
بسم الله الرحمن الرحیم

وزارت علوم، تحقیقات و فناوری

پژوهشگاه علوم و فناوری اطلاعات ایران (ایرانداک)

طرح پژوهشی

ایجاد پایگاه داده از تصاویر موجود در پایگاه اطلاعات علمی ایران (گنج) بر اساس یک روش هوشمند

مجری

آزاده فخرزاده

همکار: مجتبی زالی

دی ۱۳۹۸

وزارت علوم تحقیقات و فناوری
پژوهشگاه علوم و فناوری اطلاعات ایران
ایرانداک

شماره: ۳۳,۱۱۱.۳
تاریخ: ۱۳۹۸/۱۰/۰۹
پیوست:

پژوهش، مدیریت دانش، اینترنت

آغاز نیم قرن دوم خدمات ارزشمند ایرانداک به علم، فناوری، و نوآوری گرامی باد

۱۳۹۸-۱۳۴۷

گواهی

انجام طرح پژوهشی:

- **عنوان:** ایجاد پایگاه داده از تصاویر موجود در پایگاه اطلاعات علمی ایران (گنج) بر اساس یک روش هوشمند
- **مجری:** دکتر آزاده فخرزاده
- **همکار:** مهندس مجتبی زالی
- **شمارهٔ قرارداد:** ۳۳/۴۷۸۶
- **تاریخ قرارداد:** ۱۳۹۷/۴/۲۶
- **تاریخ شروع:** ۱۳۹۷/۴/۲۶
- **تاریخ پایان:** ۱۳۹۸/۱۰/۱
- **ناظران:** دکتر مرضیه زرین‌بال ماسوله (ناظر محتوا)، مهندس محمدهادی آقالویی (ناظر فنی)

در پژوهشگاه علوم و فناوری اطلاعات ایران گواهی می‌شود. گزارش پایانی این طرح، بر پایهٔ داوری در نشست ۳۵۷ (تاریخ ۱۳۹۸/۱۰/۸) شورای پژوهشی پژوهشگاه به تصویب رسیده است. لازم می‌دانم مراتب قدردانی خود را به مجری، همکار، و ناظران این طرح تقدیم دارم.

سیدآرزوی بهروزی
پرویز شهریاری
معاون پژوهش و آموزش

تهران . خیابان انقلاب اسلامی . چهارراه فلسطین . شمارهٔ ۱۰۹۰ صندوق پستی ۱۳۷۱-۱۳۱۸۵
تلفن: ۶۶۴۹۴۹۸۰ - ۶۶۹۵۱۴۳۰ دورنگار: ۶۶۴۶۲۲۵۴ تلفن گویا: ۶۶۴۹۴۹۵۴
irandoc.ac.ir
info@irandoc.ac.ir

ایجاد پایگاه داده از تصاویر موجود در پایگاه اطلاعات علمی ایران (گنج) بر اساس یک روش هوشمند

مدیر طرح پژوهشی: آزاده فخرزاده، پژوهشگاه علوم و فناوری اطلاعات ایران.

نشانی: تهران، خیابان انقلاب، چهارراه فلسطین، شماره ۱۰۹۰، طبقه چهارم، اتاق ۴۰۸

صندوق پستی: ۱۳۱۵۷۷۳۳۱۴ **تلفن:** ۶۶۴۹۴۹۸۰-۰۲۱(داخلی ۴۶۱)

وبگاه: https://irandoc.ac.ir/u/a-fakhrzadeh

رایانامه: Fakhrzadeh@irandoc.ac.ir

همکار(ان) اصلی: مجتبی زالی، پژوهشگاه علوم و فناوری اطلاعات ایران.

چکیده

در اسناد و مقالات علمی، تصاویر، حاوی اطلاعات مهمی هستند و در بسیاری از موارد با بررسی آنها به تنهایی می‌توان به ایده اصلی و یا نتایج مهم مقاله علمی پی‌برد، بدون اینکه لازم باشد کل سند را مطالعه کرد. به همین دلیل بسیاری از موتورهای جستجوگر مستندات علمی به دنبال فراهم کردن امکان بازیابی اطلاعات از تصاویر در پایگاه اطلاعاتی خود هستند، به طوری که کاربر با وارد کردن یک جستجو، علاوه بر متن مقالات بتواند به تصاویری هم که به آن جستجو مربوط می‌شود، دسترسی پیدا کند. هم اکنون درپایگاه اطلاعاتی گنج، که حاوی حجم زیادی از مستندات علمی و پایان‌نامه‌ها و رساله‌های فارسی کشور است، امکان جستجو براساس یک عبارت متنی پرس‌وجو و بازیابی و نمایش نتایج جستجو در قالب فرادادههای متنی (عنوان، چکیده، پدیدآور، سال انتشار،) وجود دارد. لیکن در حال حاضر اطلاعات از تصاویر موجود در اسناد گنج بازیابی نمی‌شود. قدم اول برای بازیابی اطلاعات از تصاویر ایجاد پایگاه داده تصاویر از اسناد است. در این طرح سیستمی خودکار برای ایجاد پایگاه داده از تصاویر موجود در مدارک علمی فارسی در مقیاس بزرگ ارائه می‌شود. سیستم پیشنهادی بخش‌های مختلفی دارد. در مرحله اول باید تصاویر و توضیح متنی آن‌ها استخراج گردد. به طور کلی دو رویکرد برای استخراج تصاویر و توضیح متنی آن‌ها از فایل وجود دارد. در رویکرد اول فایل به تصویر تبدیل می‌شود و از تکنیک‌های پردازش تصویر برای استخراج اطلاعات گرافیکی استفاده می‌شود. رویکرد دوم بر اساس پردازش ساختار و آرایش خود فایل است. از آنجایی که روش دوم از لحاظ سرعت و قابلیت مقیاس‌پذیری برای استفاده در موتورهای جستجو مناسب‌تر است، تمرکز این طرح بر روی روش دوم است. بر این اساس برای استخراج تصاویر و توضیح متنی آن‌ها یک روش ساختار محور معرفی می‌شود که مبتنی بر چیدمان و آرایش فایل ورد سند است.بدین ترتیب مجموعه‌ای از تصاویر به همراه توضیحات و اطلاعات مربوط به آن‌ها به دست می‌آید که باید در یک پایگاه داده تصاویر با ساختاری مشخص ذخیره گردند. سپس این اطلاعات برای بازیابی و استفاده‌های آتی در یک موتور جستجو نمایه خواهند شد. در ادامه، روش پیشنهادی در یک مطالعه موردی در پایگاه اطلاعات علمی ایران (گنج) به کار گرفته شد. روش پیشنهادی که با پردازش ساختار و آرایش فایل ورد تصاویر و زیرنویس آن‌ها را استخراج می‌کند در زبان برنامه‌نویسی پایتون پیاده‌سازی شد. استخراج تصاویر از فایل پی.دی.اف هم پیاده‌سازی و بررسی شد. تعداد ۱۵۰ سند علمی به تصادف از پایگاه گنج انتخاب شده و هر دو فایل پی دی اف و ورد آن‌ها مورد تجزیه و تحلیل قرار گرفت. استخراج اطلاعات متنی از فایل پی.دی.اف در زبان فارسی با چالش‌های زیادی روبرو است و نمی‌تواند خروجی مناسبی در این زمینه حاصل کند. از طرف دیگر میزان تصاویر نویز تولید شده از فایل پی.دی.اف بسیار زیاد است که از کاربست‌پذیری آن در شرایط واقعی می‌کاهد. از این رو روش پیشنهادی به عنوان گزینه‌ای مناسب برای استخراج تصاویر و توضیحات آن‌ها از اسناد علمی فارسی موجود در گنج و ایجاد پایگاه داده از آن‌ها پیشنهاد می‌شود.

کلیدواژه‌ها: استخراج تصویر؛ ناحیه بندی تصویر؛ استخراج زیرنویس؛ استخراج فراداده، فناوری اطلاعات

فهرست مطالب

فهرست جداول

عنوان **شماره صفحه**

فهرست شکل‌ها

عنوان **شماره صفحه**

قدردانی

اکنون‌که به یاری خدا طرح پژوهشی "ایجاد پایگاه داده از تصاویر موجود در پایگاه اطلاعات علمی ایران (گنج) بر اساس یک روش هوشمند به پایان رسیده‌است"، از نظرات همکاران پژوهشکده فناوری اطلاعات در تدوین و تصویب پیشنهاده و به‌ویژه سرکار خانم دکتر آزاده محبی، ریاست محترم پژوهشکده تشکر و قدردانی می‌نمایم. از نظرات ارزشمند جناب آقای دکتر حسین صدیقی در نحوه پیاده سازی روش معرفی شده تشکر می نمایم. درنهایت از ناظر گرامی سرکار خانم دکتر مرضیه زرین‌بال که با نظرات ارزشمندشان بر غنای این طرح پژوهشی افزودند، قدردانی ویژه می‌نمایم.

فصل اول

مقدمه و اهداف

۱-۱- مقدمه

نویسندگان مقالات علمی، از نمودار یا تصاویر برای انتقال نتایج کمی‌آزمایش‌های خود و یا فراهم کردن کمک بصری برای توضیح روش پیشنهادی خود بهره می‌برند. موتورهای جستجوگر مقالات علمی، معمولا برای هر جستجو متنی، اطلاعات فراداده متنی مانند عنوان، چکیده، پدیدآور، سال انتشار مقاله را در اختیار کاربرقرار می‌دهد. امروزه با پیشرفت علم دیجیتال و نرم افزارهای مربوطه، تصاویر و نمودارهای موجود در مقالات از اهمیت وپژه ایی برخوردار شده اند و دسترسی به آنها در کنار اطلاعات فراداده متنی می‌تواند خلاصه کاملتری از مقاله به نمایش بگذارد. به همین دلیل موضوع بازیابی اطلاعات از تصاویر موجود در اسناد علمی به یکی از موضوعات جذاب برای افرادی که در حوزه کتابخانه دیجیتال کار می‌کنند، تبدیل شده است و در سال‌های اخیر توجهات زیادی را به خود جلب نموده‌است.

پایگاه اطلاعات علمی ایران(گنج)[۱] دستاورد کاربرد فناوری اطلاعات در خدمات مدیریت اطلاعات علم و فناوری در پژوهشگاه علوم و فناوری اطلاعات ایران (ایرانداک) است. این پایگاه هم‌اکنون مرجع بسیاری از پژوهشگران ایران و جهان است و روزانه بیش از ده هزار کاربر، ده‌ها هزار جست‌وجو در آن انجام می‌دهند. این سامانه بنیان سامانه‌های دیگری در ایرانداک همچون سامانه همانندجو، سامانه پیشینه پژوهش و برخی از داشبوردهای رصدخانه پژوهش و فناوری نیز هست. هم اکنون درگنج امکان جستجو براساس یک عبارت متنی پرس‌وجو و بازیابی و نمایش نتایج جستجو در قالب فراداده‌های متنی وجود دارد. لیکن اطلاعات از تصاویر موجود در اسناد گنج بازیابی نمی‌شود. فراهم کردن چنین امکانی در گنج به صورت یک ارزش افزوده است که می‌تواند آن را از موتورهای جستجوی مشابه متمایز کند.

---

[۱] https://ganj.irandoc.ac.ir/

۲-۱- اهداف طرح

هدف نهایی از تعریف و اجرای این پژوهش فراهم کردن امکان بازیابی اطلاعات از تصاویر موجود در پایگاه اطلاعاتی گنج است. برای این منظور لازم است در ابتدا یک پایگاه داده شامل تصاویر و توضیح گویای متنی آنها ایجاد شود. درمرحله بعد باید امکان جستجو در این پایگاه داده برای کاربر فراهم گردد، به طوری که کاربر بتواند با جستجو در قالب یک عبارت متنی، علاوه بر فراداده‌های متنی اسناد مشابه با عبارت پرس‌جو، به تصاویر مربوط به عبارت پرس‌وجو نیز دسترسی داشته باشد. دراین طرح پژوهشی، به طور خاص، هدف اصلی طراحی و پیاده‌سازی روشی است که از طریق آن بتوان تصاویر را به همراه شرح متنی مختصر آنها از یک سند علمی استخراج نمود. یکی از مهمترین خروجی‌های این طرح در نهایت پایگاه داده‌های تصویری به همراه شرح متنی آنها خواهد بود که از بخشی از داده‌های گنج استخراج شده‌اند . اهداف فرعی این طرح به شرح زیر است :

- بررسی روشهای موجود برای استخراج تصویر از اسناد علمی، و معرفی یک روش که در پایگاه اطلاعاتی گنج قابل استفاده باشد
- بررسی روشهای موجود برای استخراج توضیح متنی تصویر از اسناد علمی، و معرفی یک روش در این زمینه با کاربرد در متون فارسی
- ایجاد پایگاه داده تصاویر و فراهم ساختن امکان بازیابی تصاویر برای کاربران در هر جستجو
-

پرسش‌ها/ فرضیه‌هایی که برای رسیدن به هدف‌های طرح پژوهشی باید پاسخ داده یا بررسی شوند به شرح زیر است:

۱) روش موثر و بهینه برای استخراج تصاویر از متون علمی فارسی چیست ؟ کاستی‌های روش‌هایی که تاکنون پیشنهاد شده‌اند چیست و چگونه می‌توان آنها را جهت به کارگیری در متون فارسی بهبود داد؟

۲) چگونه می‌توان یک توضیح متنی گویا برای هر تصویر از سند مربوط به آن تصویر ، استخراج کرد ؟ روش‌های یادگیری ماشینی برای انجام این فرایندبه صورت خودکار چیست؟

۳) روش ارتباط دادن هر تصویر به توضیح متنی مربوطه چیست؟

۳-۱- روش پژوهش

روش پژوهش برای بررسی انواع روش‌ها و الگوریتم‌های موجود، روش مطالعات کتابخانه‌ای است. با مطالعه تحقیقات پیشین و بررسی چالش‌های مساله پیش رو، روش جدیدی پیشنهاد می‌شود. سپس روش پیشنهادی بر روی مجموعه‌ای از داده‌ها آزمایش می‌شود تا صحت و دقت، وخطای احتمالی آن بررسی گردد و در صورت نیاز بهبودهایی روی آن صورت گیرد. بنابراین، آماده سازی این مجموعه داده برای انجام آزمایش و اندازه گیری دقت روش هم قسمتی از فعالیت‌های این پژوهش محسوب می‌شود.

گامهای اصلی این پژوهش به شرح زیر است:

۱. مطالعه روش‌های موجود در ادبیات برای استخراج هوشمند تصاویر و توضیح متی آنها ازاسناد علمی
   - بررسی روشهای موجود برای استخراج تصویر و متن مربوطه و چالشهای موجود
   - آماده سازی مجموعه داده‌های آزمایشی برای تست روش پیشنهادی

۲. طراحی و پیاده سازی یک روش استخراج تصاویر، ایجاد پایگاه داده و فراهم ساختن امکان بازیابی تصاویر برای کاربران در هر جستجو
   - طراحی روش مناسب برای استخراج هوشمند تصاویر از اسناد علمی
   - طراحی روش مناسب برای استخراج توضیح متنی از اسناد علمی
   - تست و ارزیابی روش پیشنهادی روی مجموعه داده‌های آزمایشی
   - طراحی و ایجاد پایگاه داده تصاویر، متون و متن‌های توضحی تصاویر
   - طراحی و ایجاد ارتباط بین پایگاه داده محلی و پایگاه داده گنج

٤-۱- پیوند با اساس‌نامه و برنامه استراتژیک

طرح پژوهشی حاضر با بازیابی اطلاعات از تصاویر موجود در اسناد علمی گنج در راستای گسترش پوشش مدیریت اطلاعات علمی و فناورانه به اطلاعاتی که کار نشده‌اند و تولید ارزش افزوده است.

فصل دوم

روش‌های موجود در ادبیات برای استخراج تصاویر و زیرنویس آنها از متون علمی

۱-۲- مقدمه

امروزه بیش از میلیون‌ها مقاله و سند علمی در وب وجود دارد (Khabsa and Giles 2014) و همچنان روزی هزاران سند علمی جدید تولید می‌شود. جستجو بین این حجم عظیم مقالات و فهمیدن نتایج و خلاصه آن‌ها برای یک محقق به تنهایی تقریبا غیرممکن است. به همین دلیل موتورهای جستجو سعی می‌کنند از طریق الگوریتم‌های یادگیری ماشین، اطلاعات را از اسناد علمی موجود در پایگاه خود، بازیابی و طبقه‌بندی کنند. با طبقه‌بندی اطلاعات می‌توان به کاربر برای جستجو موثر کمک کرد. تمرکز اکثر کارهایی که در حوزه تحلیل اسناد انجام شده است بر روی متن اسناد است (علایی ابوذر ۱۳۹۷؛ Chan, Ziftci, and Forsyth 2006; Liu et al. 2007; Williams et al. 2014; Wu et al. 2015). الگوریتم‌های داده‌کاوی و نمایه‌سازی موتورهای جستجوی مقالات علمی، مثل گوگل اسکالر[۱]، سایت سیر[۲] و گنج[۳] محدود به متن اسناد است و اطلاعات فراداده متنی مربوط به عبارت جستجو را در اختیار قرار می‌دهند.

ابتدا بحث بازیابی اتوماتیک اطلاعات از تصاویر موجود در مقالات پزشکی مطرح شد (Kim and Yu 2011; Lopez et al. 2011; Xu et al. 2008). با پیشرفت علم دیجیتال ابزارهای پیشرفته‌ای برای تولید تصاویر و نمودارهای علمی در دسترس قرارگرفت. امروزه دراکثر مقالات علمی، ازشکل ها و نمودارها برای ارائه خلاصه از مقاله‌ها و یا نتایج استفاده می‌شود. به همین دلیل، ایجاد یک روش خودکار برای دسترسی و بازیابی اطلاعات از تصاویرموجود در اسناد با موضوعات مختلف، بیش از پیش مورد توجه موتورهای جستجو مقالات علمی قرار گرفته است (Li et al. 2013; Choudhury and Giles 2015; Clark and Divvala 2016; Tsutsi and Crandall 2017; yang et al. 2017; Yu et. al. 2017; Seigel et al. 2018).

---

[۱] https://scholar.google.com /

[2] https://citeseerx.ist.psu.edu

[3] https://ganj.irandoc.ac.ir

هدف نهایی این پژوهش، فراهم کردن امکان بازیابی اطلاعات از تصاویر موجود در اسناد پایگاه داده گنج است. ما در این طرح، به دنبال ایجاد یک پایگاه داده از تصاویر و توضیح متنی آنها هستیم. در آینده از این اطلاعات جهت نمایه سازی تصویر و بازیابی آن استفاده خواهدشد.

به طور کلی مراحل الگوریتم های استخراج تصویر و اطلاعات مربوط به آنها از اسناد علمی به صورت زیر خلاصه می‌شود:

- استخراج تصویر و اطلاعات مربوط به طرح بندی آنها
- استخراج زیرنویس
- همخوان کردن زیرنویس و تصویر

روش‌های استخراج تصویر از فایل را می‌توان به دو گروه تقسیم کرد. دسته اول مبتنی بر روش‌های پردازش تصویر و دسته دوم مبتنی بر پردازش فایل سند است. در رویکرد اول صفحات متن علمی تبدیل به تصویر می‌شود و پس از آن از روش‌های موجود در پردازش تصویر برای ناحیه‌بندی تصویر صفحه استفاده می‌شود تا اطلاعات گرافیکی از بدنه اصلی متن جدا شود. این روش‌ها زیر مجموعه‌ایی ازروش‌های پردازش تصویر اسناد[۱]هستند. پردازش تصویر اسناد یک موضوع تحقیقاتی شناخته شده است که قدمت آن به دهه هشتاد میلادی می‌رسد.(Kumar, et al.; 2007 Fisher, et al., 1990; ; Bloomberg, 1991 ، Srihari, 1986 ; Nagy and Seth, 1984 ; Rehman and Saba, 2006 ; Giles and Choudhury, 2015)

روش دوم بر اساس تجزیه فایل است. به دلیل استقلال فایل پی دی اف از ویژگی‌های نرم افزاری و سخت افزاری سیستم، امروزه مقالات علمی اکثرا به صورت پی دی اف ثبت می‌شوند. به همین دلیل روشهای بازیابی اطلاعات از مقالات در ادبیات بیشتر در مورد فایل پی دی اف است . (Choudhury, et al., 2015; Clark and Santosh, 2016; Furelle, et al., 2003; Choudhury, et al., 2013) .

اجزای فایل پی دی اف مثل متن و تصاویر با استفاده از عمگراهای مختلف و مستقل ایجاد می شوند. به همین دلیل استخراج همزمان تصاویر و زیرنویس مربوطه چالش بر انگیز است. روش کلارک و همکاران (Clark and Santosh, 2016) بر اساس این مشا هده است که معمولا در مقالات علمی ناحیه‌ایی که متن بدنه را در بر نمی گیرد ودر مجاورت زیرنویس قرار دارد ناحیه گرافیکی است. آنها با استفاده از روشهای ابتدایی پردازش تصویر کادرهای محصور کننده بلوکهای متن و گرافیک را در هر صفحه تشخیص می‌دهند. بلوکهای متن با استفاده از یک سری ویژگی های نگارشی به زیرنویس

[۱] Document Image Analysis

و متن بدنه تقسیم بندی می‌شود. در آخر، ناحیه گرافیکی بزرگ در مجاورت بلوک زیرنویس به آن تخصیص داده می‌شود. این روش در مورد تصاویر چندگانه که از چند زیر تصویر تشکیل شده‌اند درست عمل نمی‌کند.

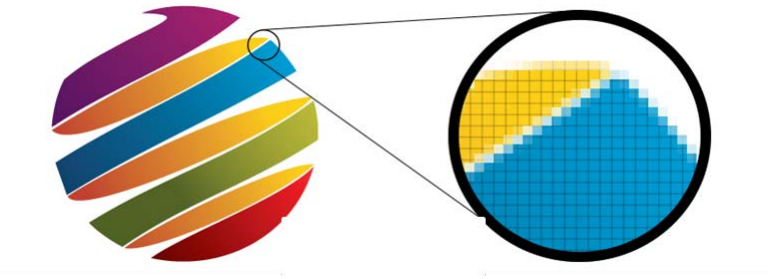
چادهاری و همکاران (Choudhury, et al., 2013) تمام خطوطی را که شامل شناسه تصویر هستند را استخراج می‌کنند. خطوط زیرنویس بر اساس ویژگی های از پیش تعیین شده بر اساس مشخصات ظاهری و نگارشی سایر خطوط تشخیص داده می‌شوند. سپس با استفاده ازنرم افزار PDFBox تصاویر ماتریسی را استخراج می‌کنند. PDFBox موقعیت تصویر در فایل و مشخصات هندسی آن مانند طول و عرض آن را فراهم می‌کند. با توجه به موقعیت تصویر یک مستطیل را زیر تصویر در نظر می گیرند و متن داخل آن را استخراج می‌کنند. از این متن استخراج شده شناسه تصویر را به دست می‌آورند. شناسه به دست آمده با شناسه متناظر با هر زیرنویس مقایسه می‌شود و برای هر تصویر یک زیرنویس در نظر گرفته می‌شود.

