح رخ می دهند. اولی فازی بودن در عناصر، و ما از درجه عضویت با همکاری این عنصرها استفاده می کنیم. دومی فازی بودن در مقادیر نسبی از عنصرها، و ما از تنظیمات فازی برای نمایش این مقادیر استفاده می کنیم.

یک مجموعه فازی که نشان دهنده یک مقدار عنصر فازی است دو نوع تفسیر "معناشناسی منفصل" و "معناشناسی متصل" دارد.

استفاده از مدل اسناد XML فازی در [18] پیشنهاد شده، با دو قطعه از اسناد XML فازی را در شکل های FIG 1 و FIG 2 نمایش داده ایم. برای اطلاعات بیشتر به [18,41] مراجعه شود.

در اسناد XML فازی، نوع نسبی گره به عنوان یک فرزند از عنصر Dist برای تشخیص نوع تفسیری از یک دستگاه فازی به کار گرفته شده است، که آیا تعدادش منفصلی یا متصلی است. گره ی نسبی Poss به عنوان زاده ی یک عنصر Val برای تشخیص، در جه عضویت از یک عنصر که در یک سند XML فازی داده شده را تشخیص می دهد که آیا مقدارش 0 یا 1 است. به علاوه هر گره ی عنصری Dist یک گره به عنوان سازنده ی آن و یک یا بیشتر گره های عنصری Valبه عنوان گره ای که از آن زاده می شود دارد.

بگذارید نگاهی به سند XML فازی نشان داده شده در FIG 1 بیندازیم. Poss ایجاد شده ی فازی با Val فازی با یکدیگر براش شناسایی درجه عضویت یک عنصر داده شده در اسناد XML فازی، اتخاذ می شوند. این درجه عضویت با یک جفت <Val Poss > و </Val> نشان داده می شود. در خط های 7-9 در FIG 1، برای مثال درجه عضویت درجه عضویت جورج به عنوان یک استاد که 0.7 را تعیین می کند. Dist ایجاد شده ی فازی برای نمایش یک دستگاه فازی (تنظیمات فازی) به عنوان یک مقدار عنصر نمایش می دهد. یک عنصر Dist ممکن است چندین عنصر Val به عنوان فرزند داشته باشد و هر Val یک Poss وابسته (مرتبط) به عنوان فرزند دارد.

```
1.      <Teaching>
2.          <Teacher Tid = “007102”>
3.            <Dist Type = “disjunctive”>
4.             <Val Poss = 1.0>
5.               <Position>Associate Professor</Position>
6.             </Val>
7.             <Val Poss = 0.7>
8.               <Position>Lecturer</Position>
9.             </Val>
10.           </Dist>
11.           <Name>George</Name>
12.         </Teacher>
13.         <Student Sid = “20130111”>
14.           <Dist Type = “conjunctive”>
15.             <Val Poss = 0.6>
16.               <Email>Mary@hotmail.com</Email>
17.             </Val>
18.             <Val Poss = 1.0>
19.               <Email>Mary@gmail.com</Email>
20.             </Val>
21.           </Dist>
22.           <Name>Mary</Name>
23.         </Student>
24.     </Teaching>
```

شکل 1 یا FIG 1 : اولین قطعه سند فازی

خط های 3-10 در FIG 1، برای مثال ارزش موقعیت آ .کادمیک جورج در یک دستگاه فازی که در آن جورج یک دستیار پروفسور یا یک استاد درجه عضویت، 1.0 و 0.7 است را تعیین می کند. قطعا جورج نمی تواند هم یک استاد و هم یک استادیار باشد، بنابراین در خط 3 در شکل یک < Dist Type=disjunctive> برای روشن سازی تفسیر منفصل (جداکننده) از دستگاه فازی به کار رفته است. شکل 1، خط های 21-14 ارزش آدرس ایمیل "ماری" را که یک دستگاه فازی (تنظیم فازی) است، جایی که آدرس ایمیل mary@hotmail.com و mary@gmail.com با درجه عضویت های 0.6 و 0.1 . ممکن است "ماری" یک ایمیل یا دو ایمیل داشته باشد بنابراین در خط 14 در FIG 1 <Dist Type=conjunctive> برای روشن سازی تفسیر متصل از دستگاه فازی به کار گرفته شده است، در این مورد، درجه عضویت ها به عنوان درجه های مطمئن یاد می شوند.

3.2 مدل های درختی سند xml فازی

با استفاده از مدل هدف دار سندی (DOM) [42] یک سد XML می تواند به عنوان یک درخت ریشه دار برچسب زده شده سفارشی به نمایش در بیاید. گره های درخت DOM عنصرهای XML را نمایش می دهند و با نام های برچسب عنصر مربوطه برچسب گذاری می شوند.

گره ها دستوراتشان را با حضور سند دنبال می کنند. جفت ها بیشتر به عنوان فرزند گره های شامل عناصرشان پدیدار می شوند که با نام های صفتی ذخیره شده اند.

اسناد XML فازی به وضوح از اسناد XML متفاوت است به دلیل اینکه سند XML فازی شامل صفت ها و المان های جدیدی است که آنها صفت های Poss ، Type و دو المان Val و Dist است. بنابراین یک سند Xml فازی نمی تواند به یک درخت Dom مستقیما تبدیل شود.

برای به تصویر کشیدن ساختار شباهتی اسناد XML فازی ما نیاز داریم یک مدل مناسب ایجاد کنیم. اینجا ما یک مدل درختی از سند XML فازی برای اسناد XML فازی ارائه کرده ایم.

تعریف 1 : FXTM : بگذارید FXTM یک درخت دستوراتی T(N, E) باشد، N و E دستورهایی از گره ها و مرزهای سند XML فازی T هستند. در FXTM یک پنج تایی است

(NodeLabel, NodeDepth, NodeFuzzy, NodeType, NodePoss)

(برچسب گره، عمق گره، گره فازی، نوع گره، Poss گره)

```
1.      <Teaching>
2.          <Teacher Tid = “007102”>
3.            <Dist Type = “disjunctive”>
4.             <Val Poss = 1.0>
5.               <Position>Associate Professor</Position>
6.             </Val>
7.             <Val Poss = 0.6>
8.               < Position>Professor</Position>
9.             </Val>
10.           </Dist>
11.           <Name>George</Name>
12.           <Office>B208</Office>
13.         </Teacher>
14.         <Student Sid = “20130425”>
15.           <Dist Type = “conjunctive”>
16.             <Val Poss = 0.7>
17.               <Email>John@hotmail.com</Email>
18.             </Val>
19.             <Val Poss = 1.0>
20.               <Email>John@gmail.com</Email>
21.             </Val>
22.           </Dist>
23.           <Name>John</Name>
24.         </Student>
25.     </Teaching>
```

شکل 2 یا FIG 2 : دومین قطعه سند فازی

(a) برچسب گره یا node label : برچسب اسم گره / صفت است.

(b) عمق گره یا node depth : عمق تودرتوی عنصر/ ویژگی در سند xml است.

(c) گره فازی یا node fuzzy : برای اینکه یک گره فازی است یا کریسپ استفاده می شود اگر مقدار گره فازی 1 باشد گره مربوط به گره فازی است و اگر مقدار آن 0 باشد گره مربوط به گره کریسپ است. ارزش گر فازی برای یک گره ی Dist یا Val صفر است.

(d) نوع گره یا node type : بیانگر نوع تفسیری یک گره فازی است که متصلی یا منفصلی است. یک گره کریسپ ارزش پیش فرض گره Dist اش مقدار null است.

(e) Node poss : گره ی Poss درجه عضویت یا ارزش یک گره را می دهد و مقدار پیش فرض برای گره ی کریسپ null می باشد.

یک درخت FXTM مجموعه ای از موجودیت ها در دنیای واقعی است. شناسایی موجودیت های اسناد XML فازی به معنی ارزیابی شباهت درختان FXTM مربوطه است. برای کاهش محاسبات غیرضروری در ارزیابی اندازه گیری شباهت مقادیر المان ها و خصیصه ها شباهت ساختاری اسناد XML صرف نظر نمی شود. اما در این مقاله مقادیر عناصر و ویژگی ها را هم برای ادغام داده ها هم برای محاسبه تشابه گره ها نگه می داریم وقتی یک درخت FXTM از یک سند XML فازی ساخته می شود. توجه کنید در یک سند XML فازی، گره های خواهر و برادر گره های Poss و Type ممکن است گره های عناصر یا صفت باشند. بنابراین ما نیاز داریم تا مقادیر فازی Poss و Type در NodeDistType و NodePoss گره های خواهر و برادر گره های Poss و Type ذخیره کنیم. اسناد Xml فازی شامل چهار نوع عنصر یا ویژگی جدید :

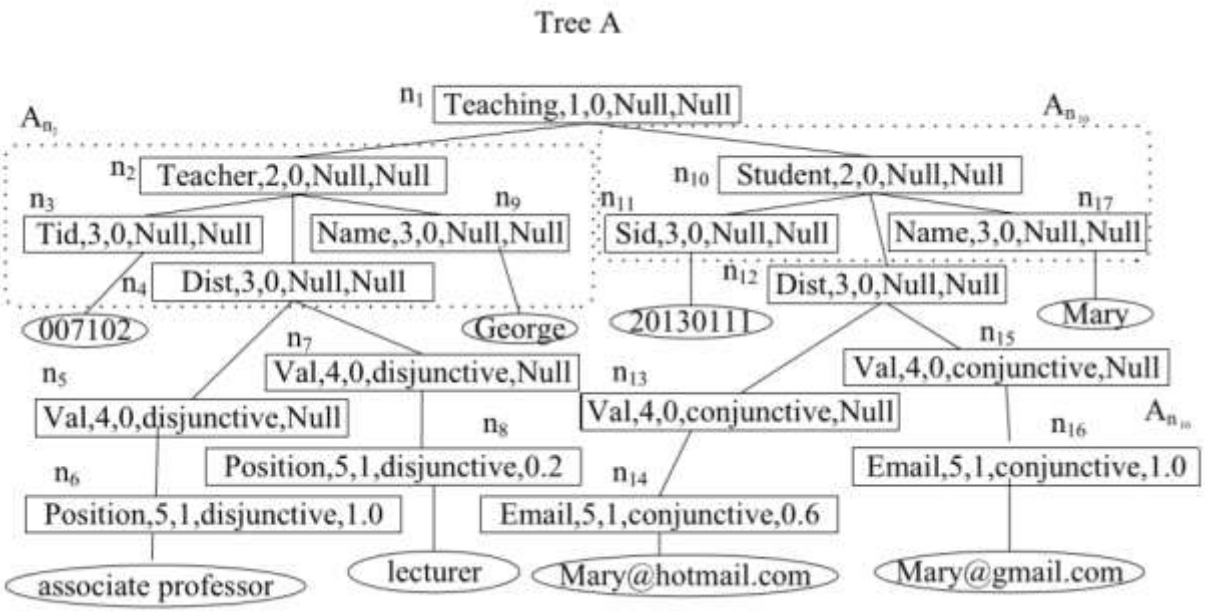
(Type, Val, Poss, Dist)

می شود که مقادیر آنها شامل اطلاعات فازی می شود.

ما نیاز داریم مقادیر فازی را به گره های مربوطه تبدیل کینم. ما با این چهار نوع گره مطابق مراحل زیر مقابله می کنیم و FXTM زمانی که از سند XML فازی گذر می کند بدست می آید.

مرحله 1 : ما مقادیر NodeFuzzy و NodeType و NodePoss از والدین آنها به ارث می بریم.

مرحله 2 : برای گره Type ما نیاز به یک کپی از مقدار Type Node و Node Type از خواهر و برادر های Type Node داریم که Type Node را از زیردرخت آن حذف می کنیم.



شکل 3 یا FIG 3 : مدل دریخت یک سند XML فازی مربوط به سند XML فازی در شکل 1 یا FIG 1

مرحله 3 : برای گره Val اگر فقط گره Poss به عنوان گره فرزند باشد ما گره Val و زیردرخت آنرا حذف می کنیم.

مرحله 4 : برای گره Poss ما یه کپی از مقدار گره Poss در NodePoss و خواهر و برادرهای گره NodePoss می دهیم و گره Poss و زیردرخت های آنرا حذف می کنیم.

مرحله 5 : برای گره Dist اگر پس از پردازیش فوق به گره برگ تبدیل می شود و ما گره Dist و زیردرخت های آنرا حذف می کنیم.

دیده می شود که هر گره اطلاعات مربوط به فازی مربوط به خود را در FXTM حمل می کند. FXTM یک درخت مرتب شده با ریشه است که در آن گره ها عناصر و ویژگی های سند XML فازی را نشان می دهند و گره های برگ مقادیر صفت هستند. گره های FXTM به ترتیب ظاهر شدن در سند XML فازی مربوطه مرتب می شوند. موقعیت یک گره در FXTM مطابق با نظم عمومی است که براساس همه عناصر و ویژگی های موجود در سند XML فازی تعریف شده است. در FXTM ، هر گره به جز گره ریشه یک والد منحصر به فرد دارد و لبه های بین گره ها رابطه بین این گره ها را نشان می دهند. فرزندان این گره شاملفرزندان آن گره و نوادگان آن گره می شود. در ادامه ما از گره برای نشان دادن المان های هر گره و ویژگی گره ها استفاده می کنیم.

