

بسم الله الرحمن الرحيم



وزارت علوم، تحقیقات و فناوری
پژوهشگاه علوم و فناوری اطلاعات ایران (ایرانداک)

طرح پژوهشی

**طراحی یک روش برای برچسب زدن تصاویر
استخراج شده از پارساهای موجود در پایگاه اطلاعات
علمی ایران (گنج)**

مجری

آزاده فخرزاده

همکار: امیرحسین صدیقی

تیر ۱۴۰۰

حقوق: پژوهشگاه علوم و فناوری اطلاعات ایران

طرح پژوهشی «طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج)» پیرو قرارداد شماره ۳۳/۱۰۰۰ مورخ ۱۳۹۹/۲/۱۰ میان پژوهشگاه علوم و فناوری اطلاعات ایران (کارفرما) و خانم آزاده فخرزاده اجرا شده است. گزارش حاضر یک جلد از مستندات این طرح است.

این گزارش و تمامی حقوق مادی آن بر اساس قانون حمایت حقوق مولفان و مصنفان و هنرمندان، مصوب ۱۳۴۸ و اصلاحیه‌های بعدی آن و همچنین آیین‌نامه‌های اجرایی این قانون متعلق به پژوهشگاه علوم و فناوری اطلاعات ایران و هرگونه استفاده از تمامی یا پاره‌ای از آن، شامل: نقل قول، تکثیر، انتشار، کاربرد نتایج، تکمیل و مانند آنها به صورت چاپی، الکترونیکی یا وسایل دیگر فقط با اجازه کتبی پژوهشگاه امکان‌پذیر است. نقل قول در حد هزار واژه در انتشارات علمی مانند کتاب و مقاله با درج اطلاعات کامل کتابشناختی، نیازی به مجوز پژوهشگاه ندارد. صحت مندرجات گزارش بر عهده مجری طرح پژوهشی است.



به نام خدا

ایرانداک در نیم‌قرن دوم کار خود همچنان جوان و کوشا، همراه هر پژوهش، پژوهشگر، و سیاست‌گذار

۱۳۴۷-۱۴۰۰

گواهی

انجام طرح پژوهشی:

- ◇ عنوان: طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج)
 - ◇ مجری: دکتر آزاده فخرزاده
 - ◇ همکار: دکتر امیرحسین صدیقی
 - ◇ شماره قرارداد: ۳۳/۱۱۵۲
 - ◇ تاریخ قرارداد: ۱۳۹۹/۲/۱۵
 - ◇ تاریخ شروع: ۱۳۹۹/۲/۱۵
 - ◇ تاریخ پایان: ۱۴۰۰/۳/۱۶
 - ◇ ناظر: دکتر نصراله پاک‌نیت
- در پژوهشگاه علوم و فناوری اطلاعات ایران گواهی می‌شود. گزارش پایانی این طرح، بر پایه داوری در نشست ۴۱۹ (تاریخ ۱۴۰۰/۵/۵) شورای پژوهشی پژوهشگاه به تصویب رسیده است. لازم می‌دانم مراتب قدردانی خود را به مجری، همکار، و ناظر این طرح تقدیم دارم.

با آرزوی بهروزی
برگزین شهریاری
معاون پژوهش و آموزش

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج)

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: ۰۲۱-۶۶۴۹۴۹۸۰ (داخلی ۴۶۱)
وبگاه <https://irandoc.ac.ir/u/a-fakhrzadeh>
رایانامه: Fakhrzadeh@irandoc.ac.ir
همکار(ان) اصلی: امیرحسین صدیقی، پژوهشگاه علوم و فناوری اطلاعات ایران.