۲-۲- استخراج تصویر و اطلاعات مربوط به طرح بندی آنها

تصاویر در فایل پی.دی.اف و ورد به صورت ماتریسی و برداری ذخیره می‌شود. از آنجایی که ویژگی ها و نحوه جایگزاری تصاویر برداری و ماتریسی در فایل متفاوت است، روش استخراج آن‌ها نیز متفاوت است. در ادامه ابتدا تصاویر ماتریسی و برداری را معرفی می‌کنیم و بعد به تفصیل روش‌های موجود در ادبیات برای استخراج تصویر را بررسی می‌کنیم.

تصاویر ماتریسی

یک تصویر ماتریسی ساختار ماتریس نقطه‌ای دارد که اطلاعات به صورت پیکسل به پیکسل ذخیره می‌شود (Wikipedia, Raster graphics). تصویر ماتریسی به صورت فایل تصویری در فرم‌های مختلفی ذخیره می‌شوند. رایج ترین فرمت آنها BMP, PNG,GIF,TIF، JPG است. شکل ۲-۱، نمونه‌ایی از یک تصویر ماتریسی را نشان می‌دهد. تصویر بزرگنمایی نشان می‌دهد که تصویربه صورت ماتریسی ذخیره شده است.

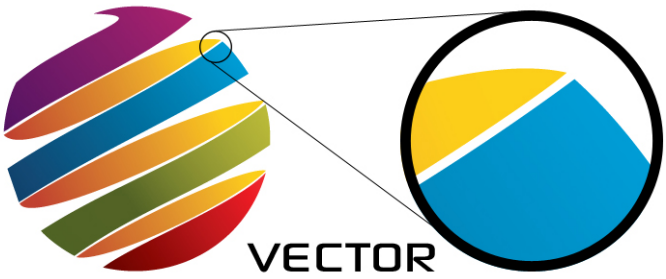


شکل ۲-۱- تصویر ماتریسی (PrintingConection, 1983)

تصاویر برداری

تصاویر برداری توسط نرم‌افزارهای کامپیوتری ساخته شده و هر تصویر در قالب مجموعه‌ای از مشخصات هندسی نقاط، خط‌ها، منحنی‌ها و چندضلعی‌ها ذخیره می‌شود. (Wikipedia Raster graphics)

تصاویر برداری، از خطوط و منحنی‌هایی به نام بردار تشکیل شده‌اند که به صورت ریاضی تعریف می‌شوند. اجزای این تصاویر را می‌توان بدون از دست دادن کیفیت به راحتی جا به جا کرد و تغییر اندازه داد. این تصاویر مستقل از رزولوشن هستند و می‌توان آن‌ها را بزرگ و کوچک کرد و در هر رزولوشن بدون از دست دادن جزئیات و وضوح چاپ کرد. شکل ۲-۲ نمونه ایی از یک تصویر برداری را نشان می‌دهد.



شکل ۲-۲- تصویر برداری (PrintingConection, 1983)

۱-۲-۲- روش‌های استخراج تصویرمبتنی بر پردازش تصویر

در این رویکرد صفحات متن علمی تبدیل به تصویر می‌شود و پس از آن از روش‌های موجود در پردازش تصویر برای ناحیه بندی تصویر صفحه استفاده می‌گردد تا اطلاعات گرافیکی از بدنه اصلی متن جدا شود. به طور کلی سه روش پیاده‌سازی در تحقیقات پیشین پیشنهاد شده است.

- در روش اول تصویر حاصل از سند به بلوک‌هایی تقسیم می‌شود و با یک روش دسته‌بندی مناسب این بلوک‌ها به نواحی متن یا گرافیک تقسیم‌بندی می‌شود (Nagy and Seth, 1984).
- روش دوم، روش‌هایی هستند که بر اساس دسته‌بندی پیکسل‌ها کار می‌کنند. به این معنی که هر پیکسل به عنوان متن یا گرافیک برچسب زده می‌شود . سپس، پیکسل‌هایی که در همسایگی هم هستند و بر چسب آنها یکی است با هم در نظر گرفته می‌شوند و یک بر چسب واحد گرافیک یا متن می‌گیرند(Srihari, 1986; Rehman and Saba, 2006).
- درروش سوم با استفاده از عملگرهای ریخت‌شناسی یک فیلتر گرافیکی طراحی می‌شود (Bloomberg, 1991; Giles and Choudhury, 2015) .

روش بلومبرگ (Bloomberg, 1991) که بر اساس عملگرهای ریخت شناسی است، در نرم افزار لپتونیکا[۱] پیاده سازی شده است و به همین دلیل به تعداد زیاد برای پردازش تصویر مدرک استفاده شده است. لپتونیکا یک نرم افزار متن باز با کاربرد پردازش تصویر اسناد است که در زبان C پیاده سازی شده است. به عنوان یک نمونه از روش‌های استخراج اطلاعات گرافیکی از فایل به روش پردازش تصویر، ما الگوریتم بلومبرگ را در ادامه معرفی می‌کنیم.

روش بلومبرگ

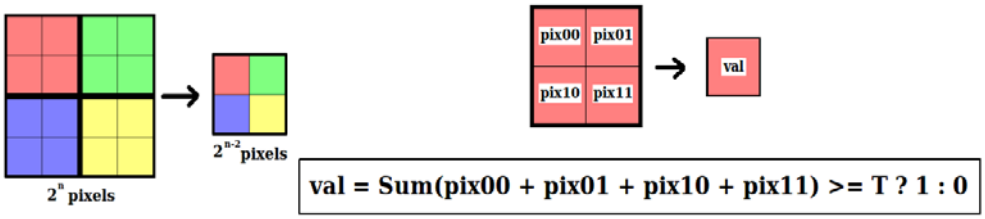
روش بلومبرگ (Bloomberg, 1991)، روش تشخیص متن از تصویر بر اساس تحلیل ریخت شناسی چندتفکیکی[۲] است. بلومبرگ روشی را به نام کاهش آستانه‌ایی معرفی کرده است که به صورت زیر عمل می‌کند. یک تصویر دودویی را در نظر بگیرید که در آن هر شیئ مقدار یک دارد و پس‌زمینه مقدار صفر دارد. تصویر را به بلاک‌های دو در دو تقسیم می‌کنیم. چهار پیکسل موجود در هر بلاک با یک پیکسل در تصویر نمونه برداری شده جایگزین می‌شود. مقدار هر پیکسل در تصویر نمونه برداری شده می‌تواند صفر یا یک باشد و این بستگی به مقدار آستانه انتخابی دارد. مقدار آستانه می‌تواند مقادیر بین یک تا چهار را اختیار کند. مقدار پیکسل نمونه برداری شده یک است اگر مجموع مقادیر چهار پیکسل مساوی یا بزرگتر از مقدار آستانه باشد در غیر این صورت صفر می‌شود (شکل۲-۳). بلومبرگ اعمال آستانه و نمونه برداری از بلوک‌های دو در دو تصویر را کاهش آستانه‌ایی می‌نامد. بعد از یک کاهش آستانه‌ایی تعداد پیکسل‌های تصویر از 2^n به 2^{n-2} کاهش

---

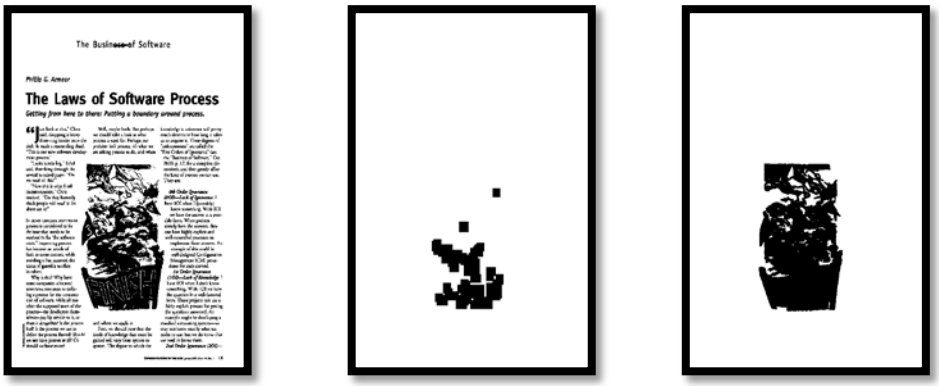
[۱] leptonica.com

[۲] analysis Multiresolution morphology

پیدا می‌کند. بلومبرگ نشان داده است که با اعمال روش کاهش آستانه‌ایی با آستانه‌های به ترتیب یک و سه چهار، اشیاء کوچک موجود در تصویر (در این مورد کلمات) حذف می‌شوند و اشیاء بزرگتر که پیوستگی بیشتری دارند (در این موردنمودارها) باقی می‌مانند. این ترتیب انتخاب آستانه ها در مورد تصاویری که چگالی پیکسلی آنها بالا است بهتر عمل می‌کند. تصویر حاصل هسته نامیده می‌شود (شکل۲- ۴) از این تصویر هسته برای بازیابی شئ متناظر باآن در تصویر اصلی استفاده می‌شود. این روش وابسته به ویژگی‌های نمودار و یا شکل موجود در سند است ودر بعضی موارد نیازمند تنظیم دستی پارامترها (مقادیر آستانه‌ها) هستیم که کلمات به درستی حذف شوند اما نمودارها باقی بمانند.



شکل ۲-۳- توضیح روش ریخت شناسی چند تفکیکی بر اساس روش کاهش آستانه ایی بلومبرگ. در سمت چپ هر چهار پیکسل در هر بلاک دو در دو با یک پیکسل جایگزین می‌شود. مقدار پیکسل جایگزین شده برابر با یک است اگر جمع مقادیر پیکسلهای در هر بلاک بزرگتر یا مساوی مقدار آستانه (T) باشد در غیر اینصورت مقدار آن صفر خواهد بود. تصویر برگرفته شده از(Syed Saqib Bukharia, 2011)



شکل ۲-٤- از چپ به راست: تصویر ورودی ، تصویر هسته ، فیلتر نمودار

۲-۲-۲ روش‌های استخراج تصویرمبتنی بر پردازش فایل

متن و تصاویر به صورت اشیاء جداگانه در فایل پی.دی.اف و ورد ذخیره می‌شوند. در روش‌های مبتنی بر پردازش فایل، فایل به اجزای تشکیل‌دهنده آن تجزیه می‌شود (Futrelle et al. 2003; Lopez et al. 2011; Choudhury et al. 2013; Praczyk and Nogueras-Iso 2013; Clark and

Divvala 2016). این روش‌ها وابسته به فرمت و ساختار فایل بوده و برای فایل ورد و پی.دی.اف متفاوت هستند. روش‌های موجود در ادبیات تنها به فایل پی.دی.اف پرداخته شده است. تصاویر در فایل به صورت ماتریسی و برداری ذخیره شده می‌شود.

استخراج تصاویر ماتریسی

برای استخراج تصویر از فایل پی.دی.اف کتابخانه‌هایی وجود دارد که Apache PDFBOX[۱]، XPDF[۲]، PyPDF[۳]، Poppler[۴] از آن جمله می‌باشد. این کتابخانه‌ها متن باز وبدون هزینه هستند. کتابخانه‌ها به صورتی که وجود دارند، فقط برای استخراج تصاویر ماتریسی مناسب هستند و قادر به استخراج تصاویر برداری نیستند. ، PDFBOX به دلیل سهلولت استفاده و تنوع ابزار، بیش از همه در ادبیات استفاده شدهاست، ما نیز از همین کتابخانه استفاده کرده ایم. در فصل چهارم(۴-۲-۱) جزییات این کتابخانه آمده‌است.

همراه با تصاویر معنی‌دار ماتریسی، تصاویر نویز مانند فرمول‌ها، قسمتی از تصاویر برداری و یا جدول‌ها ذخیره می‌شود. چالش اصلی در اینجا، جداسازی موثر تصاویر نویز از تصاویر مطلوب است. برای فیلتر کردن تصاویر نویز، لوپز و همکاران (Lopez, et al., 2011) روش‌هایی را برای فایل پی.دی.اف پیشنهاد کردند. آنها ابتدا طرح‌بندی[۵] سند که در فایل پی.دی.اف ذخیره شده است استخراج کرده اند. این شامل حدود متن و تصاویر، عرض ستونها و فاصله بین خطوط است. تصاویری که خارج از محدوده متن هستند معمولا لوگو هستند و حذف می‌شوند. در حالت ایده‌آل هر تصویر در سند باید متناظر با یک تصویر در فایل پی.دی.اف باشد. اما در عمل بعضی تصاویر به صورت مجموعه ایی از زیرتصویرها [۶] ذخیره می‌شود. در الگوریتم لوپز و همکاران تصاویر و زیرنویس آنها همزمان استخراج می‌شود و از موقعیت زیرنویس تصاویر برای یکی کردن زیر تصویرها و تشکیل یک تصویر واحد استفاده می‌شود. به این ترتیب که تصاویر پشت سر هم که بعد از آن یک زیرنویس تصویر آماده است را یکی کرده‌اند. روش استخراج زیر نویس هم به این صورت است که تمام پاراگرافهایی که با "Fig"، "FIG" ویا "Figure" شروع می‌شود را به عنوان زیرنویس در نظر می‌گیرند. در طرح‌بندی سند، اطلاعاتی راجع به موقعیت و حد و حدود پاراگراف‌ها ذخیره

---

[۱] https://pdfbox.apache.org

[۲] https://www.xpdfreader.com

[۳] https://pypi.org/project/pyPdf

[۴] https://poppler.freedesktop.org

[۵] Layout

[۶] subfigure

نمی‌شود و لوپز و همکاران هم در مقاله خود توضیحی در مورد روش مشخص کردن پاراگراف‌ها نداده‌اند. چادهاری و همکاران (Choudhury, et al., 2013) چگالی کاراکتر در تصویر، فاصله آن تا سرحدمقاله و مساحت تصویر را در نظر گرفته‌اند و تصاویر نویز را حذف کرده‌اند. کلارک و دیوالا (Clark and Divvala, 2018) هم ابتدا محتوای هر صفحه از فایل پی.دی.اف را به چهار بلوک گرافیک، زیرنویس تصاویر، متن اصلی و متن موجود درگرافیک تقسیم‌بندی کرده‌اند. سپس ناحیه زیرنویس تصویر را استخراج کرده اند و ناحیه گرافیکی بزرگ در مجاورت آن را به عنوان تصویر مربوطه در نظر گرفته‌اند.

بازیابی تصاویر برداری بسیار پیچیده‌تر از تصاویرماتریسی است. تصاویر برداری با مسیر[۱] در فایل پی.دی.اف تعیین می‌شوند. یک تصویر برداری به صورت مجموعه‌ایی از منحنی‌ها و یا خط‌ها ترسیم می‌شوند که مختصات و ویژگی‌های آنها با مسیر مشخص می‌شود. مسیر در واقع یک رشته کاراکتر(و یا عملگر) است که نقطه شروع و پایان و ویژگی منحنی را مشخص می‌کند. سه عملگر مهم که برای نمایش یک مسیراستفاده می‌شود به شرح زیر است:(Giles and Choudhury, 2015)

ساختار مسیر[۲]: عملگرهایی چون (curveto) c , (lineto) l برای تعریف نقطه شروع و انتهای مسیر استفاده می‌شود. یک خط صاف با عملگر"l " کشیده می‌شود و یک منحنی با عملگر "c" .

نگارش مسیر[۳]: عملگرهای (stroke path) S و (close and stroke path) s برای چاپ یک مسیر روی صفحه استفاده می‌شود. این عملگرها رنگ، پهنا و جزییات ظاهری را تعیین می‌کنند.

مسیرهای چیده شده[۴]: عملگره W و W^* برای ساختن مسیرهای چیده شده استفاده می‌شود. مسیرهای چیده شده ناحیه‌ای از صفحه که باید برای ترسیم تصویربرداری استفاده شود را تعریف می‌کند.

از آنجایی که عملگرهای مختلف ذکر شده همه به صورت مستقل در نمایش یک تصویر گرافیکی نقش دارند تجزیه کننده‌های متداول فایل پی.دی.اف از استخراج صحیح این دست از تصاویر باز می‌مانند.

حسن (Hassan, 2009) روشی را معرفی کرده‌است که درآن فایل پی.دی.اف را در یک سطح بالاتر از مسیرها و در سطح شئ ارائه می‌دهد. نرم افزار او[۵](pdfXtk) کادر محصور کننده [۱] تصاویر

---

[۱] path

[۲] Path construction

[۳] Path painting

[۴] Clipping paths

[۵] http://www.tamirhassan.com/html/pdfxtk.html

ماتریسی و گرافیکی را با ترکیب زیر مسیرهای مختلف تولید می‌کند. بعضی از کادرهای محصور کننده که PdfXtk نشان می‌دهد، متعلق به ناحیه گرافیکی نیستند. به عنوان مثال جدول‌ها شامل خطوط و یا نمادهایی هستند که با منحنی ها کشیده میشود و در خروجی PdfXtk دیده می‌شود.
چادهوری و همکاران (Choudhury et al., 2015) مدلی را ارائه کرده‌اند که با آن می‌توان مسیرهای نویز را حذف کرد و مسیرهای مربوط به یک تصویر گرافیکی را هم گروه کرد. آنها یک الگوریتم گروه‌بندی دودویی را در نظر گرفته‌اند که در آن هر مسیر یا به عنوان مسیر گرافیکی (گروه مثبت) یا غیر گرافیکی (گروه منفی) تقسیم بندی می‌شود. آنها چهار ویژگی به شرح زیر در نظر گرفته‌اند: چگالی کاراکترها در ناحیه اطراف مسیر نسبت به چگالی کاراکترهای کل فایل، فاصله مسیر تا سر حد صفحه ، تعداد مسیرهایی که اطراف هر مسیر وجود دارد و مساحت مسیرها . بعد از آزمایش الگوریتمهای دسته بندی مختلف، روش درخت تصمیم را برای دسته‌بندی مسیرها استفاده کرده اند. بعد از آنکه هر مسیر به عنوان مسیر گرافیک یا غیر گرافیک برچسب زده شد، با استفاده از خوشه بندی K-means مسیرهای با برچسب گرافیکی که متعلق به یک تصویر گرافیکی هستند در یک گروه قرار می‌گیرند

۳-۲- استخراج زیرنویس تصاویر

استخراج زیرنویس تصاویردر فایل ورد را با تجزیه تحلیل فایل HTML می‌توان انجام داد. در فایل پی.دی.اف تمام متن پی.دی.اف استخراج می‌شود و بر اساس ویژگی های از پیش تعیین شده زیرنویس تصاویر از خطوط دیگر تشخیص داده می‌شود.

چادهاری و همکاران (Choudhury, et al., 2013) ابتدا تمام جملاتی که شامل کلمات کلیدی همچون "Fig" و یا "fig" و یا مشتقات دیگر آن است ، را پیدا می‌کنند. از این خطوط برای پیدا کردن بلوک زیرنویس استفاده می‌کنند. پیش فرض آنها بر این اساس است که، خط شروع کننده زیرنویس تصویر علاوه بر اینکه شامل شناسه تصویر است، طول کمتری از خط قبلی خود دارد و پایان دهنده زیرنویس تصویر نیز طول کمتری از خط بعدی دارد. در اکثر مواقع زیرنویس تصاویر با حروف درشت شروع می‌شود و سایز فونت زیرنویس تصاویر با بقیه خطوط متن فرق می‌کند.

---

[۱] Bounding Box

آنها مجموعه دیگری از ویژگی ها هم بر اساس علائم دستوری در نظر گرفتند. به عنوان مثال در مقالات انگلیسی جمله شروع کننده زیرنویس تصاویر ترکیب بندی خاصی دارد؛ به این ترتیب که اول شناسه تصویر (Figure 1-3) و بعد علامت ":" و بعد از آن هم اسم می‌آید. چادهاری و همکاران از این ترکیب‌بند برای پیدا کردن شروع کننده زیرنویس استفاده کرده‌اند. بر اساس مشاهدات آنان، پایان دهنده زیرنویس تصاویر در فاصله کمی از شروع کننده زیرنویس وجود دارد و چون جمله پایانی یک پاراگراف است که در آن علائمی‌مثل علامت نقل قول و یا آکولاد و کروشه دیده نمی‌شود. همچنین اگر در خط بعدی، شناسه یک تصویر دیده شود به این معناست که خط بعدی یک زیرنویس تصویر جدید است و خط جاری حتما پایان دهنده زیر نویس است.

کلارک و همکاران (Clark & Santosh , 2016) فرازهایی که شناسه تصاویر را شامل می‌شوند را به عنوان حدس اولیه زیرنویس تصاویر در نظر گرفته اند. برای حذف کردن جملاتی که شناسه تصویر را دارند اما زیرنویس تصویر نیستند (مثبت اشتباه[۱]) با درنظرگرفتن پیش فرض انسجام درنگارش مقاله، ویژگی هایی را برای تشخیص زیرنویس تصاویر استخراج کرده‌اند. اگر در اولین تحلیل چند فراز شامل یک شناسه (مثلا Figure-1) به دست بیاید تنها یکی از این فرازها در نظر گرفته می‌شود و بقیه به عنوان مثبت اشتباه در نظر گرفته می‌شود. فلیتری که برای انتخاب مثبت درست[۲] طراحی کرده اند به شرح زیر عمل می‌کند.

- فرازهایی مثبت درست هستند که شامل نقطه باشند.
- فرازهایی مثبت درست هستند که شامل نقطه ویرگول باشند.
- فرازهایی مثبت درست هستند که شامل فونت درشت باشند.
- فرازهایی مثبت درست هستند که شامل فونت کج[۳] باشند.
- فرازهایی مثبت درست هستند که فونت متفاوتی از کلماتی که در ادامه آن فراز می‌آیند دارند.

بعداز فیلتر کردن نویز، به هر فراز زیرنویس تصویر شامل شناسه، خطی که در ادامه می‌آیند اضافه می شوند به شرط اینکه فاصله آن خط از خط بعدی آن، از مقدار میانه فاصله بین خطوط موجود در سند، کمتر باشد. این بر اساس این فرض است که زیرنویس تصاویر معمولا با فاصله زیادتری از متن اصلی مقاله هستند. در بعضی موارد در مقالاتی که فاصله زیرنویس تصاویر از متن اصلی زیاد نیست ،صرفا در نظر گرفتن فاصله خطوط برای تشخیص جملات مربوط به زیرنویس کافی نیست. برای این موارد

---

[۱] False positive

[۲] True positive

[۳] Italic font

کلارک و همکاران پیشنهاد کرده اندکه یک سری مشخصات ظاهری هم باید در نظر گرفت. اینکه دقیقا چه مشخصاتی را در نظر می‌گیرند ذکر نشده‌است. به مجموعه این جملات زیرنویس، ناحیه زیرنویس تصویر می‌گویند. از مشخصات هندسی این ناحیه برای اختصاص دادن تصویر مربوط به زیرنویس تصویر استفاده می‌کنند که در بخش بعد توضیح آن را می‌آوریم.