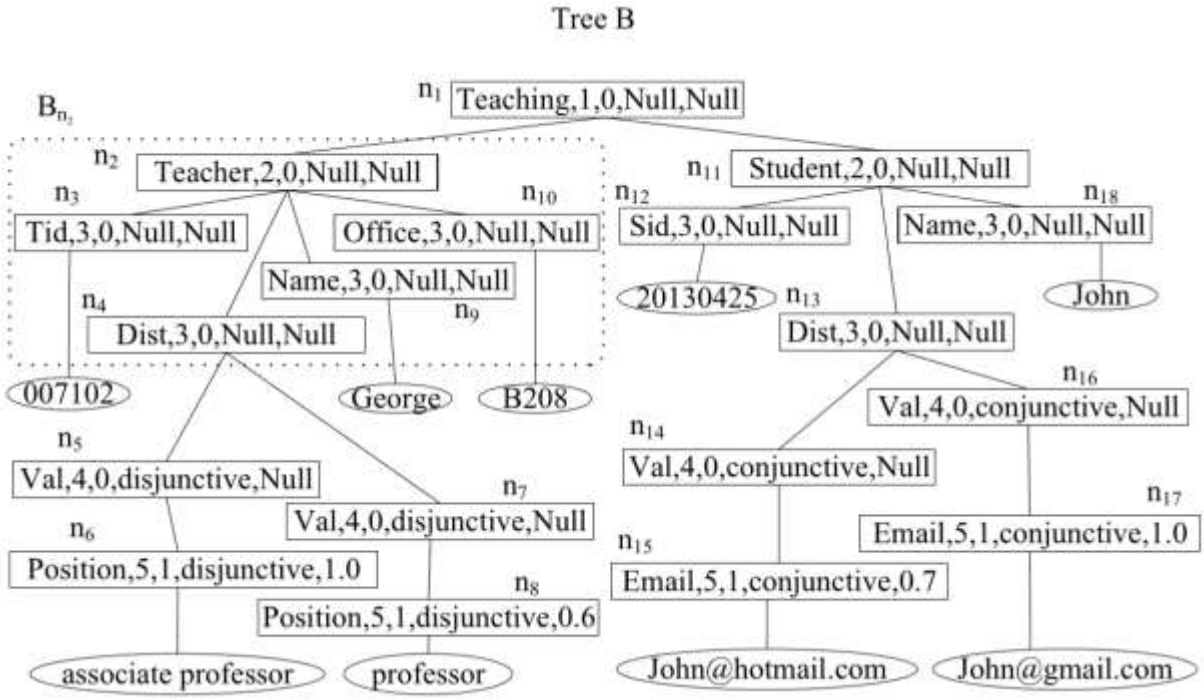
برای جمع بندی ما تمام ویژگی گره های Type و Poss را در درختان XML فازی حذف می کنیم. توجه داشته باشید المان های گره Dist و Val را نمی توان نادیده گرفت در غیراینصورت ساختار و عمق درخت تغییر نخواهد کرد و این بر نتایج مقایسه شباهت درخت تاثیر می گذارد. دو سند XML فاز در شکل 1 و 2 تبدیل به به دو درخت FXTM در شکل های 3 و 4 شده است. برای سادگی فقط مقادیر داده به عنوان گره های برگ در شکل 3 و 4 تعریف شده است. علاوه بر این به ترتیب دولایه زیر درخت A_{n2} و B_{n2} با قاب "-" در شکل های 3 و 4 مشخص شده است.

4. اقدامات مشابه

اندازه گیری شباهت اسناد XML فازی برای شناسایی درگیری ها و همپوشانی های احتمالی روی اسناد XML فازی استفاده می شود. برای ادغام اسناد XML فازی ما نیاز داریم که گره های مشابه را در در درختان اسناد XML فازی پیدا کنیم. سپس ما از یک الگوریتم فاصله، فاصله ی ویرایش شده ی درخت برای محاسبه شباهت زیر درخت های تطبیق شده استفاده می کنیم. برای این منظور ما نیاز داریم شباهت گره را قبل از مقایسه تشابه ساختار زیرسطحی مشخص کنیم.

4-1 اقدامات شباهت معنایی گره :

گره های درختان اسناد XML فازی موارد اساسی برای اقدامات مشابه هستند. بنابراین محاسبات شباهت گره یکی از مراحل اصلی در اقدامات تشابه است. برای دو گروه N_1 و N_2 که از درختان سندهای XML فازی استفاده می کنند، از $SimNode(N_1, N_2)$ استفاده می کنیم که برای نشان دادن درجه شباهت آنها مقدار [0, 1] را می گیرد. هر چه درجه شباهت بیشتر باشد دو گره شباهت بیشتری دارند. برای ارزیابی شباهت بین یک جفت گره از اسناد XML فازی ناهمگن، از ویژگی گره ها و همچنین روابط آنها باید به طور کامل در اندازه گیری شباهت استفاده شود.



شکل 4 یا 4 FIG : یک مدل درخت سند XML فازی مربوط به سند XML فازی شکل 2 یا 2 FIG

همانطور که در بخش 3-2 ذکر شد ویژگی ای یک گره با یک تاپل 5 تایی توضیح داده شده است. در این میان ویژگی ها، برچسب گره ها، نوع گره ها و NodePoss رایج ترین ویژگی ها برای ارزیابی شباهت عناصر و ویژگی های XML فازی هستند. در نتیجه ما برخی از اقدامات مشابه را با توجه به ویژگی های مختلف ارائه می دهیم.

4-1-1 اندازه گیری نام برچسب

نام برچسب گره ها به عنوان منبع مهم اطلاعاتی برای تطابق گره در نظر گرفته می شوند. نام گره ها می توانند به صورت نحوی (مانند : Stu و Student) یا از نظر معنایی مشابه باشند. (مثل : Worker و Employee)

در این مقاله، ما هم شباهت نحوی و هم شباعت معنایی را برای محاسبه شباهت بین نام های برچسب گره ها در نظر می گیریم.

اندازه گیری شباهت نحوی برای ارزیابی شباهت آنها ، نام های برچسب دو گره را می گیرد. برخی روش های متداول برای تطبیق رشته ها وجود دارد. در این مقاله با اشاره به فاصله ویرایش لونشتاین، ما اندازه گیری شباهت واگنر-فیشر را می پذیریم. اندازه گیری شباعت واگنر-فیشر بین دو رشته با حداقل تعداد عملیاتی که برای تبدیل یک رشته به رشته دیگر لازم است ارائه شده است. این عملیات شامل درج، حذف و تعویض یک کاراکتر است. به طور رسمی اجازه دهید L_1 و L_2 دو نام برچسب گره باشند شباهت نحوی بین L_1 و L_2 را می توان به شرح زیر محاسبه کرد.

$$Sim_{labelSyn}(L_1, L_2) = \frac{1}{1 + editDistance(L1, L2)}$$

اندازه گیری شباهت معنایی یکی دیگر از جنبه های مهم در ارزیابی شباهت نام برچسب گره ها است. ارزیابی شباهت معنایی کلمات و عبارات ممکن است به مقایسه مفاهیم اساسی در فضای معنایی منجر شود. در واقع ، روشهای مختلفی برای تعیین شباهت معنایی بین مفاهیم در متن شبکه معنایی ارائه شده است [43،47،48]. در این مقاله ، روش اندازه گیری تشابه معنایی Lin's را در [43] اتخاذ می کنیم:

$$Sim_{labelSem}(L_1, L_2, S_N) = \frac{2Log\ p(l_0)}{\log p(L_1) + \log p(L_2)}$$

که در آن L_0 خاص ترین جد مشترک بین L_1 و L_2 است. $P(L_0)$ احتمال وقوع مفهوم L_0 را نشان می دهد و SN یک شبکه معنایی دارای وزن بر اساس واژگان مشخص است. (به عنوان مثال یک شبکه معنایی با فرکانس های مفهومی افزوده شده)

سرانجام ، شباهت نام برچسب بین دو گره XML فازی به عنوان میانگین وزنی شباهت های نحوی آنها ($Sim_{labelSyn}$) و شباهت های معنایی ($Sim_{labelSem}$) ارزیابی می شود:

$$Sim_{labelSem}(L_1, L_2)Sim_{label}(L_1, L_2) = Sim_{labelSyn}(L_1, L_2) + (1 - \gamma)$$

در اینجا γ به عنوان یک وزن ورودی ارائه می شود. با تغییر ضریب γ ، کاربر می تواند اهمیتی را به شباهت های نحوی یا معنایی اختصاص دهد.

4-1-2 اندازه گیری اطلاعات فازی

در تعیین شباهت گره ها تنها نام های برچسب گره اعمال نمی شود. استفاده از منابع اطلاعاتی فازی در محاسبه شباهت گره و هرس برخی از مسابقات مثبت کاذب مفید است. در اینجا مقادیر NodeType و NodePoss منابع اطلاعاتی فازی هستند که می توان از آنها برای تعیین شباهت گره استفاده کرد.

توجه داشته باشید که در یک درخت سند XML فازی ، مقدار NodeType یک گره جداسازنده یا ربط دهنده است. دو گره با مقدار NodeType یکسان با درجه بالایی مشابه هستند. برای گره هایی که مقدار NodeType یکسان دارند ، شباهت NodeType به 1 تنظیم می شود ، در حالی که شباهت NodeType گره ها با مقادیر مختلف NodeType بر روی 0 قرار دارد. شباهت NodeType به صورت Sim_{Type} نمایش داده می شود.

$$Sim_{Type}(T_1, T_2) = \begin{cases} 1 & if\ T_1 = T_2 \\ 0 & if\ T_1 \neq T_2 \end{cases}$$

بنابراین ، علاوه بر این ، در یک درخت سند XML فازی ، مقدار NodePoss یک گره از 0 تا 1 به صورت متغیر است. برای دو گره از درختان سند XML فازی مختلف ، شباهت NodePoss آنها با کاهش تفاوت بین مقادیر NodePoss آنها افزایش می یابد. شباهت NodePoss به صورت $Sim_{Poss}(P_1, P_2)$ نمایش داده می شود.

$$Sim_{Poss}(P_1, P_2) = 1 - |P_1 - P_2|$$

سپس یک اندازه گیری مشابه را می توان از طریق ارزیابی ویژگی های گره بدست آورد. اصولاً باید انواع ویژگی های گره به همراه انواع اقدامات برای ارزیابی شباهت در نظر گرفته شود. شباهت کلی گره با ترکیبی از اقدامات مشابه از ویژگی های گره های مختلف بدست می آید این موضوع منجر به یک چالش به نام "نحوه ترکیب این اقدمات" می شود. بدون معنی مناسب برای ترکیب، اقدامات نهایی ممکن است نتواند بازتاب مناسب با رابطه ی مشابه یافت کند. برای دو گره N_1 و N_2 ، شباهت گره آنها که $Sim_{Node}(N_1, N_2)$ نامیده می شود ، به عنوان میانگین وزنی شباهت های ویژگی آنها ارزیابی می شود. سپس ما :

$$Sim_{Node}(N_1, N_2) = \omega_L \times Sim_{Label(L_1,L_2)} + \omega_T \times Sim_{Type}(T_1, T_2) + \omega_P \times sim\_Poss(P_1, P_2)$$

were wL, wT , and wP are provided as weight coefficients, and $\omega_L + \omega_T + \omega_P = 1$.

به طور کلی ، ضرایب وزن باید به گونه ای انتخاب شوند که شباهت کلی گره را به حداکثر برساند. ضرایب وزن معمولاً توسط یک متخصص انسانی تهیه می شود ، که براساس سناریوی مورد نظر ممکن است بر شباهت برچسب یا شباهت اطلاعات فازی گره ها تأکید کند. علاوه بر این ، ضرایب وزن با استفاده از برخی از تکنیک های شناخته شده مانند یادگیری ماشین می تواند تعیین شود تا بهترین وزن را برای یک مشکل خاص مشخص کند [50]. ما بیشتر به این موضوع نمی پردازیم زیرا این موضوع خارج از محدوده این مقاله است. در این مقاله ، ما فقط از تعداد کمی از ویژگی های گره استفاده می کنیم. برای به دست آوردن شباهت ها بیشتر ویژگی، رویکرد پیشنهادی باید گسترش یابد تا ویژگی ها گره بیشتری را در برگیرد. اما این بدان معنی نیست که بیشتر ویژگی های گره که در اندازه گیری شباهت شرکت می کنند باید به شباهت دقیق گره منجر شوند.

4-2 اقدامات تشابه ساختاری زیرسطحی :

برای ادغام مؤثر اسناد XML فازی ، ابتدا لازم است که مرزبندی تجمعی را مشخص کنیم. اگر پردازش تجمعی در سطح گره ها انجام شود، مرزبندی تجمعی خیلی کم است، و به پایان رساندن ادغام بسیار دشوار است. یک زیرشاخه دولایه با یک شیء حداقلی کامل مطابقت دارد. ساختار والدین و فرزندان آن با یک عنصر فرعی، عنصر/ویژگی، ویژگی/مقدار، عنصر/مقدار مطابقت دارد. علاوه بر این ، یک زیرشاخه دو لایه حاوی اطلاعات ساختاری و معنایی آن مانند رابطه گره های والدین و فرزندان است.