٤-۲- همخوان کردن زیرنویس تصویر و تصویر

تصاویر و زیر نویس آن‌ها طی دو فرآیند جداگانه استخراج می‌شود. مرحله آخر تخصیص یک زیرنویس به هر تصویر است. برای اینکار می‌توان موقعیت تصویر را در نظر گرفت و با توجه به اطلاعاتی که در نواحی اطراف تصویر است زیرنویس مربوطه را انتخاب کرد و یا می‌توان موقعیت زیرنویس تصویر را در نظر گرفت و با اطلاعاتی که اطراف آن است تصویر مربوطه را تخصیص داد. چادهاری و همکاران در (Choudhury, et al., 2013) با استفاده از PDFBOX تصاویر ماتریسی را استخراج کرده‌اند. PDFBOX موقعیت تصویر در فایل و مشخصات هندسی آن مانند طول و عرض آن را فراهم می‌کند. با توجه به موقعیت تصویر، یک مستطیل را زیر تصویر در نظر می‌گیرند و متن داخل آن را استخراج می‌کنند. از این متن استخراج شده شناسه تصویر را به دست می‌آورند. قسمتی از زیرنویس تصویر هم در این مستطیل قرار می‌گیرد اما کامل نیست و آنها از روش دیگری که در بخش قبل توضیح دادیم برای استخراج کامل زیرنویس تصویر استفاده می‌کنند. شناسه به دست آمده با شناسه متناظر با هر زیرنویس مقایسه می‌شود و برای هر تصویر یک زیرنویس در نظر گرفته می‌شود. کلارک و همکاران (Clark & Santosh , 2016) ابتدا هر صفحه را به چهار ناحیه گرافیک، متن بدنه ، زیرنویس تصویر و متن موجود در تصویر تقسیم‌بندی می‌کند. روش پیدا کردن زیر نویس در بخش قبل توضیح داده شده است. مرحله بعدی تشخیص متن بدنه است. با استخراج کل متن از فایل، متن‌های موجود در گرافیک و پانویس و سرتیترر هم استخراج می‌شود که جزء بدنه نیستند.

آنها ابتدا رایج‌ترین فونت و سایز فونت و عرض خطوط و فاصله بین خطوط و حاشیه ها در هر سند را پیدا می‌کنند و با استفاده از این اطلاعات آماری قوانین زیر را استخراج می‌کنند:

- هم پوشانی با گرافیک: متنی که با ناحیه گرافیکی هم پوشانی دارد به عنوان متن موجود در تصویردر نظر گرفته می‌شود؛
- متن عمودی: به عنوان متن تصویر در نظر گرفته می‌شود؛
- متن با فاصله زیاد بین کلمات: بلوک‌های متن که فاصله بین کلماتش بیشتر از میزان میانه فاصله کلمات متن است به عنوان متن تصویر در نظر گرفته می‌شود (این امر بیشتر در مورد جداول دیده می‌شود که آن را با تصویر در یک گروه قرار داده اند)؛

- فونت کوچک: متن با فونت کوچکتر از فونت رایج در سند به عنوان متن تصویر در نظر گرفته می‌شود.
- عرض خطوط: بلوک‌های متن که شامل چندین خط می‌شوند که عرض خطوط به اندازه رایج‌ترین عرض خطوط در متن است به عنوان متن بدنه در نظر گرفته می‌شود.
- هم ترازی حاشیه: متنی که هم تراز با حاشیه چپ سند است به عنوان متن بدنه در نظر گرفته می‌شود.
- سرتیتر: متنی که که هم راستا با حاشیه باشد و یا در مرکز باشد، با یک عدد شروع شده باشد و یا با حروف بزرگ نوشته شده باشد و فونت آن و یا اندازه فونت آن غیر معمول باشد به عنوان سرتیتر در نظر گرفته می‌شود که زیر مجموعه‌ایی از متن بدنه است.

وقتی همه نواحی برچسب زنی شد، مرحله آخر اختصاص هر زیرنویس تصویر به ناحیه تصویری است که به آن اشاره دارد. برای هر زیرنویس تصویر تا چهار ناحیه پیشنهادی در نظر گرفته می شود(شکل۲-۵). یک ناحیه مستطیلی در هر سمت زیرنویس تصویر در نظر گرفته می‌شود و آنقدر بسط داده می‌شود که از حاشیه متن بیشتر نشود و وارد ناحیه بدنه متن و یا زیرنویس تصویر دیگر نشود. برای مقالات دو ستونه نواحی پیشنهادی تا خط مرکزی صفحه بسط داده می‌شوند.

از بین چهار نواحی پیشنهادی یک ناحیه انتخاب می‌شود. ناحیه‌ایی که شامل یک ناحیه گرافیکی بزرگ است به عنوان تصویر مربوط به زیرنویس تصویر در نظرگرفته می‌شود. گاهی در ناحیه گرافیکی نویز هم دیده می‌شود که می‌توان با یک روش خوشه‌بندی با مرکزیت قسمت برجسته ناحیه گرافیکی آنها را حذف کرد.

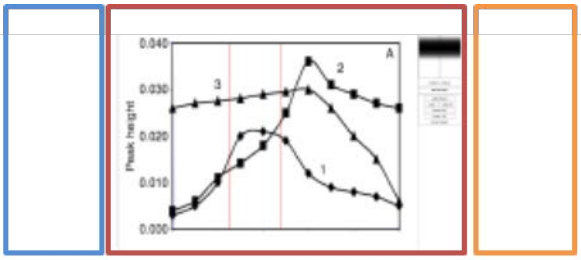


Figure 6: Guiding users in data extraction process. The red line helps to remember the x value for which the user want to extract the y values for different curves.

axis scales (linear/logarithmic) and limiting data values in X and Y axis. If the curves in the image are plotted using separate colors, it is easy to identify them. Binary/grayscale images pose greater challenges. For each point clicked on a

شکل ۲-۵- ایجاد چهارناحیه پیشنهادی برای تصویرمربوط به هر زیرنویس. برای هر زیرنویس (مستطیل سورمه ایی در مرکز) چهار ناحیه قرمز, نارنجی, سبز و فیروزه ایی در نظر گرفته می‌شود

۵-۲- مقایسه روش‌های موجود در ادبیات

استخراج تصویر از سند می‌تواند با تبدیل هر صفحه به تصویر و با استفاده از روش‌های پردازش تصویر انجام پذیرد. ویژگی این روش‌ها به شرح زیر است:

- این روش‌ها در مورد تصاویر ماتریسی و برداری یکسان عمل می‌کند و مستقل از فرمت فایل است.
- فرض اولیه آنها بر این اصل استوار است که تراکم پیکسل‌های حاوی اطلاعات در قسمت گرافیکی بیشتر از متن مقاله است. این روش‌ها در مورد تصاویری با تراکم پیکسل‌های اطلاعاتی کم، مثل نمودارهای خطی نتیجه مطلوب را به دست نمی‌دهد. بنابر این در اکثر مواقع برای گرفتن نتیجه درست نیاز به تنظیم دستی پارامترها است.
- سرعت این روش‌ها از آنجایی که باید هر صفحه را به تصویر تبدیل کنیم و الگوریتم پردازش تصویر را برای هر صفحه اجرا کنیم پایین است.
- این روش‌ها شهودی هستند و قابلیت مقیاس‌پذیری پایینی دارند.

رویکرد دیگر استخراج تصویر، بر اساس تجزیه فایل است. ویژگی این روش‌ها به شرح زیر است :

- این روش‌ها از روش‌های مبتنی بر پردازش تصویر سریع‌تر هستند.
- قابلیت تعمیم پذیری و مقیاس پذیری بیشتری دارند.
- تصاویر برداری با رشته کاراکتر مشخص و تشخیص آنها از متن مشکل است، این روشها در استخراج تصاویر برداری به مشکل برمی‌خورند.
- در مجموعه تصاویر استخراج شده تصاویر نویز هم مثل لوگوها، فرمول‌های ریاضی، بعضی از قسمت‌های تصاویر برداری و جدول‌ها هم ذخیره می‌شود. تصاویر نویز تا حدودی می توانند بر اساس ویژگی هایی همچون سایز و یا موقعیتشان در صفحه حذف گردند.

- این روش‌ها را در مورد فایل‌های اسکن شده، که متن سند در واقع تصویر است، نمی‌توان به‌کار برد.

خلاصه مقایسه این دور رویکرد در جدول۲-۱ آمده است.

جدول۲-۱- خلاصه ایی از مقایسه دو رویکرد اصلی در استخراج تصویراز سند

روشهای استخراج تصویر	مستقل از نوع فایل	سرعت	مقیاس پذیری	پیش فرض در مورد محتوا	استخراج تصاویر برداری
مبتنی بر پردازش فایل	نیست	بیشتر	بیشتر	ندارد	ندارد
مبتنی بر پردازش تصویر	هست	کمتر	کمتر	دارد	دارد

٦-۲- نتیجه گیری

روش‌های موجود در ادبیات برای استخراج تصویر و زیرنویس مربوط به آن بررسی شد. از آنجایی که اکثر مقالات و مستندات علمی به صورت پی.دی.اف ارائه می‌شود در ادبیات تنها به فایل پی.دی.اف پرداخته شده است و فایل ورد بررسی نشده است. به طور کلی دو روش برای استخراج تصاویر و توضیح متنی آن‌ها از فایل وجود دارد. در روش اول فایل به تصویر تبدیل می‌شود و از تکنیک‌های پردازش تصویر برای استخراج اطلاعات گرافیکی استفاده می‌شود. روش دوم بر اساس پردازش ساختار و آرایش خود فایل است از آنجایی که روش دوم از لحاظ سرعت و قابلیت مقیاس‌پذیری برای استفاده در موتورهای جستجو مناسب‌تر است، تمرکز این طرح بر روی روش دوم است.

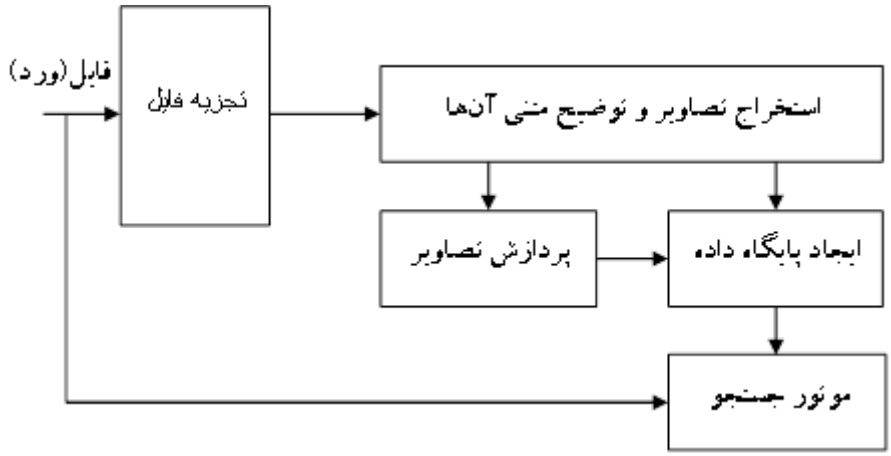
فصل سوم

روش پیشنهادی

۱-۳- مقدمه

در این فصل با استفاده از روش مطالعات اسنادی، سیستمی‌برای ایجاد پایگاه داده از تصاویر مستخرج از اسناد علمی طراحی می‌شود. اسناد علمی موجود در پایگاه گنج به صورت فایل پی.دی.اف و ورد ذخیره شده است. فایلهای پی.دی.اف اگر چه در ظاهر یکی به نظر می‌رسند اما تنوع زیادی در ساختار و طرح بندی دارند. به دلیل این تنوع ساختاری طراحی یک روش که با خطای پایین در مقیاس بزرگ به‌کار برود بسیارمشکل است.از آنجایی که تمام روش‌های موجود در ادبیات از فایل پی.دی.اف استفاده کرده‌اند ما نیز فایل پی.دی.اف را مورد بررسی قرارمی دهیم. به دلیلی مشکلات فایل پی.دی.اف، روش معرفی شده در این فصل در نهایت از فایل ورد اسناد برای استخراج داده‌های تصویر استفاده می‌کند. در ادامه مدل داده‌ای پیشنهادی برای نگهداری داده‌های مربوط به تصاویر به همراه جزئیات آن ارائه می‌شود. در ابتدا دیاگرام رابطه‌ای مدل داده و سپس موجودیت‌های شرکت کننده در این مدل مورد بررسی قرار می‌گیرد. در نهایت راهکار پیشنهادی به منظور نگهداری تصاویر، ارائه و توضیحاتی پیرامون آن آورده می‌شود.

شکل ۳-۱ واحدهای مختلف سیستم طراحی شده را نشان می‌دهد. این سیستم با تجزیه و تحلیل فایل اسناد علمی کار خود را آغاز کرده و در نهایت نتایج را برای بهره‌برداری‌های آتی در موتور جستجو نمایه و آماده بازیابی می‌نماید. از آنجاییکه در ادبیات فایل پی دی اف استفاده شده است ما در این بخش، هر دو فایل ورد و پی دی اف را بررسی کرده ایم. در نهایت به دلیل مشکلات فایل پی دی اف الگوریتم استخراج تصاویر همراه با زیرنویس را برای فایل ورد توسعه دادیم.



شکل ۳-۱- مراحل بازیابی اطلاعات از تصاویر موجود در اسناد علمی

۲-۳- تجزیه فایل

متن و تصاویر به صورت اشیاء جداگانه در فایل پی.دی.اف و ورد ذخیره می‌شوند. همان طور که در فصل دوم گفته شد دو رویکرد برای استخراج تصاویر از اسناد وجود دارد. در رویکرد اول صفحات فایل به فرمت‌های رایج تصویر تبدیل می‌شود و سپس با روش‌های پردازش تصویر ناحیه گرافیکی از متن جدا می‌شود. رویکرد دوم مبتنی بر پردازش فایل است و فایل به اجزای تشکیل‌دهنده آن تجزیه می‌شود. روش‌های مبتنی بر پردازش فایل، وابسته به فرمت و ساختار فایل بوده و برای فایل ورد و پی.دی.اف متفاوت هستند. اکثر این روش‌ها در استخراج تصاویر برداری به مشکل برمی‌خورند. روش‌های مبتنی بر پردازش فایل، در مقایسه با روش‌های مبتنی بر پردازش تصویر، قابلیت مقیاس‌پذیری و سرعت بیشتری دارند و برای به‌کارگیری در موتورهای جستجو انتخاب بهتری هستند. از این رو در این پژوهش از روش پردازش فایل برای استخراج تصاویر و توضیحات متنی کمک گرفته شده است. روش پیشنهادی در این بخش یک روش ساختار محور مبتنی بر چیدمان و آرایش فایل ورد است. فایل پی.دی.اف در ادبیات به کار برده شده‌است به همین دلیل در این طرح فایل پی.دی.اف بررسی شده‌است. به دلیل چالش‌های فایل پی.دی.اف در نهایت روش پیشنهادی برای فایلهای ورد معرفی می‌شود. روش پیشنهادی بر اساس تجزیه و تحلیل فایل XML وردانجام می گیرد.

۱-۲-۳- استخراج تصاویر و توضیح متنی آنها از فایل ورد

برای استخراج تصاویر و متن متناظر با آن‌ها نیازمند بهره‌برداری از ساختار فایل ورد هستیم، از این رو روش پیشنهادی در این بخش یک روش ساختار محور مبتنی بر چیدمان و آرایش فایل ورد است. فایل ورد با پسوند DOCX در واقع یک فایل زیپ شده از مجموعه‌ای از فایل‌های XML است. اگر پسوند یک فایل DOCX را به زیپ تغییر داده و بعد آن را غیر فشرده کنیم مجموعه‌ای از فایل‌ها تشکیل می‌شود. که تصاویر و فایل XML را در اختیار ما قرار می دهد که برای استخراج داده های مورد نظر استفاده می‌شوند. در مورد فایل های قدیمی ورد با پسوند DOC ابتدا آنها را به DOCXتبدیل می کنیم. در پوشه غیر فشرده شده از DOCX، مجموعه‌ای از فایل‌ها و فولدرها مطابق شکل ۳-۲ تشکیل می‌شود. در ادامه مهم‌ترین این فایل‌ها را معرفی خواهیم کرد.

شکل ۳-۲- ساختار فولدربندی و فایل‌های XML مهم در یک فایل ورد

- فایل document.xml فایل XML اصلی است که متن سند و مشخصات ظاهری و طرح‌بندی هر صفحه و دستورالعمل‌های جایگذاری تصاویر و جداول را شامل می‌شود.
- فایل rels. یک مرجع[۱] از محتوای فایل‌های سند را برای نرم‌افزارهای ویرایش متن نظیر مایکروسافت ورد[۲] فراهم می‌کند. مثلا آدرس فایل document.xml را می‌توان در فایل rels. پیدا کرد.
- فایل document.xml.rels یک مرجع از منابع موجود در فایل ورد مثل تصاویر جاسازی‌شده را فراهم می‌کند. تمام تصاویر جاسازی‌شده در سند در پوشه مدیا[۳] به طور جداگانه ذخیره می‌شوند.

---

[۱] Reference

[۲] Microsoft Word

[۳] Media

چالش اصلی، پیدا کردن زیرنویس مربوط به هر تصویر موجود در پوشه مدیا است. جهت استخراج توضیح متنی تصاویر نیاز است فایل document.xml را تجزیه کنیم. برای تجزیه و تحلیل فایل XML روش‌های مختلفی وجود دارد. یکی از روش‌های متداول استفاده از زبان مسیر XML[1] یا به اختصار XPath است که اجازه انتخاب و پیمایش بر روی گره‌های مختلف متناظر با ساختار درختی یک فایل XML را میسر می‌سازد. متن هر سند از تعدادی پاراگراف و جدول تشکیل شده است(شکل ۳-۳). هر پاراگراف که با گره <w:p> مشخص می‌شود می‌تواند شامل متن سند و دستور جایگذاری تصویر باشد. هر جدول که با گره <w:tbl> نشان داده می‌شود خود می‌تواند شامل پاراگراف و یا جدول دیگری باشد. متن که در واقع مجموعه‌ای از کاراکترهاست با گره <w:t> مشخص می‌شود. حدود چهل برچسب مختلف ویژگی‌های ظاهری متن را مثل فونت، سایز، رنگ و قلم مشخص می کنند.

```
<w:document>
   <w:body>
      <w:p>
         <w:pPr/>
         <w:r>
            <w:t>
            </w:t>
         </w:r>
      </w:p>
      …
      <w:tbl>
         <w:tblPr/>
         <w:tblGrid/>
         <w:tr>
            <w:tc>
               <w:tcPr/>
```

[1] XML Path Language

```
                <w:p>
                    <w:r>
                        <w:t>
                        </w:t>
                    </w:r>
                </w:p>
            </w:tc>
            <w:tc>
                …
            </w:tc>
            …
        </w:tr>
        …
    </w:tbl>
    …
</w:body>
</w:document>
```

شکل ۳-۳- ساختار کلی فایل document.xml

فایل DOCX دو نوع تصویر را پشتیبانی می‌کند. تصاویر درون خطی که در داخل پاراگراف همراه با کاراکترهای دیگر ظاهر می‌شوند. گره <w:drawing> برای مشخص کردن تصاویر درون خطی به کار می‌رود. تمام اطلاعات مربوط به سایز و طرح‌بندی تصاویر در بلوک <w:drawing> می‌آید. تصاویر شناور نوع دیگری از تصاویر هستند که مانند کاراکتر متن در نظر گرفته می‌شوند و متن در اطراف آن‌ها جاری می‌شود. تصاویر شناور با گره <wp:anchor> در درون گره <w:drawing> قرار داده می‌شوند. هر تصویر در فایل XML یک شناسه دارد و فایل document.xml.rels مشخص می‌کند که این شناسه مربوط به کدام تصویر استخراج شده در پوشه مدیا است.

استخراج زیرنویس با جستجو در پاراگراف‌های بعد از پاراگراف مربوط به تصویر انجام می‌شود. در بسیاری از موارد زیرنویس تصاویر با استفاده از قابلیت درج زیرنویس نرم‌افزار مایکروسافت ورد،

نوشته می‌شود. در این موارد متن زیرنویس در عنصر <caption> بعد از تصویر ذخیره می‌شود. اگر از قابلیت درج زیرنویس استفاده نشده باشد استخراج زیرنویس پیچیده‌تر می‌شود.

۲-۲-۳- روش پیشنهادی برای استخراج تصاویر و زیرنویس آنها از فایل ورد.

با توجه به آنچه گفته شد در هر سند به دنبال تصاویری هستیم که درون پاراگراف‌ها قرار گرفته‌اند. بدین ترتیب برای هر پاراگرافِ شامل تصویر (پاراگراف هدف)، دو حالت کلی وجود دارد. یا این پاراگراف در بدنه اصلی سند قرار دارد و یا اینکه این پاراگراف داخل یک جدول است. در صورتی که پاراگراف هدف در بدنه اصلی سند باشد، برای استخراج زیرنویس متناظر با آن تنها کافی است که عناصر بعد از این پاراگراف را برای عنصر <caption> جستجو نماییم. در حالتی که پاراگراف درون یکی از سلول‌های جدول (سلول هدف) باشد، سه مکان مختلف باید برای استخراج زیرنویس تصویر جستجو شود. ابتدا باید زیرنویس را در عناصری بررسی کنیم که بعد از پاراگراف هدف و در داخل همان سلول از جدول درج شده‌اند (زیرنویس درون سلولی). سپس باید زیرنویس را در عناصر سلول‌هایی بررسی نماییم که در سطر بعد و در زیر سلول هدف قرار دارند (زیرنویس بین سلولی). در نهایت زیرنویس را باید در عناصر بعد از جدول هدف جستجو کنیم (زیرنویس بعد از جدولی). گام‌های الگوریتم ساختار محور پیشنهادی در ادامه تشریح شده‌اند. لازم به ذکر است که در ابتدا پسوند فایل چک می شود و اگر DOC بود DOCX تبدیل می شود. برای اجرای این دستور در پلتفرم ویندوز باید نرم‌افزار Word و در Linux باید LibreOffice نصب باشد

گام ۱: فایل ورد را بارگذاری و غیر فشرده کن.

گام ۲: فایل document.xml را بارگذاری کرده و ساختار درختی متناظر با آن را استخراج کن.

گام ۳: برای هر پاراگرافی در ساختار درختی فوق موارد زیر را انجام بده:

گام ۳-۱: گره تصویر را پیدا کن.

گام ۳-۲: اگر پاراگراف دربرگیرنده گره تصویر (پاراگراف هدف) داخل یک جدول است به گام ۳-۳ برو، در غیر این صورت به گام ۳-۵ برو.

گام ۳-۳: برای تمام جداولی که پاراگراف هدف را دربرگرفته‌اند موارد زیر را انجام بده:

گام ۳-۳-۱: برای تمام عناصر همرده[۱] پاراگراف هدف موارد زیر را انجام بده:

گام ۳-۳-۱-۱: اگر عنصر بعدی یک پاراگراف بدون عکس و دارای متن بود، متن آن را به عنوان برچسب تصویر استخراج کن و به گام ۳-۳-۲ برو.

گام ۳-۳-۱-۲: اگر عنصر بعدی جدول بود، برای سطرهای آن موارد زیر را انجام بده:

گام ۳-۳-۱-۲-۱: اگر سطر بدون عکس و دارای متن بود، متن آن را به عنوان برچسب تصویر استخراج کن و به گام ۳-۳-۲ برو.

گام ۳-۳-۱-۲-۲: اگر سطر دارای عکس بود، متن مربوط به این سطر را به عنوان برچسب تصویر استخراج کن و به گام ۳-۳-۲ برو.

گام ۳-۳-۱-۳: اگر عنصر بعدی پاراگراف و یا جدول نبود به گام ۳-۳-۲ برو.

گام ۳-۳-۲: سطر و سلول دربرگیرنده پاراگراف هدف (سطر و سلول هدف) را پیدا کن.

گام ۳-۳-۳: برای تمام سلول‌هایی که در سطر بعد از سطر هدف و در زیر سلول هدف قرار دارند موارد زیر را انجام بده:

گام ۳-۳-۳-۱: اگر سلول بدون عکس و دارای متن بود، متن آن را به عنوان برچسب تصویر استخراج کن و به گام ۳-۳-۴ برو.

گام ۳-۳-۳-۲: اگر سلول بدون عکس و متن بود، ابتدا سلول‌هایی که در سطری بعدی زیر آن قرار دارند را پیدا کرده و سپس متن آن‌ها را به عنوان برچسب تصویر استخراج کن و به گام ۳-۳-۴ برو.

گام ۳-۳-۳-۳: اگر سلول دارای عکس و متن بود، متن آن را به عنوان برچسب تصویر استخراج کن و به گام ۳-۳-۴ برو.

---

[۱] Sibling

گام ۳-۳-۳-۴: اگر سلول دارای عکس و بدون متن بود، ابتدا سلول‌هایی که در سطری بعدی زیر آن قرار دارند را پیدا کرده و سپس متن آن‌ها را به عنوان برچسب تصویر استخراج کن و به گام ۳-۳-۴ برو.

گام ۳-۳-۴: برای عناصر همرده جدول دربرگیرنده پاراگراف هدف (جدول هدف) موارد زیر را انجام بده:

گام ۳-۳-۴-۱: اگر عنصر بعدی یک پاراگراف بدون عکس و دارای متن بود، آنگاه اگر عنصر <caption> داشت، متن آن را به عنوان برچسب تصویر استخراج کن. در نهایت به گام ۳-۴ برو.

گام ۳-۳-۴-۲: اگر عنصر بعدی جدول بود، برای سطرهای آن موارد زیر را انجام بده:

گام ۳-۳-۴-۲-۱: اگر سطر بدون عکس و دارای متن بود، آنگاه اگر عنصر <caption> داشت متن آن را به عنوان برچسب تصویر استخراج کن. در نهایت به گام ۳-۴ برو.

گام ۳-۳-۴-۲-۲: اگر سطر دارای عکس بود، آنگاه اگر عنصر <caption> داشت متن مربوط به این سطر را به عنوان برچسب تصویر استخراج کن. در نهایت به گام ۳-۴ برو.

گام ۳-۳-۴-۳: اگر عنصر بعدی پاراگراف و یا جدول نبود به گام ۳-۴ برو.

گام ۳-۴: اگر حداقل یکی از متن‌های استخراج شده فوق عنصر <caption> داشت، تصویر و توضیحات متناظر با آن را ذخیره کن و به گام ۳-۶ برو.

گام ۳-۵: برای عناصر همرده پاراگراف هدف موارد زیر را انجام بده:

گام ۳-۵-۱: اگر عنصر بعدی یک پاراگراف بدون عکس و دارای متن بود، آنگاه اگر عنصر <caption> داشت، متن آن را به عنوان برچسب تصویر استخراج کرده و سپس تصویر و متن متناظر را ذخیره کن. در نهایت به گام ۳-۶ برو.

گام ۳-۵-۲: اگر عنصر بعدی جدول بود، برای سطرهای آن موارد زیر را انجام بده:

گام ۳-۵-۲-۱: اگر سطر بدون عکس و دارای متن بود، آنگاه اگر عنصر <caption> داشت متن آن را به عنوان برچسب تصویر استخراج کرده و سپس تصویر و متن متناظر را ذخیره کن. در نهایت به گام ۳-۶ برو.

گام ۳-۵-۲-۲: اگر سطر دارای عکس بود، آنگاه اگر عنصر <caption> داشت متن مربوط به این سطر را به عنوان برچسب تصویر استخراج کرده و سپس تصویر و متن متناظر را ذخیره کن. در نهایت به گام ۳-۶ برو.

گام ۳-۵-۳: اگر عنصر بعدی پاراگراف و جدول نبود به گام ۳-۶ برو.

گام ۳-۶. اگر گره تصویر دیگری وجود ندارد توقف کن، در غیر این صورت به گام ۳-۱ برو.

۳-۲-۳- استخراج تصاویر و توضیح متنی آنها در فایل پی.دی.اف

تجزیه فایل پی.دی.اف به دلیل تنوع ساختاری از فایل ورد پیچیده تر است. ساختار پایه فایل پی.دی.اف در شکل ۳-۶ نمایش داده شده است. چهار قسمت اصلی فایل پی.دی.اف به شرح زیر است:

سرتیتر[۱]: اولین خط از هر فایل پی.دی.اف سرتیتر است که نسخه پی.دی.اف را مشخص می‌کند:.

%PDF-1.5

بدنه[۲]: بدنه شامل اشیایی است که رشته متن و تصاویر و اجزائ دیگرسند را ایجاد می‌کند. در واقع بدنه پی.دی.اف همه اطلاعاتی که به کاربر نمایش داده می‌شود را نگه می‌دارد.(شکل ۳-۴)

---

[۱] Header

[۲] Body

```
3 3 0 obj <<
4 /Length 138
5 /Filter /FlateDecode
6 >>
7 stream
8 ...
9 endstream
10 endobj
11 10 0 obj <<
12 /Length1 1526
13 /Length2 7193
14 /Length3 0
15 /Length 8194
16 /Filter /FlateDecode
17 >>
18 stream
19 ...
20 endstream
21 endobj
22 12 0 obj <<
23 /Length1 1509
24 /Length2 9410
25 /Length3 0
26 /Length 10422
27 /Filter /FlateDecode
28 >>
29 stream
30 ...
31 endstream
32 endobj
33 15 0 obj <<
34 /Producer (pdfTeX-1.40.12)
35 /Creator (TeX)
36 /CreationDate (D:20121012175007+02'00')
37 /ModDate (D:20121012175007+02'00')
38 /Trapped /False
39 /PTEX.Fullbanner (This is pdfTeX, Version 3.1415926-2.3-1.40.12 (TeX Live 2011) kpathsea version 6.0.1)
40 >> endobj
41 6 0 obj <<
42 /Type /ObjStm
43 /N 10
44 /First 65
45 /Length 761
46 /Filter /FlateDecode
47 >>
48 stream
49 ...
50 endstream
51 endobj
```

شکل ۳-٤- نمونه ایی از دستورات ایجاد بدنه در فایل پی دی اف.

جدول Xref: در واقع یک مرجع متقابل[۱] است. درمرجع متقابل آدرس تمام اشیائ موجود در فایل نشان داده می‌شود و مشخص می‌شود که در کجای فایل ذخیره شده‌اند. این جدول امکان دسترسی رندم به اشیاء موجود در فایل پی.دی.اف را می‌دهد و بنابر این برای دسترسی به یک شئ خاص نیاز نیست کل فایل پی.دی.اف را بررسی کنیم.(شکل ۳-۵)

```
52 16 0 obj <<
53 /Type /XRef
54 /Index [0 17]
55 /Size 17
56 /W [1 2 1]
57 /Root 14 0 R
58 /Info 15 0 R
59 /ID [<1DC2E3E09458C9B4BEC8B67F56B57B63> <1DC2E3E09458C9B4BEC8B67F56B57B63>]
60 /Length 60
61 /Filter /FlateDecode
62 >>
63 stream
64 ...
65 endstream
66 endobj
```

شکل۳ -۵- نمونه ایی از جدول Xref

پشت بند[۲]: پشت بند مشخص می‌کند اشیاء خاص مثل مرجع متقابل فایل پی.دی.اف در کدام قسمت فایل است:

trailer

<<

/Size 22

/Root 2 0 R

/Info 1 0 R

>>

startxref

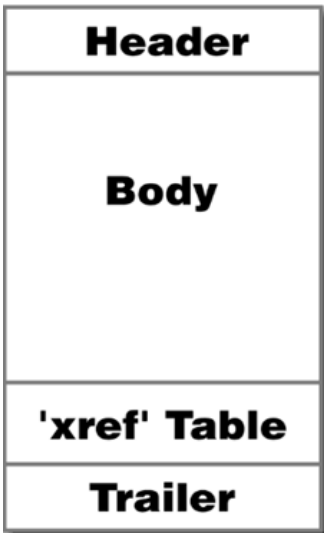
24212

%%EOF

---

[۱] Cross refrence

[۲] Trailer



شکل۳-٦-ساختار داخلی پی دی اف

محتوای فایل پی.دی.اف مثل متن و یا تصاویر و جدول‌ها و دستوالعمل‌های ایجاد آنها، در اشیاء به خصوصی در قسمت بدنه ذخیره شده اند. اکثر این اشیاء فرمت دیکشنری[۱] دارند دیکشنری نوعی متغیر است که به صورت لیستی از، کلیدها و ارزشها[۲] است. دستورالعمل‌های ایجاد متن پی.دی.اف در شئ صفحه[۳] و در زیر مجموعه شئ Content stream قرار دارند(شکل۳-۶) . جهت استخراج تصاویر ما باید سند پی.دی.اف و اشیاء آن را به فرمتی تبدیل کنیم که با نرم افزار قابل تغییر و دست کاری باشند. نرم افزارهای منبع بازی وجود دارند که این کار را انجام می‌دهند. یکی از این نرم افزارهای رایج PDFBox است. PDFBox در جاوا پیاده‌سازی شده است وکار کردن با آن ساده است. با استفاده از PDFBox و نرم افزارهای مشابه می‌توان بدنه فایل پی.دی.اف را تجزیه کرد و متن و تصاویر جایگزاری شده را جداگانه استخراج کرد. یکی از چالش‌های اصلی در پردازش فایل پی.دی.اف استخراج تصاویر برداری است. تصاویر برداری توسط رشته کاراکتر (مسیر) ایجاد می شوند وتشخیص آنها از مسیرهای دیگر که برای ایجاد متن و یا جدول‌ها هستند، به دلیل شباهت ساختاری، کار پیچیده‌ایی است. یک راه رفع این چالش، این است که به جای تجزیه فایل از

---

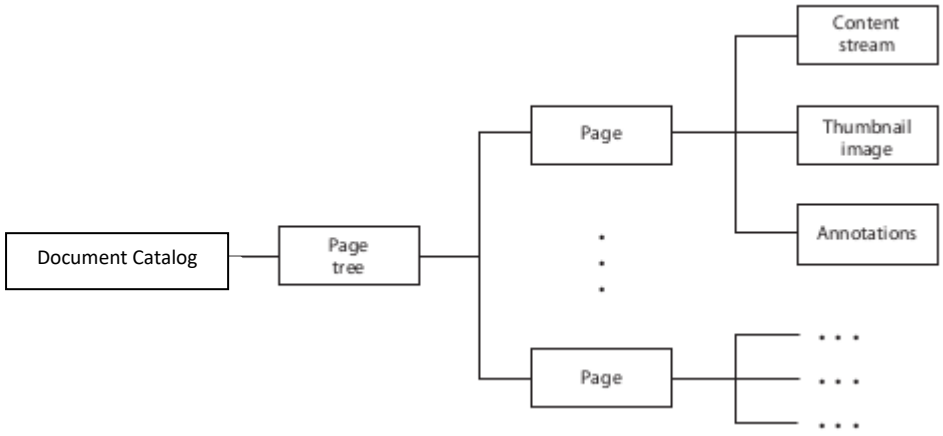
[۱] dictionary

[۲] values

[۳] page

روشهای پردازش تصویر استفاده کنیم، که به دلایلی که در مقدمه توضیح داده شد این روش برای موتور جستجو مناسب نیست.

چالش دیگر در مورد فایل‌های پی.دی.اف، استخراج زیرنویس تصاویر است. متن زیرنویس تصویر با برچسب خاصی درفایل پی.دی.اف مشخص نمی‌شود. برای تشخیص زیرنویس از بقیه متن موجود در سند، ناگزیر از استفاده از روش‌های یادگیری ماشین هستیم. جهت استفاده از این روش‌ها، نیاز داریم ویژگی‌های مشترکی بین خطوط زیرنویس در تمام اسناد پیدا کنیم به طوری که آنها را از بقیه خطوط متن متمایز کند. در سامانه گنج پایان نامه ها از دانشگاه های سراسر کشور جمع آوری شده است و استاندارد خاصی برای نوشتن متن اسناد موجود در گنج، وجود ندارد، همین امر باعث تنوع زیاد اسناد می‌شود و پیدا کردن بردار ویژگی‌ها برای تشخیص زیرنویس تصویر را بسیار پیچیده می کند. از آنجایی‌که در پایگاه گنج ما به فایل ورد اسناد هم دسترسی داریم پیشنهاد می‌کنیم از فایل ورد اسناد برای ایجاد پایگاه داده تصاویر استفاده شود.



شکل ۳-۷-آرایش درخت اشیاء پی دی اف

۳-۳- ایجاد پایگاه داده تصاویر

بعد از آنکه تصاویر و اطلاعات مربوطه از فایل استخراج شد، باید آن‌ها را مطابق با یک مدل داده‌ای که توسط موتور جستجوگر قابل استفاده باشد ذخیره‌سازی کنیم. برای طراحی و پیاده‌سازی مدل داده

ای از convention ها و استانداردهای پیشنهاد شده در MySQL reference manual استفاده شده است.

۱-۳-۳- دیاگرام رابطه‌ای مدل داده‌ای پیشنهادی

شکل ۳-۱ دیاگرام رابطه موجودیت‌ها را برای پایگاه داده تصاویر نشان می‌دهد. در این شکل ارتباط بین جدول مربوط به تصاویر (images) با دیگر جداول مربوط به اسناد علمی قابل مشاهده است. مطابق آنچه دیده می‌شود رابطه بین جدول اسناد (articles) با جدول images یک به چند[1] است؛ بدین معنا که برای هر سند چند تصویر می‌تواند وجود داشته باشد. رابطه بین جدول articles با جداول سازمان‌ها (organizations)، کلیدواژه‌ها (keywords)، و پدیدآوران (Researcher) چند به چند است یعنی به طور مثال هر سند می‌تواند چند کلیدواژه و هر کلیدواژه می‌تواند در چندین سند وجود داشته باشد. کلیدهای خارجی با رنگ نارنجی در این شکل متمایز شده‌اند و وظیفه آن‌ها ایجاد ارتباط بین جدول‌های مختلف است.

One to Many

شکل ۳-۸-نمودار رابطه موجودیت‌ها برای پایگاه اطلاعاتی تصاویر

موجودیت Article

این موجودیت دربرگیرنده اطلاعات مربوط به تمام اسناد موجود در پایگاه گنج است. این اسناد شامل انواع مختلفی از قبیل پایان‌نامه، رساله، پیشنهاده، و ... است. برای پیاده سازی پروژه اطلاعات این موجودیت در اختیار مجری قرار گرفت.

موجودیت Images

این موجودیت در برگیردنده اطلاعات اصلی مربوط به تصاویر استخراج شده خواهد بود. جزئیات صفت‌های این موجودیت در جدول ۳-۱ قابل مشاهده است. توضیحات این جدول به شرح زیر است.

۱) id: شناسه اصلی هر تصویر در پایگاه داده است که نوعا به صورت ترتیبی افزایش می‌یابد.

۲) article-id: شناسه سندی که تصویر از آن استخراج شده است را نشان می‌دهد. این شناسه در نقش یک کلید خارجی به جدول اسناد است.

۳) uid: شناسه کد شده است که یک رشته ۳۲ بیتی می‌باشد. به‌منظور نگهداری امن تصاویر استخراج شده لازم است، نامی که برای هر تصویر انتخاب می‌شود معادل با شناسه کد شده آن باشد.

۳) title_fa: زیرنویس فارسی تصویر است.

۵) title_en: چنانچه زیرنویس تصویر انگلیسی باشد در این قسمت ذخیره می‌شود.

در ردیف‌های ۶ الی ۸ مشخصات ظاهری تصویر ذخیره می‌شوند. موارد ۹ و ۱۰ تاریخ ورود و آخرین به‌روزرسانی داده در پایگاه اطلاعاتی است.

جدول ۳-۱- مدل داده برای ایجاد پایگاه اطلاعاتی تصاویر

ردیف	مشخصه	نوع داده‌ای	توضیح
۱	id	Integer	کلید اصلی
۲	article_id	Integer	شناسه مدرک، کلید خارجی به جدول (Article)
۳	uid	String	شناسه کد شده جدول
۳	title_fa	String	عنوان فارسی تصویر
۵	title_en	String	عنوان انگلیسی تصویر
۶	size	Integer	اندازه تصویر به کیلوبایت
۷	height	Integer	ارتفاع تصویر به پیکسل
۸	width	Integer	عرض تصویر به پیکسل
۹	created_at	Datetime	تاریخ ایجاد رکورد
۱۰	updated_at	Datetime	تاریخ آخرین به‌روزرسانی رکورد

نگهداری تصاویر

به‌منظور نگهداری امن تصاویر استخراج شده لازم است، نامی که برای هر تصویر انتخاب می‌شود معادل با شناسه کد شده آن باشد (uid). همچنین برای سازماندهی و نگهداری بهتر تصاویر، پیشنهاد می‌شود که تصاویر با کمک کلید اصلی آن‌ها (id) دسته‌بندی شود. از آنجایی که برای تعداد فایلهایی که در یک پوشه در ویندوز ذخیره می شود محدودیت وجود دارد، هر N تصویر در یک پوشه نگهداری می شود. عدد بهینه مربوط به N با توجه به درنظر گرفتن فاکتورهایی از قبیل محدودیت‌های فایل سیستم‌های مختلف، حجم میانگین کلیه تصاویر استخراج شده، شمار میانگین تصاویر برای یک مدرک و ... تعیین شود. در الگوریتم ارائه شده N به صورت یک پارامتر آزاد در اختیار کاربر قرار می‌گیرد.N در این طرح ۲۰۰۰ انتخاب شده است.

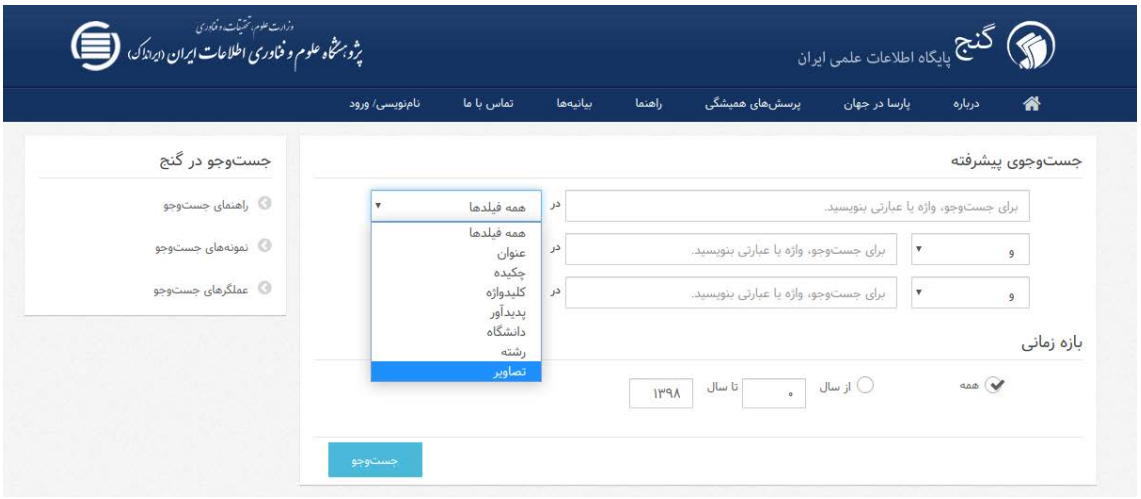
همچنین برای نمایش تصاویر در سایت گنج، این نیاز وجود دارد که برای هر تصویر یک thumbnail از پیش تولید شده و با نامی مشابه با تصویر اصلی (همانند thumbnailuid.jpg) درکنار آن موجود باشد. لازم به ذکر است که همهٔ تصاویر باید دارای یک file

٤-۳- موتور جستجو

در این واحد اطلاعات ذخیره شده در پایگاه داده ایجاد شده، در موتور جستجو نمایه شده و جهت بازیابی اطلاعات مورد استفاده قرار خواهد گرفت. لازم به ذکر است که می‌توان نمایه‌سازی را از جهات مختلفی نظیر محتوای اطلاعاتی برچسب تصاویر، مشخصات ظاهری تصاویر و اطلاعات مربوط به برچسب‌های حاصل از پردازش تصاویر انجام داد. اهمیت استفاده از هر یک از این دسته دادگان بسته به نوع کاربری مورد انتظار از بازیابی تصاویر نمایه شده متفاوت است.برای جست‌وجو در تصاویر استخراج شده از اسناد، بخش جست‌وجوی پیشرفته پایگاه گنج توسعه داده شده‌است. در این بخش، همانند شکل ۳-۹، امکان محدود کردن برای جست‌وجو در اطلاعات(زیرنویس) استخراج شده از تصاویر وجود دارد.پایگاه گنج با استفاده از زبان برنامه نویسی ruby و چارچوب توسعه نرم‌افزار Ruby on Rails توسعه داده شده است. برای تعامل پایگاه گنج با موتور جست‌وجوی solr از کتابخانه sunspot استفاده شده است. این کتابخانه برای چارچوب rails توسعه داده شده است و این امکان فراهم می‌کند که با زبان ruby بتوان تنظیمات مربوط به solr را پیکره‌بندی کرد. با استفاده از دستورات زیر در گنج، تنظیمات مربوط به نمایه سازی شدن فیلدهای اطلاعاتی تصاویر انجام شده است. لازم به اشاره است که دیگر تنظیمات مربوط به موتور جست‌وجوی solr در پایگاه گنج بدون تغییر باقی‌مانده است.

```
searchable :auto_index => false do
  text :images_title do
    images.map { |image| image.title_fa.present? ? image.title_fa.to_s :
image.title_en.to_s }
  end
```