در این مقاله ، اسناد XML فازی را مبتنی بر زیر درختان دو لایه ، که واحدهای ادغام هستند ، ادغام می کنیم. ما یک اندازه گیری شباهت را بر اساس زیر شاخه های دو لایه پیشنهاد می کنیم. بر اساس فاصله ویرایش درخت ، در این بخش رویکرد اندازه گیری تشابه ساختاری فرعی را برای حل مشکل شناسایی موجودیت معرفی می کنیم. برای این منظور چندین تعریف در مورد عملیات ویرایش درخت را به شرح زیر ارائه می دهیم.

تعریف شماره 2 (درخت مرتب) : درخت مرتب شده ، درختی ریشه دار است ، که در آن هر گره مرتب می شود. برای یک درخت T ، از T [i] برای نشان دادن گره ith T در گذر مقدم استفاده می کنیم. T[0] گره ریشه T را مشخص می کند. فرض کنید که گره T[0] دارای k فرزند و فرزندان از چپ به راست منحصر به فرد است.

T[1], T[2], . . . , T[k]

سپس

T[i].Label, T[i].Depth, T[i].Fuzzy, T [i].Type, and T[i].Poss

اطلاعات مربوط به گره ith T را بیان می کند.

تعریف 3 (زیر لایه ی دولایه) : با توجه به یک درخت T با گره بدون لایه زیردرختی که در p قرار دارد، زیر درختی دولایه T_p از T است اگر تمام گره های فرزند p در T_p گنجانده شود. توجه داشته باشید که T_p شامل گره های نوادگان p نیست. گره های T_p همان رابطه لبه والدین/فرزند و ترتیب گره را دارند همانطور که در T انجام می دهند.

به عنوان مثال، شکل 3، زیر درختان A_{n2} که در گره N_2 و A_{n16} دارند توسط یک باکس خط چین مشخص شده اند، زیر شاخه های دولایه هستند. در اینجا A_{n2} فقط گره های فرزند N_2 (یعنی گره های n_4 و n_3 و n_9) را شامل می شود به استثنای گره های بعد از n_2 (به عنوان مثال، گره های n_5 و n_7 که گره های فرزند n_4 هستند). زیر درخت A_{n10} که در گره n_{10} ریشه دارد توسط یک باکس خط چین مشخص شده است، یک زیر لایه دو لایه نیست زیرا گره n_{12} یک گره فرزند از n_{10} است اما در A_{n10} گنجانده نشده است.

تعریف 4 (زیر شاخه گره برگ) : با توجه به یک درخت T با گره بدون لایه ، فرض کنید همه گره های کودک از l گره های برگ هستند. زیر درخت گره Tl یک زیر درخت است که ریشه آن در l است ، که شامل تمام گره های فرزند l است. زیر درخت گره برگ یک مورد خاص از یک زیرشاخه دو لایه است. به عنوان مثال ، در شکل 3، زیر درخت A_{n16}، که توسط یک باکس خط چین شده مشخص شده است، یک زیر درخت گره برگ است. زیر درخت A_{n2} که توسط یک باکس خط شده مشخص می شود یک زیر گره برگ نیست زیرا گره های n_4 و n_3 و n_9 که گره های فرزند گره n_2 هستند گره های برگ نیستند.

تعریف 5 (گره درج) : با توجه به گره x ، insNode(x) عملی برای درج گره است. در گره ith روی درخت T اعمال می شود و درختی با گره های

T[1], . . . , T[i], x, T[i+1], . . . , T[m]

تولید می کند.

تعریف 6 (حذف گره) : T[i] یک گره برگ در درخت T می باشد. عمل DelNode(T[i]) عملی برای حذف گره است که روی درخت T و گره ith اعمال می شود و درختی با گره های فرزند

T[1], . . . , T[i−1] , T[i +1], . . . , T[m]

را بوجود می آورد.

تعریف 7 (ویرایش اسکریپت) : یک ویرایش اسکریپت دنباله ای از عملیات ویرایش

ES = op1, op2, . . . , opk , where opi $(1 \leq i \leq k)$

است. با استفاده از عملیات ویرایش در ES بر روی یک درخت T با توجه به ترتیب ظاهر آن در ES یک درخت T جدید بدست می آوریم. هزینه ES، با عنوان CostES، به عنوان مجموعه ای از هزینه های کلیه عملیات در ES تعریف شده است.

تعریف 8 (فاصله ویرایش درخت) : فاصله ویرایش بین دو درخت A و B با عنوان TED(A, B) به عنوان حداقل هزینه همه ی اسکریپت های ویرایش شده است که می توانند A را به B تبدیل کنند. یعنی

TED(A, B) = Min{CostES}

برای دنباله ای از اسکریپت های ویرایش شده ی مختلف {CostES} مجموعه ای از هزینه های ویرایش همه اسکریپت های ویرایش را که A را به B تبدیل می کند، مشخص می کند. فاصله ویرایش درخت که به طور گسترده برای تشخیص شباهت ساختاری اسناد XML مورد استفاده قرار می گیرد، یک پسوند طبیعی فاصله ویرایش رشته است. در اینجا فاصله بین دو درخت هزینه عملیات ویرایش است که یک درخت را به درخت دیگر تبدیل می کند. به طور کلی ، هر یک از عملیات ویرایش به عنوان هزینه عملکرد آن با یک مقدار غیر منفی همراه است. فاصله ویرایش در تعریف 8 با هزینه های کلی کار می کند. در این مقاله ، ما به جای هزینه واقعی از واحد هزینه استفاده می کنیم. هزینه واحد که مربوط به یک عمل گره است (یعنی درج یا حذف یک گره) برابر است با 1. برای زیرشاخه های دو لایه A و B ، مسئله ارزیابی شباهت ساختاری آنها می تواند به عنوان مشکل محاسبه آنها تعریف شود.

فاصله مربوط به ویرایش درخت برای گرفتن شباهت های ساختاری زیر لایه های دو لایه ، مفهوم گره های تطبیق بین زیرشاخه های دو لایه را معرفی می کند.

تعریف 9 (گره های همسان) : با توجه به زیر درختان دو لایه

$A = (a1, \ldots, am)$

$B = (b1, \ldots, bn)$

مانند یک آستانه به نام θ است. عمل تطابق بین دو گره A و B با یک جفت مجموعه تعریف می شود.

$N = \{(ai , bj)\}$, where $\{(ai , bj)\} \in A \times B$

$SimNode(ai , bj) > \theta$

به طور شهودی ، استفاده از فاصله ویرایش در درختان کامل می تواند انحراف بزرگتر از اندازه شباهت را نسبت به استفاده از فاصله ویرایش فقط برای زیر درختان داشته باشد. برای افزایش دقت محاسباتی اندازه گیری شباهت ، زیر درختان دو لایه را به عنوان محاسبه مرزبندی شباهت انتخاب می کنیم.

تعریف 10 (فاصله ویرایش زیر لایه های دولایه, TSED) : با توجه به زیرشاخه های

$A = (a1, \ldots, am)$

$B = (b1, \ldots, bn)$

TSED بین A و B به عنوان حداقل هزینه کلیه اسکریپت های ویرایش شده که A را به B تبدیل می کنند ، تعریف شده است.

از تعاریف 9 و 10 می توان دریافت که برای دو زیرشاخه A و B ، محاسبه فاصله ویرایش آنها فقط به معنی یافتن حداقل تعداد عملیات ویرایش است که می تواند A را به B تبدیل کرده و سپس برای مشخص کردن حداکثر تعداد گره های مطابق در A و B استفاده می شود. به دنبال الگوریتم ویرایش فاصله Nierman و Jagadish و همچنین الگوریتم بهبود فاصله ویرایش Tekli و Chbeir، الگوریتم TSED ما را در الگوریتم 1 معرفی می کنیم. الگوریتم ما اندازه گیری شباهت دقیق تری از زیر شاخه های دو لایه را بر اساس فاصله ویرایش درخت ارائه می دهد.

Algorithm 1 TSED.

```
Input:  A, B                                  //Two-layer subtrees to be compared
Output: TSED(A, B)                            //Edit distance between A and B
Begin
1    M = Degree(A)                            //The degree of two-layer subtrees of A
2    N = Degree(B)                            //The degree of two-layer subtrees of B
3    Dist[][] = new[0...M][0...N]
4    If (Sim_Node(A[0], B[0]) > θ)            //Node matching
5    {
6      Dist[0][0] = 0;                        //Structurally matching nodes are associated with a cost of 0
7      MatchNodeNumber add 1                  //Counting matching nodes
8    }
9    Else
10   {
11     Dist[0][0] = 1 //Unit costs
12   }
13   For (i = 1 to M) {Dist[i][0] = Dist[i-1][0] + Cost_DelNode(A[i])}   //Cost_DelNode(A[i]) is a cost of 0 or 1
                                               //Total cost of deleting all nodes in the source document tree
14   For (j = 1 to N) {Dist[0][j] = Dist[0][j-1] + Cost_InsNode(B[j])}   //Cost_InsNode(B[j]) is a cost of 0 or 1
                                               //Total cost of inserting all nodes in the destination document tree
15   For (i = 1 to M)
16   {
17     For (j = 1 to N)
18       {                                     //Identifies the set of insertion/deletion operations having the minimum overall cost
19       Dist[i][j] = Min{
20                    {IF (Sim_Node(A[i], B[j]) > θ) {Dist[i-1][j-1], MatchNodeNumber add 1}
21                      else {Dist[i - 1][j - 1] + 1}},
22                    Dist[i-1][j] + Cost_DelNode(A[i])     //Cost_DelNode(A[i]) is a cost of 0 or 1
23                    Dist[i][j-1] + Cost_InsNode(B[j])     //Cost_InsNode(B[j]) is a cost of 0 or 1
24                  }
25       }
26   }
27 Return Dist[M][N]
```

ما از الگوریتم 1 برای شناسایی شباهت ساختاری زیر لایه های دو لایه استفاده می کنیم. الگوریتم با شمارش تعداد کل گره های تطبیق بین A و B شروع می شود (خطوط 4-7 در الگوریتم 1). این الگوریتم مجموع هزینه های حذف هر گره در درخت سند منبع (خط 13 در الگوریتم 1) و درج هر گره از درخت مقصد (خط 14 در الگوریتم 1) را محاسبه می کند. سرانجام ، مجموعه عملیات درج / حذف که حداقل هزینه کلی را دارند مشخص می شود (خطوط 15 تا 26 در الگوریتم 1).

به طور خلاصه ، برای زیردرخت منبع A و زیردرخت B مقصد ، الگوریتم پیشنهادی به صورت بازگشتی از طریق آنها می گذرد و ترکیبی از عملیات درج و حذف را برای شناسایی کارهایی با حداقل هزینه انجام می دهد. سپس می توان فاصله ویرایش را که A را به B تبدیل می کند ، تعیین کرد. در همین زمان تعداد گره های قابل تطبیق بین گره A و B را که به آن MatchNodeNumber می گوییم توسط الگوریتم پیشنهادی مشخص می شود. متفاوت بودن با الگوریتم TSED، که برای شناسایی حداقل هزینه ویرایش، از زیرشاخه های دو لایه عبور می کند، الگوریتم های Nierman و Jagadish و Tekli و Chbeir فاصله ویرایش کل درختان را ضبط می کنند. واضح است، مرزبندی محاسبه شباهت ها بین الگوریتم ما و الگوریتم های Nierman و Jagadish و Tekli و Chbeir متفاوت است. علاوه بر این ، تعداد گره های مشابه با محاسبه فاصله ویرایش زیر شاخه های دو لایه با الگوریتم ما محاسبه می شود.

اکنون ضریب تنظیم α از فاصله ویرایش درخت را تعیین می کنیم و میزان تأثیر گره های تطبیق بر فاصله ویرایش را اندازه می گیریم. ما α را با توجه به تعداد گره های تطبیق محاسبه می کنیم. برای به دست آوردن مقدار α ، مقادیر جفت گره تطبیق مقادیر MatchNodeNumber را در فاصله زمانی [0, 1] از طریق کاردینال های زیرسطحی مربوطه ، حداکثر (|A|, |B|)Max عادی می کنیم. در اینجا | A | و | B | به ترتیب تعداد گره ها در A و B را نشان می دهد.

$$\alpha = \frac{1}{1 + \frac{MatchNodeNumber}{Max(|A|.|B|)}}$$

$\alpha = 1$: حداکثر مقدار α زمانی حاصل می شود که هیچ گره تطبیقی در زیردرخت منبع و مقصد مشخص نشده باشد.

$\alpha = 1/2$: کمترین مقدار α هنگامی حاصل می شود که زیردرخت منبع با زیردرخت مقصد یکسان باشد.

ما حداقل مقدار α را برای اطمینان از اینکه عملیات گره همیشه کمتر از کل عملیات فرعی هزینه می کنند تعریف می کنیم. یعنی درج یا حذف یک گره واحد هزینه کمتری نسبت به یک زیر درخت دارد. برای زیرشاخه با دو گره ، حداکثر هزینه ویرایش آن 2 و حداقل هزینه ویرایش 1 است که معادل هزینه ویرایش یک گره منفرد و نیمی از حداکثر هزینه ویرایش است. فاصله ویرایش بستگی به حداقل هزینه ویرایش دارد. الگوریتم TSED برای شناسایی حداقل هزینه ویرایش ، از زیر لایه های دو لایه عبور می کند. حداکثر مقدار $\alpha = 1$ است و حداقل نصف حداکثر آن یعنی مقدار $\alpha = 1/2$ است.

ما از $Sim_{FXST}(A, B)$ برای نشان دادن درجه شباهت زیر لایه های دو لایه A و B استفاده می کنیم. در اینجا A و B از درختان سند XML فازی مختلف سرچشمه می گیرند و FXST به معنی زیر درخت XML فازی است. ما از TSED و ضریب α برای تعریف $Sim_{FXST}(A, B)$ به شرح زیر استفاده می کنیم.

$$Sim_{FXST}(A, B) = \frac{|A| + |B| - TESD(A, B) \times \alpha}{|A| + |B|}$$

این اندازه گیری شباهت با مفهوم تشابه در [32] سازگار است و خصوصیات متریک زیر را دنبال می کند:

Sim_{FXST} (A, B) ∈ [0, 1].

اگر A و B یکسان باشند:

Sim_{FXST} (A, B) = 1

اگر A و B خصوصیات مشترکی ندارند:

Sim_{FXST} (A, B) = 0

Sim_{FXST} (A, B) = Sim_{FXST} (B, A).

ضریب α در $Sim_{FXST}(A, B)$ که با استفاده از TSED بدست می آید می تواند اثربخشی فاصله ویرایش درخت را تقویت کند. در اینجا α با توجه به تعداد گره های تطبیق محاسبه می شود. هرچه تعداد گره های مطابق کوچکتر باشد، مقادیر فاصله ویرایش و α هم بزرگتر هستند.

مثال 1 : بگذارید زیرهای دو لایه A_{n2} را در شکل 3 و B_{n2} در شکل 4 بررسی کنیم. برای محاسبه درجه تشابه A_{n2} و B_{n2} با اجرای الگوریتم TSED شروع می کنیم و فاصله ویرایش و تعداد تطبیق گره های A_{n2} و B_{n2} را محاسبه می کنیم. داریم :

TSED(An2 , Bn2) = 1

– Dist[0][0] = 0, having R(A_{n2}) and R(B_{n2}) matching.

هزینه درج گره n_{10} به زیر درخت A_{n2} :

– TSED(A_{n2}, B_{n2}) = 1,

– MatchNodeNumber = 4, number of matching nodes of A_{n2} and B_{n2} .

$$Sim_{FXST}\ (A_{n2}, B_{n2}) = \frac{4+5-1\times\frac{1}{1+\frac{4}{5}}}{4+5} = 0.9383$$

4-3 پیچیدگی های کلی

در این بخش تجزیه و تحلیل پیچیدگی برای رویکرد اندازه گیری شباهت ارائه شده در بخش 4.2 ، از جمله پیچیدگی زمانی و پیچیدگی فضا ارائه شده است. بگذارید A و B دو زیر لایه دو لایه با هم مقایسه شوند. ابتدا بگذارید پیچیدگی زمان را بررسی کنیم.

برای شناسایی هر دو شباهت ساختاری و معنایی بین A و B ، پیچیدگی کلی زمان روش اندازه گیری شباهت ما می تواند به O ساده شود:

O(|A| × |B| × |SN| × Depth(SN))

در اینجا | A | و | B | به ترتیب، نشانگرهای A و B را بیان کنید و | SN | دلالت شبکه شبکه معنایی است که برای ارزیابی تشابه معنایی مورد سوء استفاده قرار می گیرد و عمق (SN) بیانگر عمق حداکثر شبکه معنایی است. بنابراین رویکرد ما شبیه به رویکردهای مقایسه فاصله ویرایش درخت مبتنی بر XML (به عنوان مثال ، [31]) است.

حالا بگذارید پیچیدگی فضا را بررسی کنیم. رویکرد ما نیاز به استفاده از فضای حافظه برای ذخیره زیر شاخه های دو لایه در حال مقایسه و همچنین ماتریس مسافت و بردارهای معنایی در حال محاسبه دارد. پیچیدگی فضایی روش اندازه گیری شباهت ما را می توان به O (| A | | B | | |) ساده کرد ، که شبیه به رویکردهای مقایسه فاصله ویرایش درخت مبتنی بر XML است (به عنوان مثال ، [31]).

5. ادغام داده های XML فازی

در این بخش یک چارچوب ادغام داده ها برای تطبیق دادن اسناد XML فازی در منابع داده ناهمگن پیشنهاد می کنیم. اسناد XML فازی با FXTM در بخش 3.2 ارائه شده است. ایده اصلی چارچوب ادغام ما این است که ادغام درختان FXTM را با یکپارچه سازی زیر شاخه

های دو لایه پس از حل تعارضات معنایی و ساختاری کامل می کنیم. برای این منظور ما برخی از استراتژی ها و قوانین را برای ادغام انواع مختلف زیر شاخه های دو لایه تعریف می کنیم. ما از تصویر 3 و 4 برای نشان دادن قوانینی ادغام زیر درخت دولایه استفاده می کنیم. ما انواع درگیری های ساختاری و معنایی و رویکردهای خود را برای حل مناقشات در بخش 5.1 ارائه می دهیم. بخش 5.2 الگوریتم تحقق پردازش ادغام ما را تفصیل می دهد.

5-1 حل تعارض معنایی و ساختاری

یک مشکل اساسی در ادغام سند XML فازی این است کخ پس از شناسایی همان اشیاء در دنیای واقعی از منابع داده ناهمگن برای رفع ناسازگاری های احتمالی و بازنمودهای مبهم از این اشیاء است که در منابع داده های ناهمگن مختلف وجود دارد. به طور کلی ، اجزای فرعی مشابه همان شیء دنیای واقعی را توصیف می کنند. در نتیجه آنها پس از برطرف شدن درگیری های احتمالی می توانند در یک زیردرخت واحد ادغام شوند. با توجه به روابط معنایی و ساختاری بین اجزای فرعی ، ما سه نوع تعارض را به شرح زیر شناسایی می کنیم:

1. تضادهای نامگذاری عنصر/ویژگی:

دو نوع درگیری نامگذاری را می توان شناسایی کرد: مترادف و مترادف. مترادف از نظر معنایی به موارد داده مربوط است که به گونه ای متفاوت نامگذاری شده اند. همنام از نظر معنایی با موارد داده ای که معادل آنها نامگذاری شده است ، بی ارتباط است. عناصر/صفات مترادف دارای نام های مختلف هستند اما تعاریف یکسانی دارند. عناصر/صفات متشابه دارای یک نام هستند اما تعاریف متفاوتی دارند. درگیری نامگذاری متشابه ممکن است یک اختلاف مقیاس یا داده در بین عناصر ترمینال باشد. درگیری نامگذاری مترادف می تواند با استفاده از یک گروه جانشینی از طریق اندازه گیری شباهت معنایی یا از طریق تعامل انسان برطرف شود. تعارض نامگذاری همنام برای عناصر با استفاده از تنظیم مقیاس یا اتحاد داده های جداگانه غلبه می کند.

ما نمونه ای از حقوق یک کارمند را مشاهده می کنیم. فرض کنید که دستمزد عنصر به عنوان داده عدد صحیح در سند A و داده های اعشاری در سند B تعریف شده باشد. این درگیری از طریق تنظیم مقیاس قابل حل است. ما می توانیم "عدد اعشاری" را به عنوان داده ی حقوق در سند یکپارچه اختصاص دهیم. باز هم ، اجازه دهید نمونه ای از شناسه کارمند را بررسی کنیم. ممکن است که id به عنوان یک رشته در A تعریف شود و در B به عنوان یک عدد صحیح تعریف شود. این تضاد را می توان با تعیین نوع داده رشته جدید که یک اتحاد از نوع داده در سند یکپارچه است حل کرد.

2. تضاد در ارزش ها را مشخص کنید

برای دو زیر درخت که به عنوان یک شیء در دنیای واقعی شناخته می شوند ممکن است که خواص آنها دارای مقادیر متضاد باشند. روش های مختلفی برای رسیدگی به این نوع تضاد ارزش ویژگی ارائه شده است [51،52]. در اینجا مقادیر مشخصه با گره های برگ درختان سند XML فازی مطابقت دارد. در این مقاله ، درگیری ارزش ویژگی را بدون درگیری های ارزش فازی در نظر می گیریم (درگیری های ارزش فازی در مورد شماره 3 مورد بحث قرار می گیرند). ایده اصلی این است که تمام مقادیر صفت متناقض را حفظ کنید. V_A و V_B یک جفت ویژگی در $A(A_{tt},X_A)$ و $B(A_{tt}, X_B)$ است که A_{tt} یک صفت است و X_A و X_B مقادیر A_{tt} هستند. X_A و X_B تضادی در مقدار ویژگی های X_A و X_B هستند. ما X_A و X_B را یکپارچه می کنیم تا یک جفت ویژگی را نشان دهیم :

(A_{tt}, X), where $X = \{X_A, X_B\}$

با توجه به شکل شماره 3 موقعیت "جورج" دارای ارزش {professor, lecturer} یا (استاد، مدرس) است و در شکل 4 موقعیت "جورج" {associate professor, professor} یا (استادیار، استاد) است. پس بعد از عملیات ادغام ویژگی دارای ارزش های {professor, associate professor, lecturer} یا (استاد، استادیار، مدرس) می شود.

3. تضادهای ارزش فازی:

علاوه بر تضادهای که در موارد شماره 1 و 2 گفته شد نوعی تضاد وجود دارد که به مقادیر فازی متصل می شود. تضاد های ارزش فازی که در سطح گره ها اتفاق می افتد، شامل درجه عضویت ها و مجموعه های فازی متناقض است. در اینجا فرض بر این است که هیچ تعارض برای نامگذاری و صفت وجود ندارد زیرا آنها می توانند از قبل برطرف شوند. تعارض درجه عضویت با استفاده از اپراتور تقاطع زاده حل می شود [53] بگذارید زیر شاخه های دو لایه A_{n6} و B_{n8} در شکل ها 3 و 4 به ترتیب بررسی کنیم. درگیری مجموعه فازی با استفاده از اپراتور اتحادیه زاده قابل حل است. تضاد مجموعه های فازی با استفاده از اپراتور اجتماع لطفی زاده قابل حل است.

5-2 : ادغام اسناد XML فازی

برای دو سند XML فازی که باید یکپارچه شوند ، یکی به عنوان یک سند XML فازی محلی در نظر گرفته می شود و دیگری به عنوان یک سند XML فازی خارجی در نظر گرفته می شود. ما فرض می کنیم که سند XML محلی فازی و سند XML فازی خارجی یک طرح مشابه دارند به طوری که می توانیم در موارد زیر فقط روی یکپارچه سازی داده ها متمرکز شویم.

سند XML فازی خارجی معمولاً به عنوان یک منبع اطلاعاتی "جدید" برای تکمیل سند XML فازی محلی در نظر گرفته می شود. اگر اطلاعات جدید به اشیاء دنیای واقعی در سند XML محلی فازی وجود نداشته باشد می توانید مستقیماً آنرا به سند XML محلی فازی اضافه کنید. این امکان وجود دارد که اطلاعات جدید در سند XML فازی خارجی مشابه اطلاعات موجود در سند XML فازی محلی باشد. اطلاعات مشابه ممکن است به همان شیء در دنیای واقعی مربوط باشد، که هم در سند XML محلی فازی و هم در سند XML فازی خارجی وجود دارد. در این مرحله ، ما باید اطلاعات مشابه را پس از حل اختلافات احتمالی ناشی از اطلاعات مشابه، با یکدیگر ادغام کنیم. در ادامه ، ما دو قانون برای مقابله با ادغام اسناد XML فازی پیشنهاد می کنیم.

قانون 1 : برای زیردرخت دولایه ریشه در گره n_{bi} از سند XML فازی و زیر لایه دو لایه ریشه در گره n_{fi} سند XML فازی خروجی وجود دارد می گوییم که اگر درجه تشابه آنها $Sim_{FXST}(n_{bi}, n_{fj})$ بزرگتر از حد آستانه ی δ است. به عبارت دیگر ، این دو زیر درخت دارای یک جفت تناقض هستند و به همان شیء در دنیای واقعی اشاره می کنند.