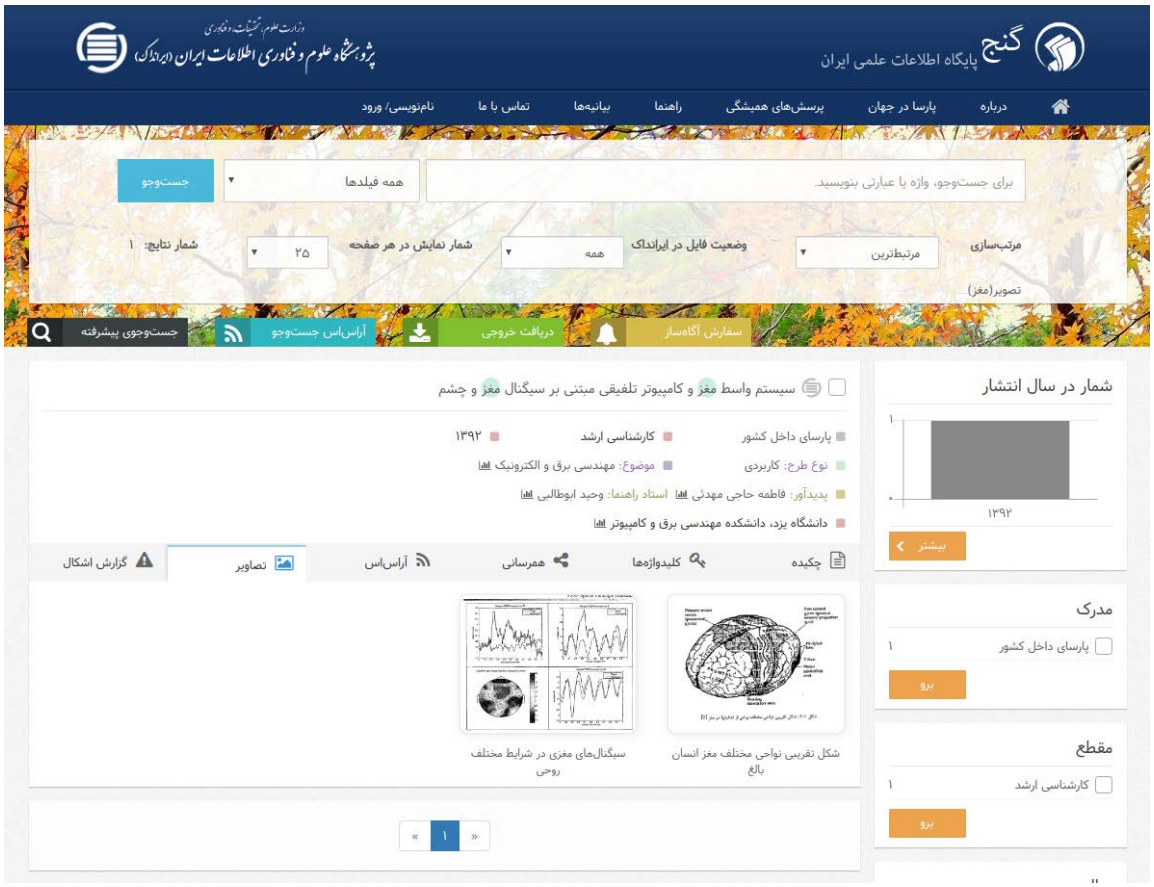
end



شکل ۳-۹- جست‌وجوی پیشرفته برای پایگاه گنج

پس از جست‌وجو در بخش جست‌وجوی پیشرفته، نتایج بازیابی شده همانند شکل ۳-۱۰ دراختیار کاربر قرار خواهد گرفت. برای هر رکورد بازیابی شده، بخشی با عنوان تصاویر افزوده شده است که تصاویر مربوط به آن سند نمایش داده می‌شود. لازم به ذکر است که برای کم شدن بارگزاری این صفحه و بازیابی سریعتر نتایج جست‌وجو با تصاویر، پس از کلیک بر روی این قسمت، درخواستی مجزا برای هر رکورد به سرور ارسال و سپس تصاویر بازیابی می‌شود. با استفاده از این روش (lazy loading) نتایج جست‌وجو باکمترین سربار[1] اضافی به کاربر نمایش داده خواهد شد. همچنین، در صفحه مربوط به هر رکورد در پایگاه گنج نیز بخشی با عنوان تصاویر افزوده شده است. در این بخش نیز، تصاویر استخراج شده از هر سند به همراه زیرنویس آن‌ها به کاربر نمایش داده می‌شود. (شکل ۳-۱۱)

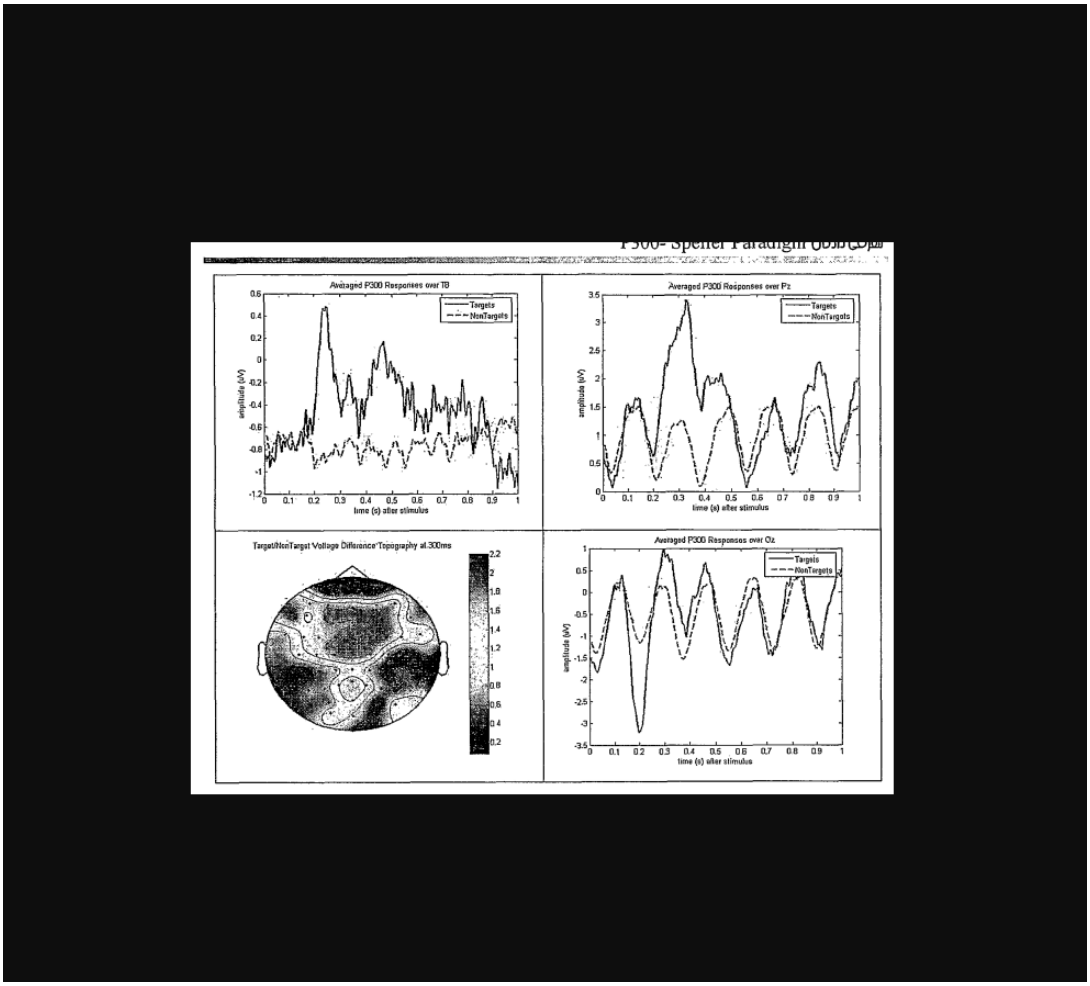
ایجاد پایگاه داده از تصاویر موجود در پایگاه اطلاعات علمی ایران (گنج) بر اساس یک روش هوشمند ۴۰



شکل ۳-۱۰- بازیابی نتایج جست‌جو در تصاویر

شکل ۳-۱۱-نمایش تصاویر هر رکورد در صفحه اختصاصی به هر سند

با کلیک بر روی هر تصویر امکانی طراحی شده است که تصویر در ابعاد اصلی به کاربر نمایش داده شود (شکل ۳-۱۲). لازم به اشاره است، در تمامی صفحات پیشین، برای لود سریع تصاویر، از تصویر کوچک (thumbnail) که در هنگام استخراج از پارساها تولید شده‌است، استفاده می‌شود و پس از کلیک بر روی تصویر موردنظر کاربر، تصویر در ابعاد اصلی نمایش داده خواهد شد.



شکل ۳-۱۲- نمایش تصویر در ابعاد اصلی

۳-۵- نتیجه گیری

در این فصل سیستمی برای ایجاد پایگاه داده تصاویر از فایلهای ورد و در نهایت استفاده از آن در پایگاه اطلاعاتی گنج معرفی شد. دز ایتدا فایل باید تجزیه شود و تصاویر همراه با زیرنویس آنها استخراج شود. از آنجاییکه در ادبیات از فایل پی دی اف استفاده شده است روشهای استخراج تصاویر از فایل پی دی اف هم بررسی شد. به دلیل چالشهای فایل پی دی اف ، از جمله پیروی نکردن آنها ازیک الگوی خاص و تولید تصاویر نامطلوب زیاد در نهایت الگوریتم پیشنهاد شده برای استخراج تصاویر در مورد فایلهای ورد به کار گرفته‌شد. بعد از آنکه تصاویر و اطلاعات مربوطه از فایل استخراج شد، مطابق با مدل داده‌ای که توسط موتور جستجوگر قابل استفاده است ذخیره سازی شد. برای جست‌وجو در تصاویر استخراج شده از اسناد، بخش جست‌وجوی پیشرفته پایگاه با استفاده از

زبان برنامه نویسی ruby و چارچوب توسعه نرم‌افزار Ruby on Rails توسعه داده شد. در فصل های بعد خروجی سیستم و نتایج بررسی می شود.

فصل چهارم

پیاده سازی روش پیشنهادی و ارزیابی آن

۱-٤- مقدمه

برای بررسی کارایی روش پیشنهادی در استخراج تصاویر و توضیحات آن‌ها از یک مطالعه موردی در پایگاه اطلاعات علمی ایران (گنج) کمک گرفتیم. این پایگاه مرجع اصلی دسترسی به تمام متن پایان‌نامه‌ها و رساله‌های تحصیلات تکمیلی در ایران می‌باشد. به صورت تصادفی ۱۵۰ سند فنی مهندسی با فرمت پی.دی.اف و ورد را از پایگاه داده گنج انتخاب کردیم که اکثر این اسناد پایان‌نامه‌های ارشد و دکتری هستند. در ادامه این اسناد توسط روش پیشنهادی مورد تجزیه و تحلیل قرار گرفتند. روش پیشنهادی ما بر اساس تجزیه و تحلیل فایل ورد است. در این فصل، استخراج تصاویر از فایل پی.دی.اف هم که در ادبیات به آن پرداخته شده است، بررسی می‌شود و نقاط ضعف آن بحث خواهد شد.

۲-٤- پیاده سازی روش پیشنهادی در مورد فایل ورد

همانطور که قبلا اشاره شد، فایل ورد یک فایل فشرده شده است و اگر آن غیرفشرده کنیم مجموعه‌ایی از فایل‌ها را به دست می‌اوریم که مهمترین آن فایل document.xml است. در روش پیشنهادی فایل XML تجزیه و تحلیل شد. برای پیاده‌سازی روش پیشنهادی از زبان برنامه‌نویسی پایتون[1] و کتابخانه lxml کمک گرفته شد. این کتابخانه فایل XML را به صورت درختی از عنصرها بارگذاری می‌کند. عنصر[2]، شیء اصلی در این کتابخانه است که به طور اختصاصی برای ذخیره‌سازی داده‌های سلسله‌مراتبی طراحی شده است. در حقیقت عنصر، حد واسط بین نوع داده‌ای لیست و دیکشنری در زبان پایتون است. فایل بارگذاری شده توسط درخت عنصرها قابلیت جستجو و

[1] Python
[2] element

استخراج داده با دستورالعمل‌های پایتون را پیدا می‌کند. تمام اطلاعات مربوط به طرح‌بندی تصویر در عنصر <drawing> ذخیره می‌شود. بدین ترتیب شناسه هر تصویر با استفاده از Xpath زیر قابل بازیابی است.

w:drawing/wp:inline/a:graphic/a:graphicData/pic:pic/pic:blipFill/a:blip/@r:embed

شناسه هر تصویر مشخص می‌کند که اطلاعات ذخیره شده در فایل XML به کدام تصویر استخراج شده در پوشه مدیا مربوط می‌شود. برای پیدا کردن زیرنویس مربوط به هر تصویر باید پاراگراف‌ها و جداول بعد از دستور جایگذاری تصویر را بررسی کرد. در مورد فایل‌هایی که در نگارش از قابلیت درج زیرنویس ورد استفاده شده است، اطلاعات مربوط به زیرنویس در عنصر <caption> ذخیره می‌شود. تصاویر و زیرنویس مربوط به آن‌ها مطابق الگوریتم ارائه شده در فصل چهارم استخراج می‌شوند.

۳-٤- پیاده سازی روش استخراج تصاویر از فایل پی.دی.اف

همان طور که پیش‌تر گفته شد، روش ایجاد پایگاه اطلاعات تصاویر در ادبیات در مورد فایل‌های پی.دی.اف است. در فایل پی.دی.اف اطلاعات در زیر شیء صفحه ذخیره می‌شود. تمام اطلاعات مهم برای ایجاد یک صفحه پی.دی.اف در این شیء آورده شده است پی.دی.اف معمولاً تصاویر ماتریسی را در یک شیء جداگانه به نام Xobject به صورت داده دودویی ذخیره می‌کند. برای استخراج تصویر باید همه اطلاعات تصویر را که در Xobject ذخیره شده، استخراج کرد و با استفاده از آن تصویر را ایجاد نمود.

بازیابی تصاویر برداری بسیار پیچیده‌تر از تصاویر ماتریسی است. تصاویر برداری در قالب مسیر می‌آید که در واقع رشته‌ای از کاراکترها هستند. به عنوان مثال در مسیر زیر دستور ایجاد یک خط در فایل پی.دی.اف آمده است.

175 720 m 175 50 m l h S

اعداد قبل از m نقطه شروع مسیر را نشان می‌دهد، l تعیین می‌کند که نوع مسیر خط راست است و h نشان می‌دهد که 175 نقطه انتهای مسیر است. S دستور کشیدن خط است. یک تصویر از مجموعه‌ای

از مسیرهای مشابه ساخته می‌شود. گاهی اوقات متن و جدول موجود در پی.دی.اف هم با مسیر مشابه تصاویر برداری ایجاد می‌شوند. تشخیص درست مسیرهای مربوط به یک تصویر بسیار چالش‌برانگیز است.

برای پردازش فایل‌های پی.دی.اف آزمایشی از کتابخانه PDFBox در جاوا استفاده کردیم. PDFBox کتابخانه‌ای قدرتمند است که در مقالات پژوهشی مختلفی برای تجزیه فایل پی.دی.اف مورد استفاده قرار گرفته است (Choudhury et al 2013; Choudhury and Giles 2015; Clark and Divvala 2016; Siegel et al. 2016). کلاس getXObject شیء XObject را استخراج می‌کند که شامل اطلاعات مربوط به تصویر می‌شود. کلاس PDImageXObject با استفاده از اطلاعات XObject تصویر مورد نظر را ایجاد می‌کند. PDFBox تصاویر برداری که با مسیر مشخص می‌شوند را استخراج نمی‌کند. در بعضی موارد، فلوچارت‌ها، جدول‌ها و فرمول‌ها از ترکیب چند XObject کوچک تشکیل می‌شوند. مشخصات هر خط و هر علامتی که در جدول وجود دارد، در یک XObject ذخیره شده و به صورت مجزا در خروجی نمایش داده می‌شود. این امر منجر به استخراج ده‌ها هزار تصویر نویز می‌گردد. خیلی از این تصاویر نویز، سایز کوچکی (حدود چند ده پیکسل) دارند. به همین دلیل تنها تصویری که ارتفاع و عرض آن‌ها از ۵۰ پیکسل بیشتر باشند در این آزمایش مدنظر قرار گرفته‌اند.

متن در پی.دی.اف با استفاده از عملگرهای متفاوت از تصویر و به صورت مستقل ایجاد می‌شود. جهت استخراج زیرنویس در فایل‌هایی که متن آن‌ها تصاویر برداری نیستند، ابتدا جملات شامل کلمات کلیدی مثل «شکل» و «نمودار» استخراج می‌شوند. سپس جملات نامربوط بر اساس یک بردار ویژگی‌های از پیش تعریف شده حذف می‌گردند. لازم به ذکر است که متن موجود در فایل‌های آزمایشی را با استفاده از کلاس getTex استخراج کردیم

٤-٤- تجزیه و تحلیل نتایج

با بررسی فایل‌های ورد دیده می‌شود که از ۱۵۰ فایل، ۱۳ سند از نظر محتوایی ناقص بودند. ۱۳۷فایل سند باقی‌مانده، با برنامه توسعه داده شده مورد تجزیه و تحلیل قرار گرفت

فایل ورد

. از ۱۳۷ فایل آزمایشی در ۵۵ سند از قابلیت درج زیرنویس استفاده شده بود.تصاویر این ۵۵ فایل ورد توسط الگوریتم معرفی شده بازیابی شد. خروجی الگوریتم مطابق جدول ۳-۱ در فصل سوم ذخیره سازی شد. نمونه ایی از این خروجی در جدول ۵-۱ آماده است.

از این ۵۵ سند ده سند را به صورت تصادفی انتخاب کردیم. کل تصاویر موجود در این ده فایل ۴۲۸ تصویر و تعداد کل تصاویر استخراج شده توسط الگوریتم ارائه شده ۳۲۵ بود. از این ۳۲۵ تصویر ۳۴ تصویر تصاویر نامطلوب بودند. این تصاویرنامطلوب شمامل قسمتی از تصویر موجود در فایل بودند و به طور کامل نمایش داده نشده بودند. به این ترتیب دقت و بازخوانی به صورت زیر محاسبه می شوند.

$$\text{Precision}=\frac{291}{325}=89\%$$

$$\text{Recall}=\frac{291}{428}=67\%$$

خلاصه نتایج الگوریتم توسعه داده شده در مورد فایل‌های ورد به صورت زیر است:

- تصاویر جایگزاری شده از تمام فایل‌هایی که از فرم استاندارد ورد را دارند، و از قابلیت درج زیرنویس ورد استفاده شده است بدون خطا استخراج می‌شود.
- تصاویر از فایل‌هایی که از درج زیرنویس استفاده نشده است، استخراج نمی‌شود
- تصاویری که با نرم افزار مایکروسافت ورد ایجاد شده اند، و تصاویر جایگزاری شده نیستند، عموما فلوچارتها و جدولها، استخراج نمی‌شوند
- بعضی از فایلها نیمه استاندار هستند. به این معنا که وقتی از Insert Table of Figures استفاده می کنیم، لیست تصاویر نشان داده می‌شود. اما در فایل XML زیرنویس این تصاویر با تگ caption مشخص نشده است. . تصاویر از این فایلها استخراج نمی‌شود.

فایل پی دی اف

در مورد فایل های پی.دی.اف . از ۱۳۷ فایل پی.دی.اف در داده آزمایشی، در ۵۹ فایل متن آن‌ها در واقع تصویر برداری هستند و نمی‌توان آن‌ها را استخراج کرد.در مورد این فایلها تنها راه استخراج متن

استفاده از روشهای نویسه خوانی نوری[1] است. در مورد بقیه فایلها، جملات استخراج شده از فارسی ترکیب چپ به راست دارند و آرایش جملات فارسی خروجی، درجایی که با کلمات انگلیسی، علائم و یا فرمول ترکیب شده‌اند به هم می‌ریزد.

از فایل‌های باقی‌مانده تقریبا در نیمی‌از آن‌ها به خاطر فرمت خاص پی.دی.اف استفاده شده صدها تصویر نویز ایجاد می‌شود که به دلیل شباهت ساختاری زیاد آن‌ها با تصاویر هدف، فیلتر کردن آن‌ها بسیار چالش‌برانگیز است. بعضی از این تصاویر نویز به شرح زیر است:

- لگو ها و بسم الله ارحمن الرحیم به صورت تصاویر جداگانه استخراج می‌شوند و باید حذف گردند (شکل ۴-۱).

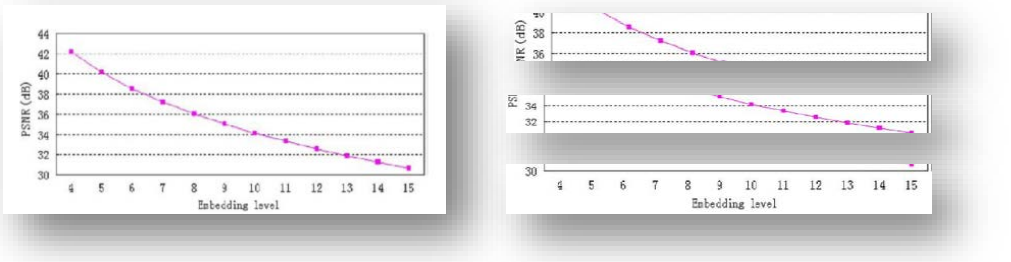
- فرمولها در بعضی موارد به صورت تصویر جایگزاری شده هستند و استخراج می‌شوند.(شکل ۴-۱)
- گاهی یک تصویر واحد ممکن است در تعدادی XObject ذخیره شودو در نتیجه در خروجی به صورت تصاویر مجزا دیده شوند(شکل ۴-۲).