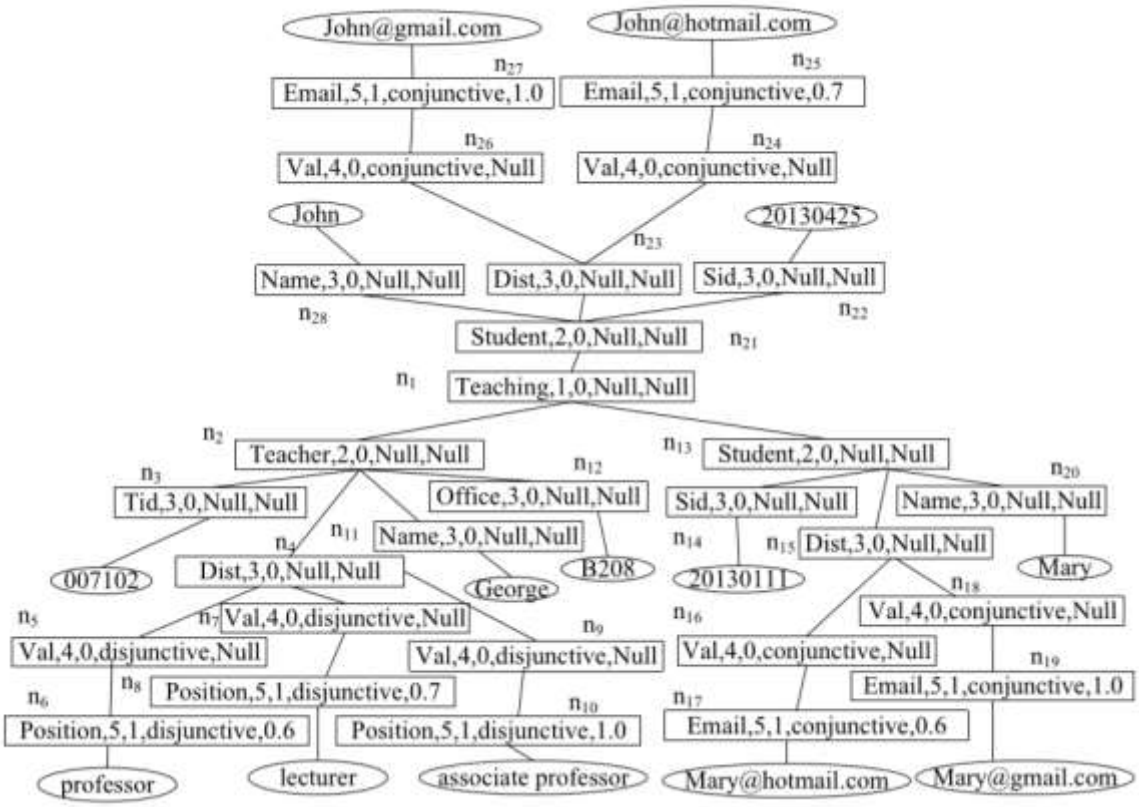
با توجه به اینکه دو زیر درخت مشابه به یک شیء در دنیای واقعی مراجعه می کنند ما فقط باید یک متن فرعی را در سند XML فازی یکپارچه حفظ کنیم. اگر زیر درختان دو لایه زیر درخت هایی با گره های برگ باشد و مقادیر گره های برگ یکسان باشد اختلاف درجه عضویت وجود دارد که با استفاده از رویکرد گفته شده در بخش 5.1 می توان آن را برطرف کرد. اگر زیر درختان دو لایه زیر درخت هایی با گره های بدون لایه باشد و درجه شباهت زیر شاخه های دو لایه که به ترتیب ریشه در گره های فرزند n_{bi} و n_{fj} دارند، از آستانه مشخص شده δ بزرگتر است ، لازم است که زیردرخت ها را به طور بازگشتی مقایسه کنیم تا درخت سند XML فازی خارجی کاملاً پیموده شده است. هنگامی که زیر درختان دو لایه زیر شاخه هایی از گره های بدون برگ باشد و درجه تشابه زیرشاخه های دو لایه که به ترتیب ریشه در گره

های فرزند n_{bi} و n_{fj} دارندکمتر از δ است. زیرشاخه دو لایه ریشه در n_{bi} سند XML فازی محلی قرار دارد. سند XML و زیر لایه دو لایه ریشه در n_{fj} سند XML فازی خارجی فقط شباهت ساختاری دارند. سپس کل زیر درختی که در n_{fj} ریشه دارد باید به درخت سند XML فازی محلی به عنوان یک زیر درخت خواهر و برادر از n_{bi} اضافه شود.

مثال 2 : اجازه دهید نگاهی به دو درخت XML فازی 3 و 4 بیندازیم. ما می توانیم درجه شباهت بین زیر لایه دو لایه ریشه در گره N_3 درخت خارجی B و زیردرخت دو لایه ریشه در گره N_3 درخت محلی A را محاسبه کنیم. درجه شباهت آنها از آستانه مشخص شده بیشتر است. در اینجا این دو زیرشاخه دو لایه نوعی زیر درخت با گره های برگ هستند و به همان شیء در دنیای واقعی اشاره می کنند. ادغام آنها منجر به یک زیرشاخه دو لایه می شود که دارای درجه عضویت Null است و سپس زیر لایه دو لایه ریشه در گره N_3 از B حذف می شود. علاوه بر این زیر لایه دو لایه ریشه در گره N_{11} از B شبیه به زیر لایه دو لایه ریشه در گره N_{10} از A است. اما گره های فرزند این دو زیر لایه دو لایه مشابه نیستند. در این مرحله ما تشخیص می دهیم که زیر لایه دو لایه ریشه در گره N_{11} از B جدید است. در نتیجه اداغام کامل شده با اضافه کردن آن به A و خواهر و برادر های گره ی n_{10} از A و سرانجام زیر درخت دو لایه ریشه در گره N_{11} از B حذف می شود. نتیجه ادغام دو درخت سند XML فازی A و B شکل 3 و 4 در شکل 5 نشان داده می شود.

قانون 2: برای زیردرخت دو لایه که در گره n_{bi} سند XML فازی محلی و درخت زیر دو لایه ریشه در گره n_{fj} سند XML فازی خارجی وجود دارد. می گوییم که اگر شباهت آنها (n_{bi} , n_{fi}) SimFXST کمتر از آستانه δ نباشد مشابه نیستند (یعنی این دو زیردرخت به اشیاء مختلف در دنیای واقعی اشاره دارند).

زیردرخت دولایه ریشه در گره n_{bi} سند XML محلی فازی و زیر لایه دو لایه ریشه در گره n_{fj} سند XML فازی خارجی مهم نیست که زیردرخت هایی با گره های برگ باشند یا گره هایی بدون لایه، در نتیجه کلیه زیرشاخه های ریشه در n_{fj} به عنوان درخت خواهر و برادر زیردرخت گره n_{bi} به درخت سند محلی فازی اضافه می شود و سپس زیر لایه دو لایه ریشه در گره n_{fj} سند XML فازی خارجی باید حذف شود.



شکل 5 یا FIG 5 : درخت درخت سند XML فازی (FXTM) درخت مربوط به سند XML فازی یکپارچه.

مثال 3 : اجازه دهید ما به دو درخت XML فازی که در شکل 3 و 4 داده شده است نگاهی بیندازیم. ما می توانیم درجه شباهت بین زیردرخت دو لایه ریشه در گره n_{10} درخت خارجی B و زیردرخت دو لایه ریشه در گره های n_3، n_4 و n_9 از درخت محلی را محاسبه کنیم. درجه شباهت آنها کمتر از آستانه داده شده است سپس می دانیم که زیردرخت دو لایه ریشه در گره شماره 10 B یک چیز جدید است. در نتیجه ادغام، با اضافه کردن آن به A به عنوان خواهر و برادر گره n_9 در A انجام می شود.

الگوریتم ادغام داده ها IFXD بر اساس قوانین ذکر شده در بالا دو پارامتر FD_b و FD_f را در نظر می گیرد. نتیجه ادغام شده یک FD_t یکپارچه XML فازی است.

ادغام دو سند XML فازی شامل ادغام همه زیر درختان دو لایه هر دو درخت سند XML فازی است. هنگامی که درخت سند فازی خارجی در حال گذر است طبق قوانین ادغام و الگوریتم ادغام در الگوریتم 2، یک زیردرخت دو لایه از سند XML فازی خارجی را می توان به عنوان یک زیردرخت جدید دو لایه به سند XML فازی محلی اضافه کرد. زیرا یک زیرشاخه دو لایه مشابه وجود ندارد که بتوان در سند XML

محلی فازی یافت شود. قوانین ادغام می تواند تضمین کند که تمام اطلاعات جدید موجود در سند XML فازی خارجی را می توان بدون از دست دادن اطلاعات به سند XML محلی فازی اضافه کرد.

به طور کلی ، یک توالی یکپارچه ممکن است برای همه زیر درختان که به اشیاء یکسان یا متفاوت در دنیای واقعی مراجعه می کنند ، چندین نتیجه ممکن داشته باشد. همچنین ، توالی های مختلف یکپارچه ممکن است منجر به یکپارچه سازی مجدد ادغام های مختلف شود. برای ادغام اطلاعات موقعیت معلمان در شکل 3 و 4 می توانیم نتایج ادغام متفاوتی بدست آوریم مهم نیست که کدامیک به عنوان درخت خارجی انتخاب شود. توجه داشته باشید که هر دو نتیجه صحیح است. با استفاده از الگوریتم 2 ، درخت FXTM یکپارچه از اسناد XML فازی شکل 1 و 2 در شکل 5 ارائه شده اند.

Algorithm 2 IFXD.

Input: FD_b, FD_f //Fuzzy XML documents FD_b and FD_f that are to be integrated
Output: FD_t //Integrated Fuzzy XML documents FD_t
Begin
$FT_b = \text{FXTM}(FD_b)$ //FXTM FT_b from FD_b
$FT_f = \text{FXTM}(FD_f)$ //FXTM FT_f from FD_f
For (each n_{b_i} in FT_b) //Two-layer subtree rooted in node n_{b_i}
{For (each n_{f_j} in FT_f) //Two-layer subtree rooted in node n_{f_j}
{ do while (!Empty(FT_f))
{If ($Sim_{FXST}(n_{b_i}, n_{f_j}) > \delta$) //Two-layer subtree similarity degree computation
If (Isleafnodesubtree (n_{b_i}, n_{f_j})) //Subtree rooted in n_{b_i}, n_{f_j} is leaf node subtree
If ($Sim_{Node}(n_{b_i} \rightarrow \text{child}, n_{f_j} \rightarrow \text{child}) > \theta$) //Value equality
{$n_{b_i} \rightarrow \text{child.Poss} = \min(n_{b_i} \rightarrow \text{child.Poss}, n_{f_j} \rightarrow \text{child.Poss})$
deltreesubtree(n_{f_j});} //Deleting subtree rooted in n_{f_j} of the foreign tree
Else //Inserting leaf nodes
IFXD($n_{b_i}, n_{f_j} \rightarrow \text{sibling}$) //Integrate sibling subtree of $n_{f_j}, n_{f_j} \rightarrow$ sibling
Else
If ($\text{Sim}_{FXST}(n_{b_i} \rightarrow \text{child}, n_{f_j} \rightarrow \text{child}) > \delta$ & !Isleafnodesubtree ($n_{b_i} \rightarrow$ child, $n_{f_j} \rightarrow$ child))
IFXD($n_{b_i} \rightarrow \text{child}, n_{f_j} \rightarrow \text{child}$) //Integrate child subtree of n_{b_i}, n_{f_j}
Else //n_{f_j} added as a sibling of n_{b_i}
{InsertSub-tree($n_{b_i} \rightarrow$ common parent, n_{f_j}); deltreesubtree(n_{f_j});}
Else
If (n_{b_i} exist sibling)
IFXD($n_{b_i} \rightarrow \text{sibling}, n_{f_j}$) //Integrate sibling subtree of $n_{b_i}, n_{b_i} \rightarrow$ sibling
Else
{InsertSub-tree($n_{b_i} \rightarrow \text{parent}, n_{f_j}$); deltreesubtree($n_{f_j}$);} //$n_{f_j}$ added as a sibling of n_{b_i}
}
}
}
$FD_t = \text{Transform}(FT_b)$ //FXTM tree is converted to fuzzy XML document
Return FD_t

5-3 پیچیدگی های کلی

پیچیدگی الگوریتم ادغام ارائه شده در بخش 5.2 شامل پیچیدگی زمانی و پیچیدگی فضا است. ابتدا بگذارید پیچیدگی زمان الگوریتم ادغام خود را بررسی کنیم. فرض کنید که دو درخت سند XML فازی A و B را با هم ادغام می کنیم. سپس پیچیدگی زمان الگوریتم ادغام ما به O(|A|2 × |B|2) که در آن |A| و |B| به ترتیب نشانه های اصلی A و B را بیان می کند.

حال اجازه دهید پیچیدگی فضای الگوریتم ادغام خود را بررسی کنیم. برای دستیابی به یکپارچه سازی درختان سند XML فازی و همچنین ماتریس های مسافت و بردارهای معنایی محاسبه شده ، رویکرد ما به فضای حافظه نیاز دارد. پیچیدگی فضایی الگوریتم ادغام ما به O(|A| × |B|) ساده شده است.

6. ارزیابی های تجربی

برای ارزیابی اثربخشی و کارآیی رویکرد ما برای یکپارچه سازی داده های XML فازی ناهمگن و بررسی بیشتر تأثیر پارامترهای مختلف بر روی آنهاست. ما چند آزمایش با مجموعه داده ها انجام دادیم و نتایج تجربی را گزارش می کنیم. در ابتدا چندین معیار ارزیابی و مجموعه داده هایی را که در آزمایش های خود استفاده کرده ایم ارائه می دهیم. سپس نتایج تجربی خود را ارائه می دهیم. آزمایشات ما در JDK 1.6 پیاده سازی شد و روی یک سیستم با پردازنده Intel Core i5 ، با 4 گیگابایت رم و در ویندوز 7 اجرا شد.

6-1 معیارهای ارزیابی

اساساً ، رویکرد ادغام خود را از دو جنبه ارزیابی می کنیم: اول کیفیت خروجی و دیگری کیفیت عملکرد. کیفیت خروجی برای اندازه گیری چگونگی عملکرد داده های نتیجه پس از ادغام با رویکرد ادغام ما می تواند داده های منبع را نشان دهد. برای این منظور ما نتایج تجربی را در برابر نتایج واقعی بدست آمده به صورت دستی مقایسه می کنیم. عملکرد کیفیت برای اندازه گیری سرعت اجرای رویکرد ادغام ما اعمال می شود. زمان اجرا و مقیاس عملکرد را ارزیابی می کنیم.

در مورد کیفیت ادغام ، ما سه اجرا را در نظر می گیریم: صحت ، کامل بودن و معیار اندازه گیری F. به دنبال کار دالماگاس و همکاران[54]، ما نمادی را معرفی می کنیم که در محاسبه اندازه گیری اثربخشی به شرح زیر است:

A تعداد اسناد/عناصر XML فازی است که بطور صحیح توسط سیستم ادغام، ادغام شده اند.

B تعداد اسناد/عناصر XML فازی است که به طور نادرست توسط سیستم ادغام، ادغام شده اند.

C تعداد اسناد / عناصر XML فازی است که باید یکپارچه شوند اما در واقع توسط سیستم ادغام، ادغام نشده اند.

با A، B و C اکنون می توانیم صحت ، کامل بودن و معیار اندازه گیری F را تعریف کنیم.

صحت برای تعیین درجه صحت نتایج ادغام استفاده می شود. به طور رسمی، به عنوان تعداد اسناد/عناصر صحیح یکپارچه تقسیم بر تعداد کل اسناد/عناصر یکپارچه تعریف شده است:

$$Correctness = \frac{A}{A+B}$$

کامل بودن برای ارزیابی میزان کامل بودن نتایج ادغام استفاده می شود. به طور رسمی، به عنوان تعدادی از اسناد/عناصر صحیح یکپارچه تقسیم بر تعداد اسناد/عناصر XML فازی تقسیم شده است.

$$Completeness\ = \frac{A}{A+C}$$

نه صحیح بودن و نه کامل بودن به تنهایی نمی توانند کیفیت خروجی را به طور دقیق ارزیابی کنند. بنابراین ، ما از مفهوم معیار اندازه گیری F برای ارزیابی کیفیت خروجی استفاده می کنیم. معیار اندازه گیری F میانگین هارمونیک وزنی از صحت و کامل بودن است:

$$F - measure = \frac{2 \times Correctness \times Completeness}{Correctness + Completeness}$$

هر سه روش ارائه شده در بالا دارای حداکثر 1.0 و حداقل 0.0 است. در این مقاله ، سه اندازه گیری شده در بالا ، در حقیقت صحت متوسط ، کامل بودن متوسط و میانگین معیار اندازه گیری F است که به ترتیب به میانگین مقادیر اقدامات نسبت به کلیه کارهای ادغام اشاره دارد. اثربخشی ادغام داده ها معمولاً با معیارهای استاندارد صحت ، کامل بودن و معیار اندازه گیری F مشخص می شود. به طور کلی، بیشتربودن صحت، کامل بودن و معیار اندازه گیری F به معنای یک رویکرد ادغام مناسب داده ها است.

6-2 دیتاست ها

ما چندین ارزیابی داده را بر اساس اسناد XML مصنوعی و واقعی برای ارزیابی های تجربی اعمال می کنیم. (تمام دیتاست ها از http://www.cs.washington.edu/research/xmldatasets جمع آوری شده است.)

6-2-2 دیتاست های واقعی

ما NASA.xml و DBLP.xml را به عنوان مجموعه داده های واقعی به ترتیب به ترتیب D3 و D4 می گیریم. برای آزمایش ها در ادغام داده های XML فازی از یک روش تولید داده تصادفی برای تبدیل داده ها در D3 و D4 به داده های فازی استفاده می کنیم. یعنی به صورت مصنوعی اطلاعات فازی را به D3 و D4 اضافه می کنیم و سپس آنها را به مجموعه داده های XML فازی تبدیل می کنیم. علاوه بر این ، برای بررسی تأثیر اندازه سند ، به طور تصادفی هر پرونده XML فازی را به 150 سند تقسیم کردیم. اسناد تقسیم شده مطابق با طرح مشابه فایل XML فازی اصلی مطابقت دارد. اندازه آنها با توجه به توزیع حسابی پیشرفت حسابی از 100 KB تا 10 MB تعیین می شود.

مشخصات D3 و D4 در جدول 2 خلاصه شده است.

Table 1
Characteristics of the synthetic data sets.

Domain	DTD	Number of documents	Number of elements	Number of fuzzy elements	Average depth	Data set
Auction	ebay.dtd	50	156	15	3.75641	D_1
	ubid.dtd	50	342	30	3.76608	
	yahoo.dtd	50	342	30	3.76608	
University	reed.dtd	50	10546	1000	3.19979	D_2
	uwm.dtd	50	66729	5000	3.95243	
	wsu.dtd	50	74557	10000	3.15787	

جدول 1 : ویژگی های مجموعه داده های مصنوعی.

Table 2
Characteristics of the real data sets.

File name	Number of documents	Number of elements	Number of fuzzy elements	Average depth	Data set
NASA.xml	150	476646	5000	5.58314	D_3
DBLP.xml	150	3332130	10000	2.90228	D_4

جدول 2 : ویژگی های مجموعه داده های واقعی.

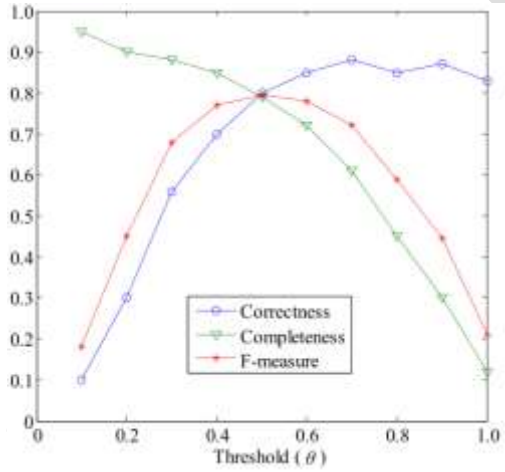
6-3 نتایج تجربی

می خواهیم در آزمایش های خود پیکربندی پارامتر بهینه را پیدا کنیم. مشاهده می کنیم که چگونه پارامترهای θ (به تعریف 9) و δ (نگاه کنید به قانون 1 و الگوریتم 2) در اندازه گیری کیفیت داده های یکپارچه (به عنوان مثال ، صحت ، کامل بودن و معیار اندازه گیری F) می پردازیم. در این سناریو، مجموعه داده مصنوعی D1 به عنوان مجموعه داده های تجربی مورد استفاده قرار می گیرد. در آزمایشات ما اسناد XML فازی را انتخاب می کنیم که درجات تشابه زوجی آنها از آستانه θ فوق بیشتر است و سپس آنها را با هم ادغام می کنیم. با توجه به اسناد XML فازی n، یک گراف کاملاً متصل G را با راس و [(n - 1) × n] / 2 لبه ی وزن دار ساختیم. وزن یک لبه با درجه شباهت بین راس های متصل مطابقت دارد. مطابق الگوریتم ویرایش فاصله Tekli و Chbeir [31] و همچنین الگوریتم TSED ما، درجه های شباهت اسناد XML فازی متصل را می توان با گذر از این درختان سند XML فازی بدست آورد.

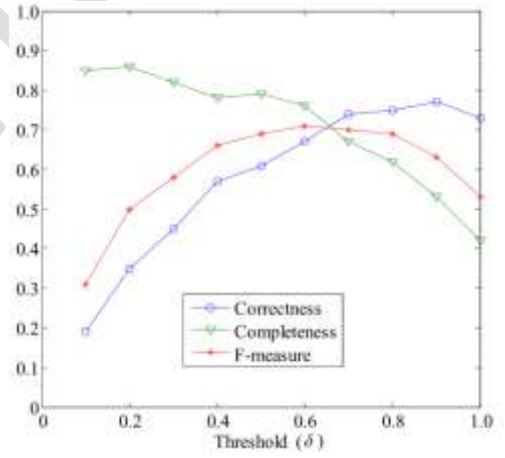
در نتیجه در اسناد XML فازی مشابه با برداشتن لبه هایی با وزن کمتر از θ شناسایی می شوند. چنین اسناد XML فازی مشابهی می توانند یکپارچه شوند. ابتدا ، هر جفت از اسناد XML فازی مشابه با استفاده از رویکرد ما روی مجموعه داده مصنوعی D1 ادغام می شوند. یک سری از کارهای ادغام را با تغییر آستانه در فاصله [1 ، 0] انجام می دهیم. در این مرحله برای نتایج ادغام با آستانه های مختلف همان سؤال مطرح شده برای آنها را اعمال می کنیم و درستی، کامل بودن و معیار اندازه گیری F آنها را مانند شکل های 6 و 7 نشان می دهیم. در شکل 6 و 7 نشان داده شده است که وقتی آستانه ها از 0 تا 1 به صورت متغیر هستند، اسناد متناقض به تدریج در همان مجموعه داده ها قرار می گیرند. واضح است که رویکرد ما کیفیت ادغام متفاوتی با مقادیر آستانه متفاوت θ و δ دارد. دلیل اصلی این

است که تعداد گره ها/زیر درختان مشخص شده که به همان اشیاء موجود در دنیای واقعی مراجعه می کنند با تغییر آستانه تغییر می کنند. در اصل پیدا کردن مسابقات مناسب برای آستانه بالاتر بسیار دشوار است (برای مثال ، آستانه 0.9 تعیین شده است). پس از در نظر گرفتن دیتاست های مصنوعی در مورد صحت، کامل بودن و معیار اندازه گیری F، برای بدست آوردن بهینه سازی کلی در کیفیت ادغام $\theta = 0.5$ و $\delta = 0.7$ را تعیین می کنیم.

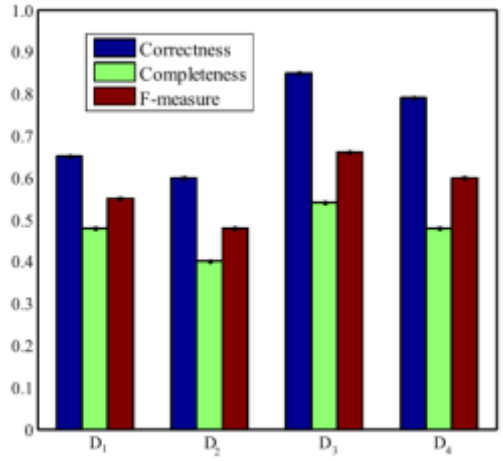
برای آزمایش اثربخشی کیفیت ادغام داده ها را نسبت به مجموعه های مختلف داده ارزیابی می کنیم. در اینجا برای یک دیتاست کیفیت ادغام آن با آستانه های مختلف به همراه متوسط ویژگی های جداگانه داده ها بدست می آید. در شکل 8 نشان داده شده است که خصوصیات ادغام نسبت به D3 و D4 از ویژگی های ادغام نسبت به D1 و D2 بهتر است. این امر عمدتاً به این دلیل است که اسناد XML فازی در مجموعه داده های واقعی یکدست تر از اسناد XML فازی در مجموعه داده های مصنوعی هستند. از طرف دیگر ، وقتی اسناد XML فازی در همان دیتاست یکدست باشد کیفیت یکپارچه سازی داده ها با افزایش اندازه دیتاست کاهش می یابد. به عنوان مثال، کیفیت ادغام داده ها نسبت به D3 از کیفیت ادغام داده ها نسبت به D4 بهتر است.



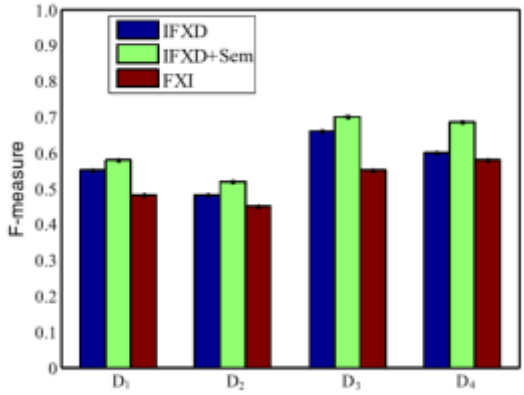
شکل 6 یا FIG 6 : کیفیت داده های یکپارچه بر اساس θ.



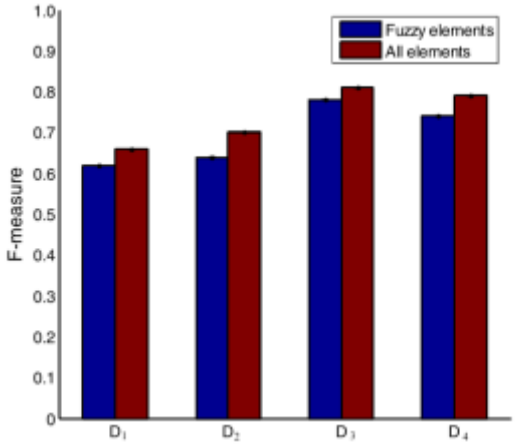
شکل 7 یا FIG 7 : کیفیت داده های یکپارچه بر اساس δ.



شکل 8 یا FIG 8 : مقایسه کیفیت ادغام داده ها با مجموعه های مختلف داده.



شکل 9 یا FIG 9 : مقایسه کیفیت ادغام داده ها با رویکردهای مختلف.