برای فایل پی.دی.اف ما نیازمند هستیم که از روشهای یادگیری ماشین ، پردازش زبان طبیعی و قانون محور برای تشخیص زیرنویس، استخراج تصاویر معنی دار، تخصیص زیرنویس به تصویر مربوطه استفاده کنیم. تنوع نوشتاری در اسناد موجود در گنج از کاربست پذیری این روشها می‌کاهد.

$$C(t) = \sum_k c_k \Phi_k(t) + \sum_j \sum_k d_{jk} \Psi_{jk}(t)$$

$$S(t) = \sum_k s_k \Phi_k(t) + \sum_j \sum_k r_{jk} \Psi_{jk}(t)$$

شکل٤- ١- نمونه هایی ازتصاویر نویز استخراج شده توسط نرم افزار تجزیه فایل



شکل ٤-۲-سمت چپ نمودار موجود در فایل پی دی اف و سمت راست خروجی حاصل از نرم افزار تجزیه فایل

جدول ٤-١- نمونه ایی از پایگاه داده تصاویر مستخرج از فایل ورد مطابق جدول ٣-١

id	article_id	uid	title_fa	size	height	width	created_a	updated_at
1	w189150	4e08a2dcba4e46	چگالي طيف توان سيگنال	27.4	406	727	2019-07-1	2019-07-13 12:10:35
2	w189150	0e2a8beefeb34e	سيگنال مربعي با فركانس 0.01	31.5	420	560	2019-07-1	2019-07-13 12:10:35
3	w189150	029524f0ed8247	سيگنال مربعي با فركانس 0.1	47.5	420	560	2019-07-1	2019-07-13 12:10:35
4	w189150	10855679db8641	چگالي طيف توان سيگنال با فركانس 0.01	15.9	420	560	2019-07-1	2019-07-13 12:10:35
5	w189150	1593b10213e248	چگالي طيف توان سيگنال با فركانس 0.1	14.8	420	560	2019-07-1	2019-07-13 12:10:35
6	w189150	8ed4ee7295db4	چگالي طيف توان (فركانس سيگنال در نمودار آبي 0.1 و در نه	12.5	420	560	2019-07-1	2019-07-13 12:10:35
7	w189150	b084ddeb84e44	سيگنال اصلي	18.5	420	560	2019-07-1	2019-07-13 12:10:35
8	w189150	087d9a3dd7a940	سيگنال اصلي در رشته شبه تصادفي	22.1	420	561	2019-07-1	2019-07-13 12:10:35
9	w189150	c4db1b69ebaa49	سيگنال مدوله شده(آبي) ،سيگنال مدوله شده و عبور كرده از كا	41.8	420	560	2019-07-1	2019-07-13 12:10:35
10	w189150	fbe61b333e884c	نرخ خطاي بيت براي رشته با طول 10	15	420	561	2019-07-1	2019-07-13 12:10:35
11	w189150	69faf1d28c7a4a	نرخ خطاي بيت براي رشته با طول 32	13.7	420	560	2019-07-1	2019-07-13 12:10:35
12	w189150	234da3a8b07b44	نرخ خطاي بيت براي رشته هاي با طول مختلف	33.7	420	560	2019-07-1	2019-07-13 12:10:35
13	w189150	90a5126c163547	فرستنده سيستم با كد مرجع انتقال داده شده	28.6	378	596	2019-07-1	2019-07-13 12:10:35
14	w189150	20002a0c6d0341	گيرنده سيستم با كد مرجع انتقال داده شده	28.9	278	948	2019-07-1	2019-07-13 12:10:35
15	w189150	940198da6c0d4c	سيگنال حاصل جمع دو سيگنال 15 و 7 كيلو هرتز	35.4	420	560	2019-07-1	2019-07-13 12:10:35
16	w189150	33dd465e310b4f	سيگنال اصلي (آبي) و آشكار شده (قرمز) در فركانس 7 كيلو ه	38.3	420	560	2019-07-1	2019-07-13 12:10:35
17	w189150	a2ae0196da3843	نرخ خطاي بيت براي روش ارسال كد مرجع	14.9	420	560	2019-07-1	2019-07-13 12:10:35
18	w189150	7ab7abe372bd4a	گيرنده هم بستگي طيف گسترده	13.4	203	473	2019-07-1	2019-07-13 12:10:35
19	w189150	b855285347d648	گيرنده ويولت طيف گسترده	15	222	531	2019-07-1	2019-07-13 12:10:35
20	w189150	c3231ac799aa4c	ضرايب ويولت	21.1	420	560	2019-07-1	2019-07-13 12:10:35
21	w189150	1343706519a947	نرخ خطاي بيت روش ويولت	13.7	420	560	2019-07-1	2019-07-13 12:10:35
22	w189150	bfc6d30a101841	ديا‌گرام PFDM	4.5	214	539	2019-07-1	2019-07-13 12:10:35
23	w189150	53e59e2c60c242	با ورودي سيگنال PFDM	9.7	214	539	2019-07-1	2019-07-13 12:10:35
24	w189150	547a107250114a	ساختار Modac	82.4	558	1264	2019-07-1	2019-07-13 12:10:35
25	w189150	d02b5ac58c384b	نرخ خطاي بيت سه‌روش هم بستگي(آبي) و ويولت(صورتي)	18.9	420	560	2019-07-1	2019-07-13 12:10:35
26	w189150	ffd958d5038347	نرخ خطاي بيت دو روش هم بستگي(شكل آبي) و ويولت(شكل	16.4	420	560	2019-07-1	2019-07-13 12:10:35
27	w189643	4a48a33d15c648	توزيع تنش روي سطح براده	2083.7	601	887	2019-07-1	2019-07-13 12:10:36
28	w192302	d2c65220cb3f4d	نمونهای از بازوی مکانیکی ماهر متحرک [4]	106.3	481	285	2019-07-1	2019-07-13 12:10:39
29	w192302	6a1a650aa6c047	پایه متحرک تفاضلی به همراه چرخ هرز گرد	31.3	385	502	2019-07-1	2019-07-13 12:10:39
30	w192302	1023e120ebb841	پایه متحرک چهارچرخ	103.2	154	171	2019-07-1	2019-07-13 12:10:39
31	w192302	e787496d98594c	بازوی ماهر دولینکی متحرک غیر هولونومیک [9]	350	245	363	2019-07-1	2019-07-13 12:10:39
32	w192302	34d39c38936c49	جواب های چندگانه با هدف واحد	198.9	186	271	2019-07-1	2019-07-13 12:10:39
33	w192302	bc88d45fe4604b	شماتیک انتخاب کلونال [69]	838.8	401	535	2019-07-1	2019-07-13 12:10:39
34	w192302	a38c357667a848	فلوچارت الگوریتم انتخاب منفی	93.7	351	1016	2019-07-1	2019-07-13 12:10:39
35	w192302	75107ee6e7fa4c	بازوی مکانیکی دو لینکی ثابت	19.4	190	275	2019-07-1	2019-07-13 12:10:39
36	w192302	49bc8d25703245	مینیمم محلی و کلی	27.5	588	787	2019-07-1	2019-07-13 12:10:39
37	w192302	d4cef0ccc667486	مثال هایی از مسیرهای ممکن [104]	961.5	389	631	2019-07-1	2019-07-13 12:10:39
38	w192302	f6f1b31ffc6e44b	جابجایی، سرعت و شتاب مفصل اول	70.2	553	632	2019-07-1	2019-07-13 12:10:39
39	w192302	a0fabb73344146	جابجایی، سرعت و شتاب مفصل دوم	67.9	553	632	2019-07-1	2019-07-13 12:10:39

جمع‌بندی و نتیجه‌گیری

جمع بندی

در طرح حاضر روش‌های استخراج تصویر و زیرنویس جهت ایجاد پایگاه داده تصاویر از اسناد علمی بررسی شد. در این راستا دو فرمت رایج ورد و پی.دی.اف مورد مطالعه و مقایسه قرار گرفتند. سپس با توجه به نوع نیازمندی‌های اسناد علمی منتشر شده به زبان فارسی، روشی ساختار محور و مبتنی بر پردازش فایل ورد برای ایجاد پایگاه داده تصاویر پیشنهاد شد. این روش قادر است تصاویر ۴۰ درصد از فایلهارا با دقت ۸۹ درصد استخراج کند.

استخراج تصاویر از فایل پی.دی.اف هم بررسی شد که به دلیل چالش‌هایی که فایل پی.دی.اف دارد استفاده از فایل پی.دی.اف پیشنهاد نمی‌شود. چالشهای فایل پی.دی.اف به طور خلاصه به شرح زیر است.

- در حدود یک سوم از فایلها متن موجود در واقع تصویربرداری بود. در مورد این فایلها برای استخراج زیرنویس ناگزیر از استفاده از روشهای نویسه خوانه نوری هستیم. روشها و نرم افزارهای موجود در مورد متون فارسی عملکرد خوبی ندارند.
- خطوط زیرنویس در فایل پی.دی.اف با برچسب خاصی مشخص نمی‌شود برای تشخیص زیرنویس از بقیه خطوط باید از روشهای یادگیری ماشین و قانون محور استفاده کرد. برای این منظور، باید بتوانیم ویژگی‌هایی را استخراج کنیم که در تمام اسناد موجود در گنج متمایز کننده زیرنویس از سایر خطوط باشد. پایگاه اطلاع گنج استاندارد خاصی ندارد و فایلهای موجود تنوع نوشتاری زیادی دارند. روشهای موجود در ادبیات هم دامنه تحقیق خود را روی اسناد از یک گروه از مجلات خاص با ویژگی های مشخص متمرکز کرده اند و در این جا کاربرد ندارند.
- استخراج تصاویر برداری به دلیل شباهت ساختاری آنها با اجزا دیگر موجود در فایل پی.دی.اف مثل جداول و متن بسیار پیچیده است.

- تصاویر نویز استخراج شده بسیار زیاد است. لوگوها و فرمولها و قسمتی از جدولها استخراج می‌شوند. همچنین بعضی از تصاویر ماتریسی هم تجزیه شده و به صورت قطعات مجزا در خروجی دیده می‌شوند.
- متن فارسی ساختار راست به چپ دارد وآرایش جملات استخراج شده فارسی، از اکثرفایل های پی دی اف به هم می ریزد.

چنانجه بر تمام مشکلات خروج زیرنویس و تصاویر فائق بیاییم مرحله بعد اختصاص درست هر تصویر به زیرنویس مربوطه که خود آن هم چالش مربوط به خودش را دارد است.

کارهای آینده

الگوریتم پیشنهادی می‌تواند بهبود پیدا کند. برخی ازفایلها اگر چه فرمت استاندارد ندارند، اما زیرنویس آنها به شکل شی مجزا در نرم افزار مایکروسافت ورد قابل تشخیص هستند. در مورد این فایلها می‌توان یک نمونه آماری معتبر جمع آوری کرد و چنانچه بین همه آنها از نظر ساختاری شباهت وجود داشته باشد یک الگوریتم قانون محور برای استخراج زیرنویس آنها طراحی کرد. چنانچه در سامانه گنج همراه با فایل سند، لیست زیرنویس تصاویر به صورت اجباری بارگزاری شود، شاید بتوان از این لیست می‌توان برای استخراج بهتر تصاویر بهره برد. در این صورت با روشهای پردازش زبانهای طبیعی و قانون محور موقعیت زیرنویس را در فایل پیدا کرده و ناحیه گرافیکی نزدیک به آن را به آن اختصاص دهیم.

جهت بهبود عملکرد جستجو برای هر تصویر یک جدول برچسب می‌تواند ایجاد شود. برای ایجاد این جدول برچسب می‌توان تمام جملاتی که در آن به تصویر اشاره شده است را استخراج کرد و با استفاده از روشهای پردازش زبانهای طبیعی کلمات کلیدی را استخراج کرد. همچنین می‌توان با استفاده از روشهای پردازش تصویر و نویسه خوانی متن داخل نصویر هم استخراج کرد و به عنوان برچسب در نظر گرفت. کلیه تصاویر استخراج شده می‌تواند به یکی از گروههای نمودار خطی، فلوچارت، تصاویر طبیعی، جدول و غیره تقسیم بندی شود و گروه تصویر هم به عنوان یک برچسب در نظر گرفته شود . برای اینکار می‌توان از روشهای یادگیری عمیق استفاده کرد.

مراجع

علایی ابوذر، الهام. ۱۳۹۷. معرفی رویکردی ماشینی با استفاده از الگوریتم لسک و برچسب‌دهی نحوی جهت رفع ابهام از معنای کلمات. *پژوهشنامه پردازش و مدیریت اطلاعات* ۳۳ (۳): ۱۱۶۵-۱۱۸۲.

Bloomberg, Dan S. 1991. *Multiresolution morphological approach to document image analysis*. In Proceedings of the International Conference on Document Analysis and Recognition, pp. 963-971.

Chan, J., C. Ziftci, and D. Forsyth. 2006. *Searching off-line Arabic documents*. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1455-1462.

Choudhury, S. R., and C. L. Giles. 2015. *An architecture for information extraction from figures in digital libraries*. In Proceedings of the 24th International Conference on World Wide Web Companion, pp. 667-672.

Choudhury, S. R., P. Mitra, A. Kirk, S. Szep, D. Pellegrino, S. Jones, and C. L. Giles. 2013. *Figure metadata extraction from digital documents*. In Proceedings of the 12th International Conference on Document Analysis and Recognition, pp. 135-139.

Choudhury, S. R., P. Mitra, and C. L. Giles. 2015. *Automatic extraction of figures from scholarly documents*. In Proceedings of the 2015 ACM Symposium on Document Engineering, pp. 47-50.

Clark, C., and S. Divvala. 2016. *PDFFigures 2.0: Mining figures from research papers*. In Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL), pp. 143-152.

Cohen, R., A. Asi, K. Kedem, J. El-Sana, and I. Dinstein. 2013. *Robust text and drawing segmentation algorithm for historical documents*. In Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing, pp. 110-117.

Khabsa, M., and C. L. Giles. 2014. *The number of scholarly documents on the public web*. PLoS ONE, vol 9, no. 5, pp.e93949.

Kim, D and Yu, H., 2011. *Figure Text Extraction in Biomedical Literature*. PLoS ONE, vol. 6, no. 1, pp. e15 338+.

Kumar, S., R. Gupta, N. Khanna, S. Chaudhury, and S. D. Joshi. 2007. *Text extraction and document image segmentation using matched Wavelets and MRF model*. IEEE Transactions on Image Processing 16 (8): 2117-2128.

Lemaitre, A., J. Camillerapp, and B. Coüasnon. 2008. *Multiresolution cooperation makes easier document structure recognition*. International Journal of Document Analysis and Recognition 11 (2): 97-109.

Li, Z., M. Stagitis, S. Carberry, and K. F. McCoy. 2013. *Towards retrieving relevant information graphics*. In Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 789-792.

Liu, Y., K. Bai, P. Mitra, and C. L. Giles. 2007. *TableSeer: Automatic table metadata extraction and searching in digital libraries*. In Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL), pp. 91-100.

Lopez, L., J. Yu, C. N. Arighi, H. Huang, H. Shatkay, and C. Wu. 2011. *An automatic system for extracting figures and captions in biomedical PDF documents*. In Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 578-581.

Nagy, G., and S. C. Seth. 1984. *Hierarchical representation of optically scanned documents*. In Proceedings of the 7th International Conference on Pattern Recognition (ICPR), pp. 347-349.

Praczyk, P. A., and J. Nogueras-Iso. 2013. *Automatic extraction of figures from scientific publications in high-energy physics*. Information Technology and Libraries 32 (4): 25-52.

Printing Conection, 1983, https://www.printcnx.com/

Rehman, A., and T. Saba. 2014. *Neural networks for document image preprocessing: state of the art.* Artificial Intelligence Review 42 (2): 253-273.

Savva, M., N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. 2011. *ReVision: automated classification, analysis and redesign of chart images*. In Proceedings of the 24th annual ACM symposium on User Interface Software and Technology (UIST), pp. 393-402.

Siegel, N., Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi. 2016. *FigureSeer: Parsing result-figures in research papers.* In In Proceedings of *Computer Vision ECCV 2016.*

Siegel N., Lourie N, Power R, and W. Ammar 2018. Extracting Scientific Figures with Distantly Supervised Neural Networks. In Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries (JCDL '18). ACM, New York, NY, USA, 223-232. DOI: https://doi.org/10.1145/3197026.3197040

Srihari, S. N. 1986. *Document image understanding*. In proceedings of the ACM Fall Joint Computer Conference, pp. 87-96.

Tsutsui S. and David J. Crandall. 2017. *A Data Driven Approach for Compound Figure Separation Using Convolutional Neural Networks*. In proceedings of ICDAR 2017.

Williams, K., L. Li, M. Khabsa, J. Wu, P. C. Shih, and C. L. Giles. 2014. *A Web Service for Scholarly Big Data Information Extraction.* In Proceedings of the IEEE International Conference on Web Services, pp. 105-112.

Wu, J., K. M. Williams, H. H. Chen, M. Khabsa, C. Caragea, S. Tuarob, A. G. Ororbia, D. Jordan, P. Mitra, and C. L. Giles. 2015. *CiteSeerX: AI in a digital library search engine*. *AI Magazine* 36 (3): 35-48.

Xu, S., J. McCusker, and M. Krauthammer. 2008. Yale Image Finder (YIF): *a new search engine for retrieving biomedical images. Bioinformatics* 24 (17): 1968-1970**.**

Yu Y., Lin H., Meng J., Wei X., and Zhao Z.. 2017. *Assembling Deep Neural Networks for Medical Compound Figure Detection*. Information 8 (2017), 48.

Yang X., Yumer E., Asente P., Kraley M., Kifer D., and Giles C. Lee. 2017. *Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Networks*. In proceeding of CVPR 2017

Wikipedia contributors. Wikipedia, The Free Encyclopedia, Vector graphics. https://en.wikipedia.org/w/index.php?title=Vector_graphics&oldid=904405531 Date of last revision: 1 July 2019 23:07 UTC

Wikipedia contributors. Wikipedia, The Free Encyclopedia, Raster graphics. https://en.wikipedia.org/w/index.php?title=Vector_graphics&oldid=904405531 Date of last revision: 19 April 2019 00:08 UTC

Futrelle, R. P., M. Shao, C. Cieslik, and A. E. Grimes. 2003. *Extraction, layout analysis and classification of diagrams in PDF documents*. In Proceedings of the 7th International Conference on Document Analysis and Recognition, pp. 1007-1013.

پیوست

پیوست الف

سند استقرار سامانه

برای توسعه این ماژول که در نهایت در پایگاه گنج نصب خواهد شد از زبان برنامه نویسی Ruby و چارچوب Rails برای توسعه آن استفاده شده است. این ماژول قابلیت نصب و راه‌اندازی بر روی هرکدام از توزیع‌های لینوکس را داراست. در ادامه نحوه نصب این سامانه و به همراه تمامی وابستگی‌های آن بر روی توزیع دبیان از لینوکس شرح داده شده است.

نصب Ruby

برای نصب روبی در ابتدا باید dependency های مورد نیاز برای این زبان برنامه نویسی را نصب کنید. به همین منظور دستورات زیر را در ترمینال وارد کنید:

```
sudo apt-get update
```

```
sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev python-software-properties libffi-dev
```

پس از نصب پیش‌نیازها، برای نصب روبی باید مراحل زیر به ترتیب انجام شود:

```
git clone git://github.com/sstephenson/rbenv.git .rbenv

echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc

echo 'eval "$(rbenv init -)"' >> ~/.bashrc

exec $SHELL

git clone git://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby-build

echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc

exec $SHELL

git clone https://github.com/sstephenson/rbenv-gem-rehash.git ~/.rbenv/plugins/rbenv-gem-rehash

rbenv install 2.2.2

rbenv global 2.2.2
```

برای اینکه پس نصب هر بسته نرم‌افزاری gem نیاز نباشد راهنماها و مستندات آن‌ها را نیز بر روی سرور نصب کرد، می‌توان دستورات زیر را اجرا کنید:

```
echo "gem: --no-ri --no-rdoc" > ~/.gemrc

gem install bundler
```

نصب چارچوب Ruby on Rails

پس از نصب روبی ، برای نصب Rails در ابتدا نیاز به نصب NodeJS است. دستورات زیر، در ابتدا repository مورد نیاز را به سرور افزوده و سپس اقدام به نصب NodeJS می‌کند:

```
sudo add-apt-repository ppa:chris-lea/node.js

sudo apt-get update

sudo apt-get install nodejs
```

و درنهایت با دستور زیر چارچوب Rails را نصب می‌کنیم:

```
gem install rails -v 4.2.1

rbenv rehash

rails -v
```

نصب پایگاه داده MySQL

برای نصب پایگاه داده MySQL می‌توان از دستورات زیر بهره‌برد:

```
sudo apt-get install mysql-server

sudo apt-get install mysql-client

sudo apt-get install libmysqlclient-dev
```

نصب وب سرور Apache و Phusion Passenger

برای deploy کردن یک برنامه تحت چارچوب Rails علاوه بر وب سرور Apache (و یا Nginix) نیاز به ماژول‌های Phusion Passenger نیز وجود دارد تا برنامه کاربردی با کارایی بالا توسط این وب سرور serve شود. برای نصب آپاچی از دستور زیر استفاده کنید.

```
sudo apt-get install apache2
```

پس از نصب آپاچی، نوبت به نصب phusion Passenger می‌رسد. در ابتدا، برای دانلود و نصب phusion passenger باید pgp key شرکت میزبان این نرم افزار را به سرور افزود. برای این کار باید دستور زیر را در ترمینال وارد کنید:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 561F9B9CAC40B2F7
```

در حال حاضر Phusion passanger از طریق پروتکل https قابل دانلود می‌باشد. با استفاده از دستور زیر می‌توان این اطمینان را حاصل کرد که نرم‌افزارهای مورد نیاز برای پشتیبانی از این پروتکل در دسترس است.

```
sudo apt-get install apt-transport-https ca-certificates
```

در مرحله بعد باید repositoryهای مورد نیاز برای دانلود phusion passenger را به سرور اضافه کرد. لیست repositoryها معمولا در فایل‌های موجود در این مسیر /etc/apt/sources.list.d/ قرار گرفته‌اند.

```
##### Only add ONE of these lines needed
# Ubuntu 15.04
deb https://oss-binaries.phusionpassenger.com/apt/passenger vivid main
# Ubuntu 14.04
deb https://oss-binaries.phusionpassenger.com/apt/passenger trusty main
```

در نهایت برای نصب phusion passenger بر روی سرور دستورات زیر را وارد کنید.

```
sudo apt-get update

sudo apt-get install libapache2-mod-passenger

sudo a2enmod passenger

sudo apache2ctl stop

sudo apache2ctl start
```

راه‌اندازی وبسایت

پس از نصب تمامی موارد اشاره شده در بالا، پیشنهاد می‌شود با استفاده از یک virtual host برنامه را deploy کنید. به همین منظور می‌توان در یک فایل config در مسیری همانند /etc/apache2/sites-available/education.conf با وارد کردن اطلاعات زیر (با درنظر گرفتن آنکه تمامی فایل‌های برنامه در مسیر /home/edu قرار دارند) وب سایت را راه‌اندازی کرد.

```
>VirtualHost *:80<
    ServerName edu.irandoc.ac.ir
    DocumentRoot /home/edu/public
>    Directory /home/edu/public<
       Allow from all
       Options -MultiViews
       Require all granted
        RailsEnv production # determine working environment [development]
/>   Directory<

>   Location /phpmyadmin<
      PassengerEnabled off
/>   Location<
/>VirtualHost<
```

در وب سایت توسعه داده شده کنونی از کتابخانه‌هایی در بخش‌های مختلف استفاده شده است. تمامی کتابخانه‌های بکارگرفته شده در این وب سایت در فایل Gemfile در root برنامه موجود است. برای نصب تمامی کتابخانه‌های مورد نیاز تنها کافی است در روت برنامه دستور bundle install را اجرا کنید. تمامی این کتابخانه‌ها از repository مربوط به rails دانلود و نصب خواهد شد. لیست مربوط به این کتابخانه‌ها در ادامه موجود است.

۱. **Recaptcha**: برای استفاده از captcha
۲. **premailer-rails** و **nokogiri** : برای رعایت فرمت استاندار در نمایش ایمیل‌ها
۳. **ruby-hmac**: برای تولید کدهای hash استفاده شده در ماژول پرداخت آنلاین
۴. **Ckeditor**: ویرایشگر استاندارد تولید متن
۵. **jquery-ui-rails**: از تابع datepicker این کتابخانه برای انتخاب تاریخ استفاده شده است
۶. **wice_grid**: برای نمایش لیست در صفحه‌هایی همانند دوره‌ها با قابلیت جست‌وجو و فیلتر کردن استفاده شده است.
۷. **Paperclip**: برای مدیریت آپلود
۸. **Pundit**: برای مدیریت سطوح دسترسی از این کتابخانه استفاده شده است.
۹. **Devise**: برای authentication از این کتابخانه استفاده شده است.
۱۰. **Prawn**: برای تولید گزارشات در فرمت pdf از این کتابخانه استفاده می‌شود.
۱۱. **Jalalidate**: از این کتابخانه برای نمایش تاریخ‌ها در فرمت شمسی استفاده می‌شود.

در مرحله آخر با اجرای دو دستور زیر به ترتیب تمامی پایگاه داده (شامل جداول و روابط بین آن‌ها) ساخته شده و دیتاهای مورد نیاز که باید برای شروع سامانه در جداول موجود باشند (مانند رشته‌های تحصیلی و لیست سازمان‌ها که در عضویت مورد نیاز می‌باشند)، مقداردهی خواهد شد.