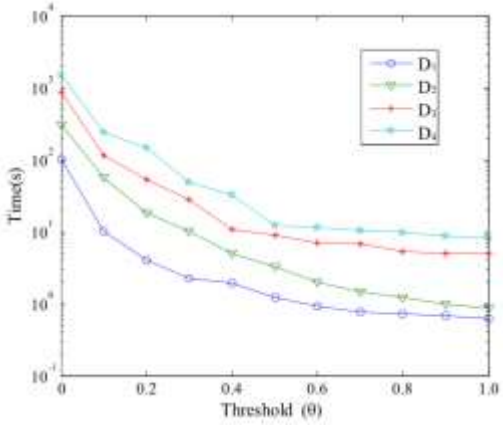


شکل 10 یا FIG 10 : مقایسه کیفیت داده های یکپارچه در سطح عناصر.

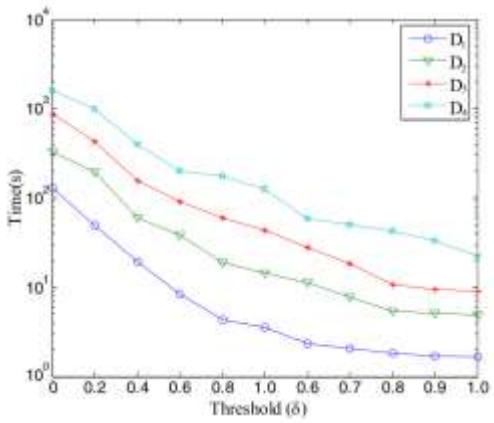
همچنین ، ما رویکرد ادغام IFXD را با رویکرد ادغام XML فازی در در زمینه اقدامات معیار اندازه گیری F مقایسه می کنیم. برای IFXD دو مورد را شناسایی می کنیم: مورد اول، IFXD است که فقط دارای شناسایی ساختاری برای هر جفت گره است با IFXD مشخص شده است و مورد دوم، IFXD با شناسایی ساختاری و معنایی برای هر جفت گره با نام IFXD + Sem است. در شکل 9 نشان داده شده است که (1) هم IFXD و هم IFXD + Sem دارای مقادیر F بالاتر از ادغام XML فازی هستند و (2) اختلاف مقادیر F بین اندازه گیری بین IFXD و IFXD + Sem اندک است.

اولا، در مقایسه با IFXD و IFXD + Sem، ادغام XML فازی یا FXI فاقد مکانیزمی برای پردازش شباهت ساختاری است. در نتیجه به طور موثر زیردرخاتن تکراری آن را به رسمیت نمیش شناسد. بنابراین ، ادغام XML فازی یا FXI مقادیر F نسبتاً کم دارد. دوما، دیتاست ها حاوی تعداد نسبتاً کمی مترادف هستند. بنابراین تفاوت قابل توجهی بین جفت گره های شناسایی شده توسط رویکردهای معنایی و جفت های گره مشخص شده توسط هر دو رویکرد ساختاری و معنایی وجود ندارد. مقادیر F از IFXD نزدیک به مقادیر F از IFXD + Sem است.

برای بررسی بیشتر کیفیت ادغام در سطح عناصر، 20 جفت سند را با یک پیچیدگی ساختاری مشابه از هر مجموعه داده انتخاب می کنیم. سپس می توانیم مقادیر متوسط کیفیت ادغام عناصر (فازی) را بدست آوریم. از شکل 10 دیده می شود که عناصر اسناد XML فازی بطور مؤثر یکپارچه شده اند. در مقایسه با اسناد XML فازی از مجموعه داده های مصنوعی، اسناد XML فازی از مجموعه داده های واقعی دارای کیفیت ادغام بهتری هستند. این امر عمدتاً به این دلیل است که اسناد XML فازی از مجموعه داده های واقعی دارای طرح مشابه هستند. علاوه بر این در شکل 10 نشان داده شده است که معیار اندازه گیری F از عناصر فازی پایین تر از معیار اندازه گیری F همه عناصر است. دلیل اصلی این است که ادغام عناصر فازی باید ویژگی های عناصر بیشتری را در نظر بگیرد.



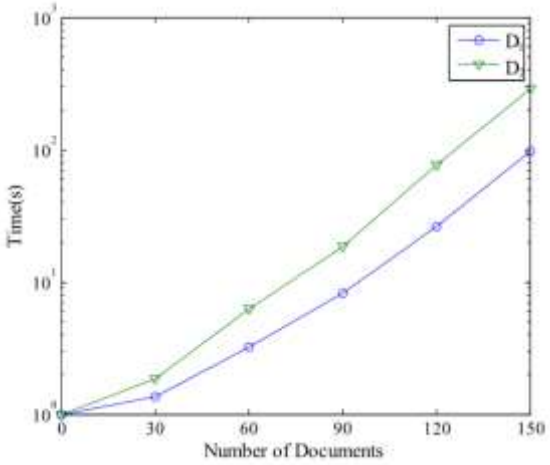
شکل 11 یا FIG 11 : زمان اجرای مجموعه داده های مختلف و آستانه θ.



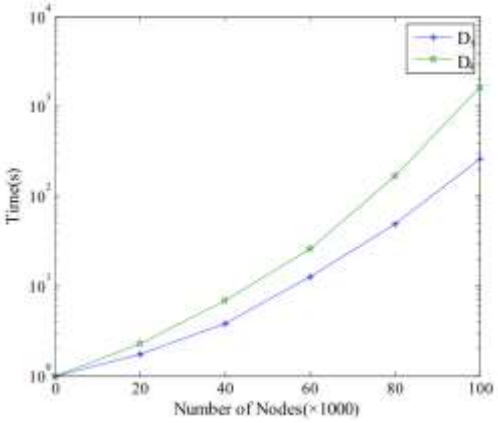
شکل 12 یا FIG 12 : زمان اجرای مجموعه داده های مختلف و آستانه δ.

علاوه بر کیفیت ادغام داده ها ، زمان اجرای آن نیز یک عامل مهم برای ارزیابی استراتژی ادغام است. ما مشاهده می کنیم که پارامترهای θ و δ چگونه بر عملکرد ادغام در مجموعه های مختلف داده تأثیر می گذارد. ما برای انجام آزمایشات خود از کلیه داده های مصنوعی و واقعی استفاده می کنیم. در شکل 11 و 12 نشان داده شده است که زمان ادغام اسناد XML فازی با اندازه و پیچیدگی مجموعه داده ها مشخص می شود. در مقایسه با مجموعه داده های مصنوعی D1 و D2 ، مجموعه داده های واقعی D3 و D4 دارای گره های بیشتری هستند و ساختارهای پیچیده تری دارند. بنابراین ادغام D3 و D4 بیشتر از زمان ادغام D1 و D2 در زمان ادغام هزینه دارد. همچنین در شکل 11 و 12 نشان داده شده است که بسته به مقادیر فزاینده آستانه θ و δ، زمان ادغام کاهش می یابد. این امر به این دلیل است که آستانه در حال افزایش می تواند منجر به ادغام تعداد کمتری از اسناد XML فازی شود.

برای ارزیابی روش ادغام اسناد XML فازی از یک مقیاس بالا برای ارزیابی زمان پردازش ادغام در هنگام افزایش تعداد اسناد/گره ها استفاده می کنیم. اولا ما از مجموعه داده های مصنوعی D1 و D2 استفاده می کنیم تا آزمایش های خود را بر روی تعداد مختلف اسناد انجام دهیم (از 2 تا 150). شکل 13 نشان می دهد که با افزایش تعداد اسناد عملکرد مقیاس خوب است. در مقایسه با ادغام D1 از ادغام D2 در زمان پردازشِ ادغام، هزینه بیشتری دارد. این امر عمدتاً به این دلیل است که D2 پیچیده تر است و شامل گره های بیشتری از عناصر/ویژگی ها است که تأثیر زیادی در عملکرد ادغام دارد. دوما ما از مجموعه داده های واقعی D3 و D4 که حاوی نمونه های گسترده اسناد هستند برای انجام آزمایش های خود بر روی تعداد مختلف گره استفاده می کنیم. شکل 14 نشان می دهد که با افزایش تعداد گره ها زمان پردازش ادغام به صورت خطی افزایش می یابد. همچنین نشان داده شده است که ادغام بیش از D3 در زمان پردازش ادغام نسبت به ادغام در D4 به دلیل میانگین متوسط بسیار بیشتر گره در D4 هزینه کمتری دارد.



شکل 13 یا FIG 13 : مقایسه زمان ادغام برای مجموعه های مصنوعی.

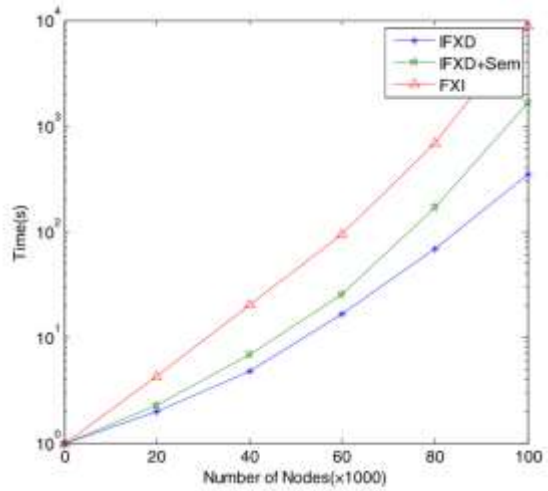


شکل 14 یا FIG 14 : مقایسه زمان ادغام برای مجموعه داده های واقعی.

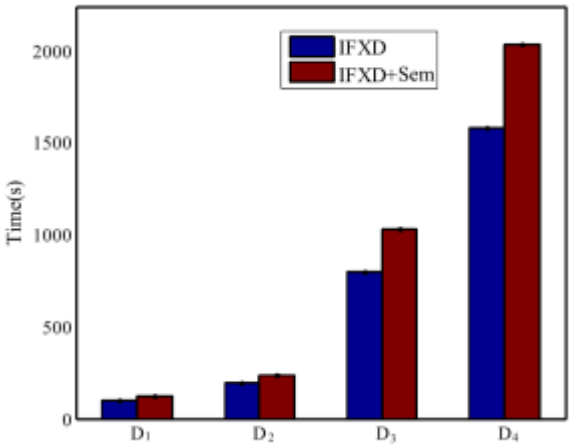
اکنون عملکرد اجرای استراتژی های مختلف ادغام را در یک مجموعه داده داده نشان می دهیم. ما سه استراتژی یکپارچه سازی را اعمال می کنیم که به ترتیب مربوط به FXI، IFXD و IFXD + Sem هستند و آزمایشات خود را بر روی D4 با تنظیمات تجربی مشابه انجام می دهند. در شکل 15 به روشنی نشان داده شده است که سه استراتژی ادغام دارای زمان پردازش نمایی هستند. همانطور که در بخش 4.1 ذکر شد IFXD + Sem به پردازش کلیه اطلاعات معنایی گره ها به زمان اضافی احتیاج دارد. بنابراین IFXD کمتر از IFXD + Sem هزینه دارد. در اصل، عملکرد هر دو استراتژی برای رایانه شخصی قابل قبول است که بتواند اسناد گسترده XML را اداره کند. همچنین در شکل 15 نشان داده شده است که هم IFXD و هم IFXD + Sem هزینه کمتری از FXI دارند. این امر به این دلیل است که FXI برای انتخاب کاندیداها به زمان بیشتری نیاز دارد.

ما همچنین میانگین زمان اجرای متوسط را نشان می دهیم که سه استراتژی ادغام بر روی دیتاست های D1، D2، D3 و D4 اعمال می شود. در شکل 16 نشان داده شده است که ادغام داده ها بر روی مجموعه داده های مصنوعی به وضوح هزینه کمتری نسبت به ادغام داده ها نسبت به مجموعه داده های واقعی دارد. دلیل این امر این است که تعداد زیر درختانی که باید شناسایی شوند و سپس در مجموعه دادههای واقعی یکپارچه شوند بسیار بیشتر از تعدادی است که باید شناسایی شوند و سپس در مجموعه دادههای مصنوعی

ادغام شوند. علاوه بر این ، در شکل 16 نشان داده شده است که IFXD + Sem بیش از IFXD بیش از همان مجموعه های داده (مصنوعی یا واقعی) هزینه دارد.



شکل 15 یا FIG 15 : مقایسه زمان اجرا برای استراتژی های مختلف.



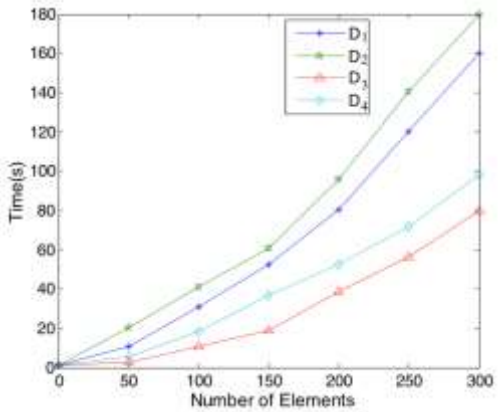
شکل 16 یا FIG 16 : مقایسه زمان اجرای متوسط برای دیتاست های مختلف.

مجموعه داده های مصنوعی علاوه بر این در شکل 16 نشان داده شده است که IFXD + Sem بیش از IFXD و بیش از همان مجموعه های داده (مصنوعی یا واقعی) هزینه دارد.

برای نشان دادن عملکرد اجرای رویکرد ادغام از هر مجموعه داده 20 جفت سند استخراج می کنیم و سپس به ترتیب میانگین زمان اجرای را برای ادغام هر جفت سند محاسبه می کنیم. در شکل 17 نشان داده شده است که ادغام اسناد XML فازی از مجموعه داده های مصنوعی به زمان اجرای بیشتری نسبت به ادغام اسناد XML فازی از مجموعه داده های واقعی نیاز دارد. این به دلیل اصلی بودن اسناد XML فازی از مجموعه داده های مصنوعی ناهمگن است. همچنین ، در شکل 17 نشان داده شده است که میانگین زمان اجرای برای ادغام جفت سند XML فازی بر D2 بیشتر از زمان اجرای متوسط برای ادغام جفت سند فازی XML بر D1 است. میانگین زمان

اجرای برای ادغام جفت سند XML فازی بر D4 بیشتر از زمان اجرای متوسط برای ادغام جفت سند XML فازی بر D3 است. دلیل اصلی این است که اسناد XML فازی بیش از D2 و D4 به ترتیب ساختار پیچیده تری نسبت به اسناد XML فازی نسبت به D1 و D3 دارند.

ما آزمایش های خود را در دو سطح مختلف انجام دادیم: سطح المان ها و سطح سند. نتایج تجربی نشان می دهد که انتخاب پارامترها تأثیر زیادی در کیفیت و عملکرد ادغام داده ها دارد. همچنین ، کیفیت خروجی ادغام داده ها و عملکرد اجرایی با اندازه مجموعه داده ها و همچنین پیچیدگی ساختاری اسناد XML فازی ارتباط نزدیکی دارد. نشان داده شده است که رویکرد ما از کیفیت ادغام داده ها یك مزیت دارد.



شکل 17 یا FIG 17 مقایسه زمان اجرای متوسط برای تعداد مختلف عناصر.

7. نتیجه گیری

برای مقابله مؤثر با ادغام داده های اسناد XML فازی ، در این مقاله ، ما یک مدل درخت XML فازی جدید به نام مدل درخت فازی (FXTM) برای گرفتن اطلاعات ساختاری و معنایی اسناد XML فازی پیشنهاد کردیم. بر اساس FXTM ، ما یک روش اندازه گیری شباهت شامل ساختارهای زیرسطحی دو لایه را پیشنهاد کردیم. علاوه بر این ، ما یک چارچوب برای ادغام داده های XML فازی ناهمگن ارائه کردیم. نتایج تجربی نشان می دهد که رویکرد ما می تواند یکپارچه سازی سند XML فازی را انجام دهد. تازگی رویکرد ما در استفاده از اندازه گیری شباهت از زیر درختان دو لایه است که برای اسناد XML فازی اعمال می شود.

ما چندین موضوع را که باید در ادغام سند XML فازی بررسی شود شناسایی کردیم. اولین مسئله این است که در هنگام ایجاد سند XML فازی یکپارچه ، در نظر گرفتن محدودیت های شماتیک (به عنوان مثال ، محدودیت های کاردینالیتی و محدودیت های یکپارچگی). کار آینده الگوریتم شناسایی هویت را برای شرایطی که محدودیت های شماتیک در زیر لایه های دو لایه رخ می دهد، گسترش می دهد. مسئله دوم در نظر گرفتن تنظیم خودکار آستانه ها و وزن های ارائه شده توسط کاربر است به گونه ای که می توان نتایج یکپارچه سازی بهتری را برای یک دامنه کاربرد واقعی به دست آورد. سرانجام ، در چارچوب ادغام پیشنهادی در این مقاله فرض بر این است که هر یک از منابع

داده در حال ادغام تنها یک سند XML فازی واحد هستند. این امکان وجود دارد که یک منبع داده یکپارچه از چندین سند XML فازی مشابه شود. در این مرحله ، رویکرد پیشنهادی ادغام داده های XML فازی ناهمگن باید برای رسیدگی به چنین مشکلی گسترش یابد.

تشکر ها

نویسندگان از داوران ناشناس بخاطر نظرات و پیشنهادات ارزشمند خود ، که باعث بهبود محتوای فنی و ارائه مقاله می شوند ، تشکر می کنند. این اثر توسط بنیاد ملی علوم طبیعی چین (61772269 و 61370075) پشتیبانی شده است.

مرجع:

[1] L. Zamboulis, XML data integration by graph restructuring, in: Proceedings of the 2004 British National Conference on Databases, 2004, pp. 57–71.

[2] F. Tseng, XML-based heterogeneous database integration for data warehouse creation, in: Proceedings of the 2005 Pacific-Asia Conference on Information Systems, 2005, p. 48.

[3] A. Thomo, S. Venkatesh, Rewriting of visibly pushdown languages for XML data integration, Theor. Comput. Sci. 412 (39) (2011) 5285–5297.

[4] N. Bikakis, et al., The XML and semantic web worlds: technologies, interoperability and integration: a survey of the state of the art, in: Semantic Hyper/Multi-media Adaptation, Springer, Berlin, 2013, pp. 319–360.

[5] S. Abiteboul, L. Segoufin, V. Vianu, Representing and querying XML with incomplete information, ACM Trans. Database Syst. 31 (1) (2006) 208–254.

[6] A. Nierrman, H.V. Jagadish, ProTDB: probabilistic data in XML, in: Proceedings of the 28th International Conference on Very Large Data Bases, 2002, pp. 646–657.

[7] A.D. Keijzer, Data integration using uncertain XML, in: Soft Computing in XML Data Management, Springer, Berlin, 2010, pp. 79–103.

[8] E. Hung, L. Getoor, V.S. Subrahmanian, PXML: a probabilistic semistructured data model and algebra, in: Proceedings of the 19th IEEE International Conference on Data Engineering, 2003, pp. 467–478.

[9] B. Kimelfeld, Y. Kosharovski, Y. Sagiv, Query efficiency in probabilistic XML models, in: Proceedings of the 2008 ACM SIGMOD Interna-tional Conference on Management of Data, 2008, pp. 701–714.

[10] S. Abiteboul, B. Kimelfeld, Y. Sagiv, P. Senellart, On the expressiveness of probabilistic XML models, VLDB J. 18 (5) (2009) 1041–1064.

[11] M.V. Keulen, A.D. Keijzer, Qualitative effects of knowledge rules and user feedback in probabilistic data integration, VLDB J. 18 (5) (2009) 1191–1217.

[12] B. Kimelfeld, Y. Kosharovsky, Y. Sagiv, Query evaluation over probabilistic XML, VLDB J. 18 (5) (2009) 1117–1140.

[13] B. Kimelfeld, P. Senellart, Probabilistic XML: models and complexity, in: Advances in Probabilistic Databases for Uncertain Information Management, Springer, Berlin, 2013, pp. 39–66.

[14] A. Gaurav, R. Alhajj, Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML, in: Proceedings of the 2006 ACM Symposium on Applied Computing, 2006, pp. 456 460.

[15] K. Turowski, U. Weng, Representing and processing fuzzy information – an XML-based approach, Knowl.-Based Syst. 15 (1) (2002) 67–75.

[16] B. Oliboni, G. Pozzani, Representing fuzzy information by using XML schema, in: Proceedings of the 19th International Conference on Database and Expert Systems Application, 2008, pp. 683–687.

[17] J. Lee, Y.-Y. Fanjiang, Modeling imprecise requirements with XML, Inf. Softw. Technol. 45 (7) (2002) 445–460.

[18] Z.M. Ma, L. Yan, Fuzzy XML data modeling with the UML and relational data models, Data Knowl. Eng. 63 (3) (2007) 972–996.

[19] L. Yan, Z.M. Ma, F. Zhang, Fuzzy XML Data Management, Springer, Berlin, 2014.

[20] G. Panic, M. Rackovic, S. Skrbic, Fuzzy XML with implementation, in: Proceedings of the 2012 Balkan Conference in Informatics, 2012, pp.58–62.

[21] G. Panic,´ M. Rackovic,´ S. Škrbic,´ Fuzzy XML and prioritized fuzzy XQuery with implementation, J. Intell. Fuzzy Syst. 26 (1) (2014) 303–316.

[22] X. Yang, M.L. Lee, T.W. Ling, Resolving structural conflicts in the integration of XML schemas: a semantic approach, in: Proceedings of the 2003 International Conference on Conceptual Modeling, Springer, Berlin, 2003, pp. 520–533.

[23] H. Köpcke, E. Rahm, Frameworks for entity matching: a comparison, Data Knowl. Eng. 69 (2) (2010) 197–210.

[24] X.L. Zhang, T. Yang, B.Q. Fan, Novel method for measuring structure and semantic similarity of XML documents based on extended adja-cency matrix, Phys. Proc. 24 (2012) 1452–1461.

[25] A. Nierman, H.V. Jagadish, Evaluating structural similarity in XML documents, in: Proceedings of the 5th International Workshop on the Web and Databases, 2002, pp. 61–66.

[26] A. Poggi, S. Abiteboul, XML data integration with identification, in: International Workshop on Database Programming Languages, 2005, pp. 106–121.

[27] W. Liang, H. Yokota, LAX: an efficient approximate XML join based on clustered leaf nodes for XML data integration, in: British National Conference on Databases, 2005, pp. 82–97.

[28] L. Ribeiro, T. Härder, Entity identification in XML documents, Grundl. Datenbanken (2006) 130–134.

[29] Y. Qi, et al., XML data integration: merging, query processing and conflict resolution, in: Advanced Applications and Structures in XML Processing: Label Streams, Semantics Utilization and Data Query Technologies, IGI Global, Hershey, 2010, pp. 333–360.

[30] S. Agreste, P.D. Meo, E. Ferrara, D. Ursino, XML matchers: approaches and challenges, Knowl.-Based Syst. 66 (2014) 190–209.

[31] J. Tekli, R. Chbeir, A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics, J. Web Semant. 11 (2012) 14–40.

[32] J. Tekli, et al., Approximate XML structure validation based on document–grammar tree similarity, Inf. Sci. 295 (2015) 258–302.

[33] D.D.B. Saccol, C.A. Heuser, Integration of XML data, in: Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, Springer, Berlin, 2003, pp. 68–80.

[34] A.M.D. Nascimento, C.S. Hara, A model for XML instance level integration, in: Proceedings of the 23rd Brazilian Symposium on Databases, Sociedade Brasileira de Computação, Porto Alegre, 2008, pp. 46–60.

[35] A.M. Kade, C.A. Heuser, Matching XML documents in highly dynamic applications, in: Proceedings of the 8th ACM Symposium on Docu-ment Engineering, 2008, pp. 191–198.

[36] M. van Keulen, A. de Keijzer, W. Alink, A probabilistic XML approach to data integration, in: Proceedings of the 21st International Conference on Data Engineering, 2005, pp. 459–470.

[37] T. Pankowski, Reconciling inconsistent data in probabilistic XML data integration, in: Proceedings of the 2008 British National Conference on Databases, 2008, pp. 75–86.

[38] A. Hamissi, B.B. Yaghlane, Belief integration approach of uncertain XML documents, in: Proceedings of IPMU'08, 2008, pp. 370–377.

[39] M.L. Ba, et al., Integration of web sources under uncertainty and dependencies using probabilistic XML, in: Proceedings of the 2014 Interna-tional Conference on Database Systems for Advanced Applications, 2014, pp. 360–375.

[40] J. Liu, X.X. Zhang, Data integration in fuzzy XML documents, Inf. Sci. 280 (2014) 82–97.

[41] L. Yan, Z.M. Ma, J. Liu, Fuzzy data modeling based on XML schema, in: Proceedings of the 2009 ACM Symposium on Applied Computing, 2009, pp. 1563–1567.

[42] G. Nicol, et al., Document object model (DOM) level 3 core specification, W3C Working Draft 13 (2001) 1–146.

[43] R.A. Wagner, M.J. Fisher, The string-to-string correction problem, J. ACM 21 (1) (1974) 168–173.

[44] W. Cohen, P. Ravikumar, S. Fienberg, A comparison of string metrics for matching names and records, in: KDD Workshop on Data Cleaning and Object Consolidation, 2003, pp. 73–78.

[45] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, Sov. Phys. Dokl. 10 (8) (1966) 707–710.

[46] G. Navarro, A guided tour to approximate string matching, ACM Comput. Surv. 33 (1) (2001) 31–88.

[47] J.J. Jiang, D.W. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, in: Proceedings of the International Conference on Research in Computational Linguistics, 1997.

[48] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1995, pp. 448–453.

[49] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.

[50] A. Marie, A. Gal, Boosting schema matchers, in: Proceedings of the OTM 2008 Confederated International Conferences, 2008, pp. 283–300.

[51] S. Madria, K. Passi, S. Bhowmick, An XML schema integration and query mechanism system, Data Knowl. Eng. 65 (2) (2008) 266–303.

[52] Z.M. Ma, L. Yan, Conflicts and their resolutions in fuzzy relational multidatabases, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 18 (2) (2010) 169–195.

[53] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets Syst. 1 (1) (1978) 3–28.

[54] T. Dalamagas, et al., A methodology for clustering XML documents by structure, Inf. Syst. 31 (3) (2006) 187–228.

[55] Z. Ma, L. Yan, Modeling fuzzy data with XML: a survey, Fuzzy Sets Syst. 301 (2016) 146–159.