```
rake db:migrate

rake db:seed
```

نگهداری سامانه

برای نگهداری و خطایابی این ماژول می‌توان با توجه به علائم و مشکل به وجود آمده از بخش‌های زیر به منظور حل مشکل استفاده کرد.

تغییر در فایل‌های برنامه

در صورت نیاز در تغییر در هر کدام از فایل‌های مربوط به برنامه، با توجه به حالت برنامه که در حالت production است، این نیاز وجود دارد که وب سرور یکبار ریستارت شده تا تغییرات در برنامه اعمال شود. به همین منظور برای ریستارت کردن وب سرور باید دستور زیر در ترمینال وارد شود:

```
service apache2 restart
```

در صورتی که فایل‌های مربوط به javascript و cssها تغییر داده شد (خصوصا در زمانیکه این تغییرات زیاد است)، توصیه می‌شود با اجرای دستور زیر این فایل‌ها را precompile کرد. با این روش از فایل‌های مربوطه min و نسخه کوچک آن‌ها توسط چارچوب rails ایجاد می‌شوند و به این ترتیب لود شدن این فایل‌ها با سرعت زیادی انجام می‌شود.

```
RAILS_ENV=production bundle exec rake assets:precompile
```

مشکل در پایگاه داده

در صورت بروز هر گونه مشکل در پایگاه داده از قبیل کند شدن mysql و یا عدم اتصال برنامه با mysql می‌توان با استفاده از دستور زیر mysql را ریستارت کرد.

```
service mysql restart
```

در صورت لزوم به تغییر پسورد و یا کانکشن‌های mysql این تغییرات باید در برنامه نیز اعمال شود.

اطلاعات مربوط تنظیمات پایگاه داده در فایلی به مسیر

/rootOfapplication/config/database.yml وجود دارد.

```
default: &default
  adapter: mysql۲
  encoding: utf۸
  pool: ۵
  username: ##################
  password: ##################
  socket: /var/run/mysqld/mysqld.sock

development:
* :>>  default
  database: edu_development
```

خطایابی

به منظور خطایابی در برنامه‌های تحت چارچوب rails موثرترین روش بررسی لاگ‌های مربوط به برنامه است. درحال حاضر لاگ‌های برنامه سایت کنونی بر روی سطح error تنظیم شده است. بدین ترتیب در صورت به وجود آمدن هر error در فرآیند برنامه، لاگ‌های آن تولید شده و به انتهای فایل /rootOfapplication/log/production.log افزوده می‌شود. همچنین می‌توان سطح تولید لاگ را تنظیم کرد و بر روی سطوح دیگر نظیر debugging قرار داد. برای مثال در این سطح لاگ مربوط به تمامی فرآیندها از لود شدن یک صفحه، درخواست اتصال به پایگاه داده تولید می‌شود. از این سطح تولید لاگ معمولا در زمان طراحی نرم‌افزار استفاده می‌شود. معمولا برای خطایابی برنامه‌های تحت اجرا سطح error می‌تواند برای حل تمامی مشکلات استفاده شود. سطح دیگری نیز برای تولید لاگ وجود دارد که با عنوان warning استفاده می‌شود. د یان سطح

هشدارهای احتمالی نیز در فرایندهای اجرای برنامه لاگ می‌شوند. برای تغییر این سطح تولید لاگ از فایل زیر استفاده کنید.

```
/root_of_application/environments/production.rb
```

در فایل بالا با استفاده از دستورات زیر می‌توان سطح لاگ را بررسی کرد:

```
 #  Use the lowest log level to ensure availability of diagnostic information
 #  when problems arise.
#  config.log_level = :debug
  config.log_level = :error
```

پیوست ب

اسکریپ‌های طراحی شده به منظور تولید پایگاه داده موردنیاز برای نگهداری و بازیابی تصاویر در ادامه آورده شده است.

```
-- phpMyAdmin SQL Dump
-- version 4.8.3
-- https://www.phpmyadmin.net/
--
-- Host: localhost
-- Generation Time: Oct 19, 2018 at 02:00 AM
-- Server version: 5.7.23-0ubuntu0.18.04.1
-- PHP Version: 7.2.10-0ubuntu0.18.04.1

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
--
-- Database: `ganj2_development`
```

```
--
-- ------------------------------------------------------
--
-- Table structure for table `images`
--

CREATE TABLE `images` (
 `id` int(11) NOT NULL,
 `article_id` int(11) DEFAULT NULL,
 `uuid` varchar(255) COLLATE utf8_persian_ci DEFAULT NULL,
 `slug` varchar(255) COLLATE utf8_persian_ci DEFAULT NULL,
 `title_fa` varchar(255) COLLATE utf8_persian_ci DEFAULT NULL,
 `title_en` varchar(255) COLLATE utf8_persian_ci DEFAULT NULL,
 `description` text COLLATE utf8_persian_ci,
 `size` int(11) DEFAULT NULL,
 `height` int(11) DEFAULT NULL,
 `width` int(11) DEFAULT NULL,
 `created_at` datetime NOT NULL,
 `updated_at` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_persian_ci;

-- ------------------------------------------------------
--
-- Table structure for table `image_descriptors`
--
CREATE TABLE `image_descriptors` (
 `id` int(11) NOT NULL,
```

```
 `image_id` int(11) DEFAULT NULL,
 `image_tag_id` int(11) DEFAULT NULL,
 `created_at` datetime NOT NULL,
 `updated_at` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_persian_ci;
-- --------------------------------------------------------
--
-- Table structure for table `image_tags`
--
CREATE TABLE `image_tags` (
 `id` int(11) NOT NULL,
 `title_fa` varchar(255) COLLATE utf8_persian_ci DEFAULT NULL,
 `title_en` varchar(255) COLLATE utf8_persian_ci DEFAULT NULL,
 `created_at` datetime NOT NULL,
 `updated_at` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_persian_ci;
--
-- Indexes for dumped tables
--
--
-- Indexes for table `images`
--
ALTER TABLE `images`
 ADD PRIMARY KEY (`id`),
 ADD KEY `index_images_on_article_id` (`article_id`),
 ADD KEY `index_images_on_uuid` (`uuid`);
--
```

```
-- Indexes for table `image_descriptors`
--
ALTER TABLE `image_descriptors`
 ADD PRIMARY KEY (`id`),
 ADD KEY `index_image_descriptors_on_image_id` (`image_id`),
 ADD KEY `index_image_descriptors_on_image_tag_id` (`image_tag_id`);
--
-- Indexes for table `image_tags`
--
ALTER TABLE `image_tags`
 ADD PRIMARY KEY (`id`);


--
-- AUTO_INCREMENT for dumped tables
--
--
-- AUTO_INCREMENT for table `images`
--
ALTER TABLE `images`
 MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `image_descriptors`
--
ALTER TABLE `image_descriptors`
 MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `image_tags`
```

```
--

ALTER TABLE `image_tags`

 MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--

-- Constraints for dumped tables

--

--

-- Constraints for table `images`

--

ALTER TABLE `images`

 ADD CONSTRAINT `fk_rails_028d35b473` FOREIGN KEY (`article_id`)
REFERENCES `articles` (`id`);

--

-- Constraints for table `image_descriptors`

--

ALTER TABLE `image_descriptors`

 ADD CONSTRAINT `fk_rails_1269f692d9` FOREIGN KEY (`image_tag_id`)
REFERENCES `image_tags` (`id`),

 ADD CONSTRAINT `fk_rails_f5c156ecbc` FOREIGN KEY (`image_id`)
REFERENCES `images` (`id`);

COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
*/;
```

پیوست پ

کد پایتون برای استخراج تصاویر از فایل ورد در ادامه آورده شده است.

```
# In[]:
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 23 19:51:01 2019

@author: Seddighi
@modified by: Fakhrzadeh
"""
import uuid
import zipfile
import os
from lxml import etree
import re
from openpyxl import Workbook
from openpyxl import load_workbook
import datetime
from PIL import Image
import csv
import shutil
#from hazm import *   #added by AF
#import re   #added by AF
import platform
import subprocess
import win32com.client

# XML Namespaces
nsmap = {
   'a': 'http://schemas.openxmlformats.org/drawingml/2006/main',
   'a14': 'http://schemas.microsoft.com/office/drawing/2010/main',
   'm': 'http://schemas.openxmlformats.org/officeDocument/2006/math',
   'mc': 'http://schemas.openxmlformats.org/markup-compatibility/2006',
   'o': 'urn:schemas-microsoft-com:office:office',
   'pic': 'http://schemas.openxmlformats.org/drawingml/2006/picture',
   'r': 'http://schemas.openxmlformats.org/officeDocument/2006/relationships',
```

```
    'v': 'urn:schemas-microsoft-com:vml',
    'w': 'http://schemas.openxmlformats.org/wordprocessingml/2006/main',
    'w10': 'urn:schemas-microsoft-com:office:word',
    'w14': 'http://schemas.microsoft.com/office/word/2010/wordml',
    'wne': 'http://schemas.microsoft.com/office/word/2006/wordml',
    'wp': 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing',
    'wp14': 'http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing',
    'wpc': 'http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas',
    'wpg': 'http://schemas.microsoft.com/office/word/2010/wordprocessingGroup',
    'wpi': 'http://schemas.microsoft.com/office/word/2010/wordprocessingInk',
    'wps': 'http://schemas.microsoft.com/office/word/2010/wordprocessingShape',
}

def doc2docx(src_path, dst_path):
    '''
    Convert a doc file to a docx file using Microsft Word 2007 or higher in Windows
    and LibreOffice in Linux platform.
    src_path: relative path to the source file (path/*.doc)
    dst_path: relative path to the destination file (path/*.docx)
    '''
    status = 1 # Normal status
    current_platform = platform.system()
    if(current_platform == 'Linux'):
        if src_path.endswith('.doc'):
            try:
                dst_folder = os.path.dirname(dst_path)
                subprocess.call(['soffice', '--headless', '--convert-to', 'docx', '--outdir', dst_folder,
src_path])
            except:
                status = 0 # required software has not installed

    elif(current_platform == 'Windows'):
        if src_path.endswith('.doc'):
            try:
                word = win32com.client.Dispatch("Word.Application")
                src_abs_path = os.path.abspath(src_path)
                dst_abs_path = os.path.abspath(dst_path)
                doc = word.Documents.Open(src_abs_path)
                doc.SaveAs2(dst_abs_path, FileFormat=16) # file format for docx
                doc.Close()
                word.Quit()
            except:
                status = -1 # required software has not installed
```

```
    else:
        # Mac: Darwin
        status = -2 # platform is not supported

    return status

def qn(tag):
    '''
    Stands for 'qualified name', a utility function to turn a namespace
    prefixed tag name into a Clark-notation qualified tag name for lxml. For
    example, ``qn('p:cSld')`` returns ``'{http://schemas.../main}cSld'``.
    Source: https://github.com/python-openxml/python-docx/
    '''
    prefix, tagroot = tag.split(':')
    uri = nsmap[prefix]
    return '{{{}}}{}'.format(uri, tagroot)

def get_word_xml(docx_filename, xml_name = 'word/document.xml'):
    '''
    Reads a docx file and extracts the embedded xml file from it.
    Then it returns the xml tree of the target xml file.
    The target xml file is defined by xml_name parameter, which is
    document.xml (main xml file in docx format) by default.
    '''
    with open(docx_filename, 'rb') as f:
        ziph = zipfile.ZipFile(f)
        xml_content = ziph.read(xml_name)
        xml_tree = etree.fromstring(xml_content)
        ziph.close()
    return xml_tree

def update_xml_nsmap(xml_tree):
    '''
    Updates the default xml namespaces using a new xml tree.
    It is used to include the new namespaces, which are not
    in our default list.
    '''
    for child in xml_tree.iter():
        nsmap.update(child.nsmap)
    return nsmap

def none_handling(text, replace=u''):
    '''
```

```
    Replaces input (text) that is none with empty srting (replace parameter).
    '''
    result = text if text is not None else replace
    return result

def elm2text(xml_element):
    '''
    A string representing the textual content of a xml element, with content
    child elements like ``<w:tab/>`` translated to their Python
    equivalent.
    Adapted from: https://github.com/python-openxml/python-docx/
    '''
    text = u''
    for child in xml_element.iterdescendants():
        if child.tag == qn('w:t'):
            t_text = child.text
            text += none_handling(t_text)
        elif child.tag == qn('w:noBreakHyphen'):
            text += '\u2011'
        elif child.tag == qn('w:tab'):
            text += '\t'
        elif child.tag in (qn('w:br'), qn('w:cr')):
            text += '\n'
    return text

def p_is_blank(paragraph):
    '''
    Checkes whether a paragraph is blank or not. Ruterns true or false.
    '''
    result = False

    # Matches any non-whitespace character; this is equivalent to the class [^ \t\n\r\f\v]
    pattern = re.compile('\S')

    if(etree.iselement(paragraph) and paragraph.tag == qn('w:p')):
        if(len(paragraph)):
            for child in paragraph.iterdescendants():
                if child.tag == qn('w:drawing'):
                    result = False
                    break
                elif child.tag == qn('w:noBreakHyphen'):
                    result = False
                    break
```

```
            elif child.tag == qn('w:softHyphen'):
                result = False
                break
            elif child.tag == qn('w:sym'):
                result = False
                break
            elif child.tag == qn('w:bookmarkStart'):
                result = True
            elif child.tag == qn('w:bookmarkEnd'):
                result = True
            elif child.tag == qn('w:t'):
                t_text = child.text
                t_text = none_handling(t_text)

                # search for non-whitespace characters
                if pattern.search(t_text) is not None:
                    result = False
                    break
                else:
                    result = True
            else:
                result = True

        else:
            result = True
    else:
        result = None

    return result

def is_blank(element):
    '''
    Checkes whether an element is blank or not. Ruterns true or false.
    '''
    result = False

    if(etree.iselement(element)):
        # self and childeren which are paragraph
        paragraphs = list(element.iter(qn('w:p')))
        if(len(paragraphs)):
            for child in element.iter(qn('w:p')):
                result = p_is_blank(child)
                if(result == False):
```

```
                break
        else:
            result = True
    else:
        result = None

    return result

def is_within_table(element):
    '''
    Checkes whether an element is within table or not. Ruterns true or false.
    '''
    result = False
    if(etree.iselement(element)):
        tables = list(element.iterancestors(qn('w:tbl')))
        if(len(tables)):
            result = True
    else:
        result = None

    return result

def has_element(element, target_element):
    '''
    Checkes whether an element has a terget element in its descendants or not.
    Ruterns true or false.
    '''
    result = False
    if(etree.iselement(element)):
        target_list = list(element.iterdescendants(qn(target_element)))
        if(len(target_list)):
            result = True
    else:
        result = None

    return result

def has_picture(element):
    '''
    Checkes whether an element has a picture or not. Ruterns true or false.
    '''
    return has_element(element, 'a:blip')
```

```
def has_caption(element):
    '''
    Checkes whether an element has a caption or not. Ruterns true or false.
    '''
    result = False
    if(etree.iselement(element)):
        for caption in element.iterdescendants(qn('w:pStyle')):
            if(caption.get(qn('w:val')) == "Caption"):
                result = True
                break
    else:
        result = None

    return result

def get_cell_width(cell):
    '''
    Ruterns the width of a cell.
    '''
    width = 0
    if(etree.iselement(cell) and cell.tag == qn('w:tc')):
        for child in cell.iterdescendants(qn('w:tcW')):
            width = int(child.get(qn('w:w')))
            width = none_handling(width, 0)
    else:
        width = None

    return width

def cells_intersection(dis, width, w_sum, tc_width):
    '''
    Computes the intersection of two cells, which are in two consecutive rows.
    Returns the start point (distance from left) and width of intersection
    part.
    inputs
    dis: cell1 distance from left
    width: cell1 width
    w_sum: cell2 distance from left plus its width
    tc_width: cell2 width
    outputs
    new_dis: intersection distance from left
    new_width: intersection width
    '''
```

```
    tc_left = w_sum - tc_width
    tc_right = w_sum
    if(tc_left < dis):
        new_dis = dis
        if(tc_right < dis + width):
            new_width = w_sum - dis
        else:
            new_width = width
    else:
        new_dis = w_sum - tc_width
        if(tc_right < dis + width):
            new_width = tc_width
        else:
            new_width = tc_width - w_sum + dis + width

    return new_dis, new_width

def add_text(info, text, delimiter=' | '):
    '''
    Adds a text to an info using a delimiter.
    '''
    if(info == u''):
        result = text
    elif(text == u''):
        result = info
    else:
        result = info + delimiter + text
    return result

def intercell_info(table, tr, dis, width, text_state = False):
    '''
    Extracts the information that is in the next row of a table and under a
    cell determined with dis and width.
    table: table element
    tr: row element which includes the target cell
    dis: cell distance from left
    width: cell width
    text_state: if cell has innertext then text_state will be true.
    '''
    info = u''
    caption_state = False
    if (etree.iselement(table) and table.tag == qn('w:tbl') and etree.iselement(tr) and tr.tag
== qn('w:tr')):
```

```
      next_tr = tr.getnext()
      if(next_tr is not None):
        w_sum = 0
        for tc in next_tr.iterdescendants(qn('w:tc')):
          tc_width = get_cell_width(tc)
          w_sum += tc_width
          if(w_sum > dis):
            if(not has_picture(tc)):
              if(not is_blank(tc)):
                info = add_text(info, elm2text(tc))
                if(has_caption(tc)):
                  caption_state = True
              else:
                # check next row for blank cells
                new_dis, new_width = cells_intersection(dis, width, w_sum, tc_width)

                new_info, temp_state = intercell_info(table, next_tr, new_dis, new_width)
                new_info = none_handling(new_info)
                info = add_text(info, new_info)
                if(temp_state == True):
                  caption_state = True
            else:
              if(not text_state):
                temp = elm2text(tc)
                if(temp == u''):
                  # check next row for picture cells which doesn't have text
                  new_dis, new_width = cells_intersection(dis, width, w_sum, tc_width)

                  new_info, temp_state = intercell_info(table, next_tr, new_dis,
new_width)
                  new_info = none_handling(new_info)
                  info = add_text(info, new_info)
                  if(temp_state == True):
                    caption_state = True
                else:
                  info = add_text(info, temp)
                  if(has_caption(tc)):
                    caption_state = True

            if(w_sum < dis + width):
              # more cells must processed
              continue
            else:
```

```
                break
    else:
        info = None
        caption_state = None

    return info, caption_state

def is_cell_vertical(cell):
    '''
    Checkes whether a cell is vertical or not. Ruterns true or false and the
    state of the vertical cell.
    '''
    result = False
    state = None
    if(etree.iselement(cell) and cell.tag == qn('w:tc')):
        vertical= list(cell.iterdescendants(qn('w:vMerge')))
        if(len(vertical)):
            result = True
            for child in cell.iterdescendants(qn('w:vMerge')):
                value = child.get(qn('w:val'))
                if(value == 'restart'):
                    state = 'start'
                else:
                    state = 'continue'
    else:
        result = None

    return result, state

def get_next_info(element):
    '''
    Returns the text information that comes after an element.
    '''
    # element can be either paragraph or table
    info = u''
    caption_state = False
    if(etree.iselement(element)):
        for next_element in element.itersiblings():
            if(next_element.tag == qn('w:p')):
                paragraph = next_element
                if(not has_picture(paragraph)):
                    if(not is_blank(paragraph)):
                        info = elm2text(paragraph)
```

```
                if(has_caption(paragraph)):
                    caption_state = True
                break
            else:
                continue
        else:
            continue
    elif(next_element.tag == qn('w:tbl')):
        table = next_element
        for tr in table.iterdescendants(qn('w:tr')):
            if(not has_picture(tr)):
                if(not is_blank(tr)):
                    info = elm2text(tr)
                    if(has_caption(tr)):
                        caption_state = True
                    break
                else:
                    continue
            else:
                temp = elm2text(tr)
                if (temp == u''):
                    continue
                else:
                    info = temp
                    if(has_caption(tr)):
                        caption_state = True
                    break
    else:
        break

else:
    info = None
    caption_state = None

return info, caption_state

def get_next_caption(element):
    '''
    Returns the caption information that comes after an element.
    '''
    # element can be either paragraph or table
    info = u''
    caption_state = False
```

```
if(etree.iselement(element)):
    for next_element in element.itersiblings():
        if(next_element.tag == qn('w:p')):
            paragraph = next_element
            if(not has_picture(paragraph)):
                if(not is_blank(paragraph)):
                    caption = list(paragraph.iterdescendants(qn('w:pStyle')))
                    if(len(caption)):
                        caption = caption[0]
                        if(caption.get(qn('w:val')) == "Caption"):
                            info = elm2text(paragraph)
                            caption_state = True
                            break
                        else:
                            break
                    else:
                        break
                else:
                    continue
            else:
                continue
        elif(next_element.tag == qn('w:tbl')):
            table = next_element
            for tr in table.iterdescendants(qn('w:tr')):
                if(not has_picture(tr)):
                    if(not is_blank(tr)):
                        for paragraph in tr.iterdescendants(qn('w:p')):
                            caption = list(paragraph.iterdescendants(qn('w:pStyle')))
                            if(len(caption)):
                                caption = caption[0]
                                if(caption.get(qn('w:val')) == "Caption"):
                                    info = elm2text(paragraph)
                                    caption_state = True
                                    break
                                else:
                                    continue
                            else:
                                continue
                        break
                    else:
                        continue
                else:
                    temp = elm2text(tr)
```

```
                if (temp == u''):
                    continue
                else:
                    for paragraph in tr.iterdescendants(qn('w:p')):
                        caption = list(paragraph.iterdescendants(qn('w:pStyle')))
                        if(len(caption)):
                            caption = caption[0]
                            if(caption.get(qn('w:val')) == "Caption"):
                                info = elm2text(paragraph)
                                caption_state = True
                                break
                            else:
                                continue
                        else:
                            continue
                    break
            else:
                break
    else:
        info = None
        caption_state = None

    return info, caption_state

def cell_info(table, cell, element):
    '''
    Extracts the information related to an element, which is in a cell within
    a table.
    '''
    intrainfo = u''
    interinfo = u''
    tableinfo = u''
    intra_caption_state = False
    inter_caption_state = False
    table_caption_state = False
    if (etree.iselement(table) and table.tag == qn('w:tbl') and etree.iselement(cell) and
cell.tag == qn('w:tc')):
        if(has_picture(cell)):
            #-----------------------------------------------------
            # extract intracell info
            #-----------------------------------------------------
            intrainfo, intra_caption_state = get_next_info(element)
            #-----------------------------------------------------
```

```
        # extract intercell info
        #----------------------------------------------------
        # compute dis and width
        width = get_cell_width(cell)
        dis = 0
        for pre_sib in cell.itersiblings(tag=qn('w:tc'), preceding=True):
            dis += get_cell_width(pre_sib)
        tr = cell.getparent()

        # check for vertical cell
        result, state = is_cell_vertical(cell)
        while(result):
            next_tr = tr.getnext()
            if(next_tr is not None):
                w_sum = 0
                for tc in next_tr.iterdescendants(qn('w:tc')):
                    tc_width = get_cell_width(tc)
                    w_sum += tc_width
                    if(w_sum > dis):
                        if(tc_width == width):
                            tc_result, tc_state = is_cell_vertical(tc)
                            if(tc_result and tc_state == 'continue'):
                                tr = next_tr
                            else:
                                result = False
                                break
                        else:
                            result = False
                            break
            else:
                result = False
                break

        if(intrainfo == u''):
            text_state = False
        else:
            text_state = True
        interinfo, inter_caption_state = intercell_info(table, tr, dis, width, text_state)
        #----------------------------------------------------
        # extract table next info
        #----------------------------------------------------
        tableinfo, table_caption_state = get_next_caption(table)
    else:
```

```
        intrainfo = None
        interinfo = None
        tableinfo = None
        intra_caption_state = None
        inter_caption_state = None
        table_caption_state = None

    return intrainfo, interinfo, tableinfo, intra_caption_state, inter_caption_state,
table_caption_state

def detect_language(text):
    '''
    Detects the text language.
    '''
    lang = 'English'

    # Persian alphabet
    persian = re.compile('[\u0621-\u0628\u062A-\u063A\u0641-\u0642\u0644-
\u0648\u064E-\u0651\u0655\u067E\u0686\u0698\u06A9\u06AF\u06BE\u06CC]')

    # search for persian characters
    if persian.search(text) is not None:
        lang = 'Persian'

    return lang

def get_file_list(path, file_type='docx'):
    '''
    Returns a list of files in a certain path, which have a common file type.
    '''
    file_list = []
    for file in os.listdir(path):
        if file.endswith("." + str(file_type)):
            file_list.append(file)
    return file_list

def desc2vec(pic_desc, delimiter=' | '):
    '''
    Combines the text of pic_desc using a delimiter to vectorize the information.
    '''
    pic_direct_desc = pic_desc['pic_direct_desc']
    pic_intermediate_desc = pic_desc['pic_intermediate_desc']
    pic_general_desc = pic_desc['pic_general_desc']
```

```
    desc = u''
    for i in reversed(range(len(pic_general_desc))):
        desc = add_text(desc, pic_general_desc[i], delimiter)
        desc = add_text(desc, pic_intermediate_desc[i], delimiter)
        desc = add_text(desc, pic_direct_desc[i], delimiter)
    return desc

def unique(item):
    '''
    Returns a list of unique itmes.
    '''
    result = []
    length = len(item)
    for i in range(length):
        state = 1 # item is unique
        for j in range(i+1,length):
            if item[i] == item[j]:
                state = 0
                break
        if state:
            result.append(item[i])
    return result

def img_extract(docx_filename, img_dir='media', save_policy='normal',
file_per_folder=1000):
    '''
    Extracts the images and their corresponding captions (main function).
    docx_filename: document filename
    img_dir: image directory that will be used to store extracted images.
    save_policy: policy for saving images (normal or multiple folder)
    file_per_folder: number of files in each folder if save_policy is multiple
    '''
    if not os.path.exists(docx_filename):
        print('File {} does not exist.'.format(docx_filename))
    else:
        # read document file
        xml_tree = get_word_xml(docx_filename)

        # update namespaces
        global nsmap
        nsmap = update_xml_nsmap(xml_tree)

        # find id and description of pictures which have caption
```

```
    pics = []
    lang = ''
    for paragraph in xml_tree.iter(qn('w:p')):
      for child in paragraph.iterdescendants(qn('a:blip')):
        pic_id = child.get(qn('r:embed'))
        pic_desc = u''

        # paragraph is within a table
        if(is_within_table(paragraph)):
          pic_direct_desc = []
          pic_intermediate_desc = []
          pic_general_desc = []
          empty = []
          cell = paragraph.getparent()
          element = paragraph
          lang = 'English'
          caption_state = False
          for table in paragraph.iterancestors(qn('w:tbl')):
            intrainfo, interinfo, tableinfo, intra_caption_state, inter_caption_state,
table_caption_state = cell_info(table, cell, element)
            intrainfo = none_handling(intrainfo)
            interinfo = none_handling(interinfo)
            tableinfo = none_handling(tableinfo)
            if(detect_language(intrainfo) == 'Persian' or detect_language(interinfo) ==
'Persian' or detect_language(tableinfo) == 'Persian'):
              lang = 'Persian'
            if(intra_caption_state or inter_caption_state or table_caption_state):
              caption_state = True
            pic_direct_desc.append(intrainfo)
            pic_intermediate_desc.append(interinfo)
            pic_general_desc.append(tableinfo)
            empty.append(u'')
            cell = table.getparent()
            element = table
          if(pic_direct_desc != empty or pic_intermediate_desc != empty or
pic_general_desc != empty):
            if(caption_state):
              pic_desc = {'pic_direct_desc': pic_direct_desc, 'pic_intermediate_desc':
pic_intermediate_desc, 'pic_general_desc': pic_general_desc}
            else:
              pic_desc = u''
          else:
            pic_desc = u''
```

```
        else:
            # paragraph is within body
            pic_desc, _ = get_next_caption(paragraph)
            pic_desc = none_handling(pic_desc)
            lang = detect_language(pic_desc)

        if(pic_desc != u''):
            pics.append([pic_id, pic_desc, lang])

# remove duplicates
pics = unique(pics)

# find filenames of pictures
xml_name = 'word/_rels/document.xml.rels'
rels_tree = get_word_xml(docx_filename, xml_name = xml_name)
pics_info = []
for pic in pics:
    pid = pic[0]
    pdesc = pic[1]
    lang = pic[2]
    puid = str(uuid.uuid4()).replace('-', '')
    for child in rels_tree.iter():
        if child.get('Id') == pid:
            path = child.get('Target')
            pics_info.append({'path':path, 'id': pid, 'desc': pdesc, 'lang': lang, 'uid': puid})

# create img_dir
if save_policy == 'multiple':
    # computing number of folders for saving images under multiple folder policy
    num_pics = len(pics_info)
    num_folder = 0
    if num_pics % file_per_folder > 0:
        num_folder = round(num_pics / file_per_folder) + 1
    else:
        num_folder = round(num_pics / file_per_folder)
    # create folders
    img_dir_base = img_dir
    for idx in range(1,num_folder + 1):
        img_dir = img_dir_base + '_' + str(idx)
        if not os.path.exists(img_dir):
            try:
                os.makedirs(img_dir)
            except OSError:
```

```
            print("Unable to create image directory {}".format(img_dir))
            return None

        # extract pictures and write them to folders
        with open(docx_filename, 'rb') as f:
          ziph = zipfile.ZipFile(f)
          end_range = idx*file_per_folder
          if idx == num_folder:
            end_range = num_pics
          for pic in pics_info[(idx-1)*file_per_folder:end_range]:
            src_path = "word/" + pic["path"]
            dst_path = os.path.join(img_dir, os.path.basename(pic["path"]))

            with open(dst_path, "wb") as dst_f:
              dst_f.write(ziph.read(src_path))

          ziph.close()
    else:
      if not os.path.exists(img_dir):
        try:
          os.makedirs(img_dir)
        except OSError:
          print("Unable to create image directory {}".format(img_dir))
          return None

      # extract pictures and write them to img_dir
      with open(docx_filename, 'rb') as f:
        ziph = zipfile.ZipFile(f)
        for pic in pics_info:
          src_path = "word/" + pic["path"]
          dst_path = os.path.join(img_dir, os.path.basename(pic["path"]))

          with open(dst_path, "wb") as dst_f:
            dst_f.write(ziph.read(src_path))

        ziph.close()

    return pics_info

def create_image_db(path_src, path_dst, db_title='ImageDB', thumbnail_size=(128,128),
save_policy='normal', file_per_folder=1000):
  '''
  Creates an image database in excel and csv format. Extracts all images,
```

```
    their corresponding information, and image thumbnails.
    '''
    # Creating an excel file for saving information
    wb = Workbook()
    ws = wb.active
    ws.title = db_title

    # information header
    row = 1
    # the last element fig number is added by AF
    header = ['id', 'article_id', 'uid', 'title_fa', 'title_en', 'description', 'size', 'height', 'width',
'created_at', 'updated_at']
    for j in range(len(header)):
        ws.cell(row=row, column=j+1, value=header[j])

    # Converting doc files to docx files
    doc_files = get_file_list(path_src, 'doc')
    state = 1 # normal convert
    for doc in doc_files:
        file_type = 'doc'
        doc_name_only = doc[:-(len(file_type)+1)]
        doc_src_path = path_src + '/' + doc
        docx_dst_path = path_src + '/' + doc_name_only + '.docx'
        state = doc2docx(doc_src_path, docx_dst_path)

    if(state == 0):
        print("LibreOffice must be installed on this machine.")
    elif(state == -1):
        print("Microsoft Word 2007 or higher must be installed on this machine.")
    elif(state == -2):
        print("Platform is not supported.")

    path_dst_temp = path_dst+'/'+'temp'
    file_type = 'docx'

    filenames = get_file_list(path_src, 'docx')
    counter = 0
    withoutC = 0 #AF

    # processing each file
    for file in filenames:
        try:
            print('file {} is processed'.format(file)) #AF
```

```
docx_name_only = file[:-(len(file_type)+1)]
docx_filename = path_src + '/' + file
img_dir = path_dst + '/' + docx_name_only
img_dir_temp = path_dst_temp + '/' + docx_name_only
pics_info = img_extract(docx_filename, img_dir_temp, save_policy, file_per_folder)

pics_info = unique(pics_info)

if not pics_info:  #AF
   withoutC = withoutC + 1   #AF

if save_policy == 'multiple':
   # computing number of folders for saving images under multiple folder policy
   num_pics = len(pics_info)
   num_folder = 0
   if num_pics % file_per_folder > 0:
      num_folder = round(num_pics / file_per_folder) + 1
   else:
      num_folder = round(num_pics / file_per_folder)

   img_dir_base = img_dir
   img_dir_temp_base = img_dir_temp
   for idx in range(1,num_folder + 1):
      img_dir = img_dir_base + '_' + str(idx)
      img_dir_temp = img_dir_temp_base + '_' + str(idx)
      end_range = idx*file_per_folder
      if idx == num_folder:
         end_range = num_pics
      for pic in pics_info[(idx-1)*file_per_folder:end_range]:
         row += 1
         desc = pic['desc']
         lang = pic['lang']
         title_fa = ''
         title_en = ''
         if(type(desc) is str):
            if lang == 'Persian':
               title_fa = desc
               title_en = ''
            else:
               title_fa = ''
               title_en = desc
         elif(type(desc) is dict):
            if lang == 'Persian':
```

```
                title_fa = desc2vec(desc)
                title_en = ''
            else:
                title_fa = ''
                title_en = desc2vec(desc)

        pic_name = pic['path'][6:]
        img_filename = img_dir_temp + '/' + pic_name

        # image file size in KB
        img_size = round(os.path.getsize(img_filename)/1024,1)

        # create img_dir
        if not os.path.exists(img_dir):
            try:
                os.makedirs(img_dir)
            except OSError:
                print("Unable to create image directory {}".format(img_dir))

        # image change type to jpeg and create thumbnails
        # image dimensions in pixel
        with Image.open(img_filename) as image:
            width, height = image.size
            pic_name = pic["uid"] + '.jpg'
            img_filename = img_dir + '/' + pic_name
            image.convert('RGB').save(os.path.join(img_dir, pic_name), 'JPEG')
            image.thumbnail(thumbnail_size)
            image.convert('RGB').save(os.path.join(img_dir, "thumbnail" + pic_name),
'JPEG')

        record = [row - 1, docx_name_only, pic["uid"], title_fa, title_en, '',
                  img_size, height, width, datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S"),
                  datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")]

        for j in range(len(header)):
            ws.cell(row=row, column=j+1, value=record[j])

    else:
        for pic in pics_info:
            row += 1
            desc = pic['desc']
            lang = pic['lang']
```

```
            title_fa = ''
            title_en = ''
            if(type(desc) is str):
              if lang == 'Persian':
                title_fa = desc
                title_en = ''
              else:
                title_fa = ''
                title_en = desc
            elif(type(desc) is dict):
              if lang == 'Persian':
                title_fa = desc2vec(desc)
                title_en = ''
              else:
                title_fa = ''
                title_en = desc2vec(desc)

            pic_name = pic['path'][6:]
            img_filename = img_dir_temp + '/' + pic_name

            # image file size in KB
            img_size = round(os.path.getsize(img_filename)/1024,1)

            # create img_dir
            if not os.path.exists(img_dir):
              try:
                os.makedirs(img_dir)
              except OSError:
                print("Unable to create image directory {}".format(img_dir))

            # image change type to jpeg and create thumbnails
            # image dimensions in pixel
            with Image.open(img_filename) as image:
              width, height = image.size
              pic_name = pic["uid"] + '.jpg'
              img_filename = img_dir + '/' + pic_name
              image.convert('RGB').save(os.path.join(img_dir, pic_name), 'JPEG')
              image.thumbnail(thumbnail_size)
              image.convert('RGB').save(os.path.join(img_dir, "thumbnail" + pic_name),
'JPEG')

            record = [row - 1, docx_name_only, pic["uid"], title_fa, title_en, '',
                  img_size, height, width, datetime.datetime.now().strftime("%Y-%m-%d
```

```
%H:%M:%S"),
                    datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")]

            for j in range(len(header)):
              ws.cell(row=row, column=j+1, value=record[j])

        counter += 1

      except Exception as exp:
        print('somthing is wrong about file {}. Following exception occurred during image
extraction: \n{}'.format(file, exp))

  # removing temporary folder
  if os.path.exists(path_dst_temp):
    shutil.rmtree(path_dst_temp)  # remove dir and all contains

  # saving information in excel file
  try:
    print('{} files were processed.'.format(counter))
    print('{} files did not have caption.'.format(withoutC)) #added by AF
    outputpath=path_src+'/'+'ImageDB.xlsx' ##AF
    wb.save( outputpath)
    wb = load_workbook(filename = outputpath)
    sh = wb.active
    cvs_path=path_src+'/'+'ImageDB.csv'
    with open(cvs_path, 'w', newline="", encoding='utf-8') as csv_file:
      c = csv.writer(csv_file)
      for r in sh.rows:
        c.writerow([cell.value for cell in r])

  except Exception as exp:
    print('somthing is wrong. Following exception occurred during saving Excel file:
\n{}'.format(exp))

# In[]:

# information about files and paths
#path_src = 'C://Users//Fakhrzadeh.INTRANET//Documents//word_fani' ##AF
path_src = 'words'
path_dst = path_src+'/'+'images' ##AF

db_title='ImageDB'
```

```
# thumbnail size
thumbnail_size = (128,128)

# save policy
save_policy = 'multiple' #or 'normal'
file_per_folder = 10

create_image_db(path_src, path_dst, db_title, thumbnail_size, save_policy, file_per_folder)
```

پیوست ت

کد جاوا برای استخراج تصاویر از فایل پی.دی.اف

```
package org.pdfBoxPoject.V11;

import org.apache.pdfbox.cos.COSBase;

import org.apache.pdfbox.cos.COSName;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDPage;

import org.apache.pdfbox.pdmodel.graphics.PDXObject;

import org.apache.pdfbox.pdmodel.graphics.form.PDFormXObject;

import org.apache.pdfbox.pdmodel.graphics.image.PDImageXObject;

import org.apache.pdfbox.util.Matrix;

import org.apache.pdfbox.contentstream.operator.DrawObject;

import org.apache.pdfbox.contentstream.operator.Operator;

import org.apache.pdfbox.contentstream.PDFStreamEngine;


import java.awt.geom.AffineTransform;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;


import org.apache.pdfbox.contentstream.operator.state.Concatenate;

import org.apache.pdfbox.contentstream.operator.state.Restore;

import org.apache.pdfbox.contentstream.operator.state.Save;

import org.apache.pdfbox.contentstream.operator.state.SetGraphicsStateParameters;
```

```
import org.apache.pdfbox.contentstream.operator.state.SetMatrix;


**/
 *This is an example on how to get the x/y coordinates of image location and size of image.
*https://www.tutorialkart.com/pdfbox/how-to-get-location-and-size-of-images-in-pdf/
/*
public class GetImageLocationsAndSize extends PDFStreamEngine
}
        ArrayList<Integer> imageWidth=new ArrayList<Integer;()<
         ArrayList<Integer> imageHeight=new ArrayList<Integer;()<

         ArrayList<Float> imageXScale=new ArrayList<Float;()<
         ArrayList<Float> imageYScale=new ArrayList<Float;()<

         ArrayList<Float> xLocationInPdf=new ArrayList<Float;()<
         ArrayList<Float> yLocationInPdf=new ArrayList<Float;()<

         ArrayList<PDImageXObject> images=new ArrayList<PDImageXObject;()<
        //private int imageHeight;
        */private float imageXScale;
        private float imageYScale;
        float xLocationInPdf;
        float yLocationInPdf/*;
```

```
            //PDDocument document;
**/
@ *    throws IOException If there is an error loading text stripper properties.
/*
  public GetImageLocationsAndSize() throws IOException
}
 //     preparing PDFStreamEngine
    addOperator(new Concatenate;(()
    addOperator(new DrawObject;(()
    addOperator(new SetGraphicsStateParameters;(()
    addOperator(new Save;(()
    addOperator(new Restore;(()
    addOperator(new SetMatrix;(()
{


**/
@ *    throws IOException If there is an error parsing the document.
/*
*/  public static void main( String[] args ) throws IOException
}
    PDDocument document = null;
    String fileName = "C:/Users/ashoo/OneDrive/Documents/PDFBox/test/t/t.pdf;"
    try
}
      document = PDDocument.load( new File(fileName;( (
      GetImageLocationsAndSize printer = new GetImageLocationsAndSize;()
      int pageNum = 0;
```

```
     for( PDPage page : document.getPages( ()
}
       pageNum;++
       System.out.println( "\n\nProcessing page: " + pageNum +"\n----------
;("--
       printer.processPage(page;(
{
{
   finally
}
     if( document != null(
}
       document.close;()
{
{
{

**/
@ *    param operator The operation to perform.
@ *    param operands The list of arguments.
*
@ *    throws IOException If there is an error processing the operation.
/*

@  Override
  protected void processOperator( Operator operator, List<COSBase> operands) throws
IOException
}
```

```
    String operation = operator.getName;()



    if( "Do".equals(operation( (
}
        COSName objectName = (COSName) operands.get( 0;(
 //      get the PDF object
        PDXObject xobject = getResources().getXObject( objectName;(
 //      check if the object is an image object
        if( xobject instanceof PDImageXObject(
}
          PDImageXObject image = (PDImageXObject)xobject;
           imageWidth.add(image.getWidth;(()
           imageHeight.add(image.getHeight;(()
           images.add(image;(

          Matrix ctmNew = getGraphicsState().getCurrentTransformationMatrix;()




          imageXScale.add(ctmNew.getScalingFactorX;(()
           imageYScale.add(ctmNew.getScalingFactorY;(()



           xLocationInPdf.add (ctmNew.getTranslateX;(()
```

```
        yLocationInPdf.add(ctmNew.getTranslateY;(()


 // */        position of image in the pdf in terms of user space units

       System.out.println("position in PDF = " + ctmNew.getTranslateX() + ", " +
ctmNew.getTranslateY() + " in user space units;("

 //        raw size in pixels

       System.out.println("raw image size = " + imageWidth + ", " + imageHeight + " in
pixels;("

 //        displayed size in user space units

       System.out.println("displayed size = " + imageXScale + ", " + imageYScale + " in user
space units/*;("

{

     else if(xobject instanceof PDFormXObject(

}

       PDFormXObject form = (PDFormXObject)xobject;

       showForm(form;(

{

{

   else

}

     super.processOperator( operator, operands;(

{

{


          public ArrayList<Float> getyLocationInPdf} ()

                    return yLocationInPdf;

          {
```

```
public ArrayList<Float> getxLocationInPdf} ()
        return xLocationInPdf;
{

public ArrayList<Float> getImageYScale} ()
        return imageYScale;
{

public ArrayList<Float> getImageXScale} ()
        return imageXScale;
{

public ArrayList<Integer> getImageHeight} ()
        return imageHeight;
{

public ArrayList<Integer> getImageWidth} ()
        return imageWidth;
{

public ArrayList<PDImageXObject> getImages} ()
        return images;
```

Abstract

Figures in scientific documents are rich source of information. Making a figure database can help better summarizing and information retrieving of the articles in search engines. To the best of our knowledge, there is no digital library search engine in Persian that can retrieve information from the figures in the documents. To this end, we develop a system for generating a figure database from scholarly Persian documents, in large scale.

The system that we suggest has different modules. In the first module the file is parsed and figures are extracted together with their text description**.** There are two general approaches for extracting figures from documents, one is based on image processing methods and another one is based on processing the file primitives. The focus of this paper is on later one since it is shown to be a better choice for the search engines because of its speed and scalability properties. We propose a structure based method that extracts the figures and their description by analyzing the file layout.

As a result of this step a set of images and their corresponding description are obtained. This information is saved in a database with a specific structure and is indexed for retrieval in search engine. We applied our method on sample documents from Iran scientific information database (Ganj).

Our algorithm is based on processing the Word file layout and is implemented in Python. As a benchmark we used the basic method in the literature which is based on the processing PDF file. We applied our method on 150 scientific documents, randomly chosen from Ganj dataset. Based on our experimental results, the proposed method is more efficient than the basic method especially when we are working with Persian documents. There are many unanswered challenges for Persian documents when using the basic method. The number of noise images resulted from the basic method is high and Persian text extracted is not well organized. Our proposed method overcomes some of these drawbacks and is recommended for generating figure database from scientific Persian documents.

Keywords: Image Processing, Image Extraction, Metadata Extraction, Information Technology.

Iranian Research Institute

for Information Science and Technology (Irandoc)

Research Project

An intelligent method for making figure database from Iranian scientific information database (Ganj)

By:

Azadeh Fakhrzadeh

and

Mojtaba Zali

December 2019