چکیده

در این طرح پژوهشی یک روش جدید برای برچسب‌گذاری تصاویر موجود در متون علمی فارسی معرفی می‌شود. در اسناد 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 پی برد بدون اینکه لازم باشد کل مقاله را مطالعه کرد. به دلیل رشد روز افزون داده‌های تصویری، بسیاری از موتورهای جستجوگر مستندات علمی، به دنبال فراهم کردن امکان بازیابی تصاویر از پایگاه اطلاعاتی خود هستند، به طوری که کاربر با وارد کردن یک جستجو، علاوه بر متن مقالات بتواند به تصاویر مربوط به آن جستجو نیز دسترسی پیدا کند. هم اکنون در پایگاه اطلاعاتی گنج، که حاوی حجم زیادی از مستندات علمی و پایان‌نامه‌ها و رساله‌های فارسی کشور است، امکان جستجو بر اساس یک عبارت متنی و بازیابی و نمایش نتایج جستجو در قالب فراداده‌های متنی (عنوان، چکیده، پدیدآور، سال انتشار) وجود دارد. لیکن تصاویر موجود در اسناد «گنج» بازیابی نمی‌شود. فراهم کردن چنین امکانی در «گنج» به عنوان یک ارزش افزوده می‌تواند آن را از موتورهای جستجوی مشابه متمایز سازد. برای بازیابی تصاویر نیاز است که برچسب‌های توصیف‌کننده‌ای برای هر تصویر استخراج شود. موتور جستجو با استفاده از این برچسب‌ها می‌تواند تصاویری را بازیابی کند که کلمات جستجو شده در لیست برچسب‌های آنها وجود دارد. در اینجا برای استخراج برچسب تصاویر از متن سندی که تصویر به آن تعلق دارد استفاده شده است. زیرنویس و قسمتی از متن سند که در آن، به تصویر مورد نظر اشاره شده است در نظر گرفته می‌شود. عبارات اسمی در متن با استفاده از پنج روش متفاوت؛ فراوانی عبارات در سند، معکوس فراوانی سند، فراوانی کلمه- معکوس فراوانی سند، شباهت کسینوسی عبارات با زیرنویس و ترکیب روش فراوانی کلمه- معکوس فراوانی سند و شباهت کسینوسی با زیرنویس، رتبه‌بندی می‌شوند. برچسب‌های انتخابی برای تصویر در هر روش، عبارات اسمی با رتبه بالاتر در آن روش است. روش‌های معرفی شده با استفاده از داده آزمایشی از گنج، ارزیابی می‌شوند. طبق نتایج بدست آمده بر روی داده آزمایشی در این تحقیق، روش فراوانی کلمه- معکوس فراوانی سند با معیار F_1 ، ۳۸٪ بهترین روش برای برچسب زدن تصاویر و پس از آن ترکیب روش فراوانی کلمه- معکوس فراوانی سند و شباهت کسینوسی با زیرنویس با معیار F_1 ، ۳۶٪ قرار دارد. از آنجاییکه در مواردی زیرنویس تصاویر توصیف کاملی از تصاویر نمی‌دهد، با در نظر گرفتن شباهت کسینوسی با زیرنویس تصاویر، معیار F_1 کاهش می‌یابد.

کلیدواژه‌ها: برچسب زدن تصویر، متن کاوی، استخراج کلیدواژه، پردازش زبان طبیعی

فهرست مطالب

عنوان	شماره صفحه
چکیده..... ب	
فصل اول: مقدمه و اهداف..... ۲	
۱-۱- مقدمه..... ۲	
۱-۲- اهداف طرح..... ۳	
۱-۳- روش پژوهش..... ۳	
۱-۴- پیوند با اساس نامه و برنامه استراتژیک..... ۴	
فصل دوم: پیشینه پژوهش و پیش نیازها..... ۵	
۲-۱- مقدمه..... ۶	
۲-۲- روش های استخراج کلمات کلیدی یک متن..... ۱۰	
۲-۳- درون سازی یا تعبیه کلمه..... ۱۲	
۲-۳-۱- بررسی روش آماری tf-idf..... ۱۲	
۲-۳-۲- مدل N-grams..... ۱۳	
۲-۳-۳- مدل word2vec..... ۱۵	
۲-۴- جمع بندی..... ۱۷	
فصل سوم: روش پیشنهادی..... ۲۰	
۳-۱- مقدمه..... ۲۱	
۳-۲- روش پیشنهادی..... ۲۲	

۲۳	۳-۲-۱- پیش پردازش.....
۲۲	۳-۲-۲- استخراج عبارات اسمی.....
۲۳	۳-۲-۳- استخراج مدل های کلمات از پیکره.....
۲۴	۳-۲-۵- استخراج برچسب ها.....
۲۸	فصل چهارم: آماده سازی داده.....
۲۸	۴-۱- آماده سازی داده آزمایشی.....
۲۸	۴-۲- پیکره.....
۲۹	۴-۳- داده های آزمایشی.....
۳۴	فصل پنجم: پیاده سازی و ارزیابی نتایج.....
۳۴	۵-۱- مقدمه.....
۳۴	۵-۲- پیاده سازی.....
۳۹	۵-۳- معیارهای ارزیابی.....
۴۰	۵-۴- بررسی نتایج.....
۴۸	فصل ششم: نتیجه گیری و پیشنهادات آتی.....
۴۸	منابع.....

فهرست جداول

عنوان	شماره صفحه
جدول ۳-۱ نمونه‌ای از یک متن ورودی و متن پیش‌پردازش شده	۲۴
جدول ۵-۱ مقایسه ریشه‌یابی هضم و پارسیور	۳۵
جدول ۵-۲ مقایسه پنج روش پیشنهادی	۴۲

فهرست شکل ها

عنوان	شماره صفحه
شکل ۱-۲) بلوک دیاگرام اختصاص یک ردیف به هر شکل در ویکیپدیا (Josi et al. 2018).....	۱۰
شکل ۲-۲) شبکه عصبی CBOW در چپ و شبکه عصبی Skip-gram در راست نشان داده شده است.....	۱۶
شکل ۱-۳) روش پیشنهادی برای استخراج برجسب تصویر.....	۲۲
شکل ۱-۴) تقسیم بندی موضوعی پیکره.....	۳۰
شکل ۲-۴) تقسیم بندی موضوعی فایل هایی که تصویر از آنها استخراج شده است.....	۳۱
شکل ۳-۴) نمونه ای از داده های آزمایشی (ضیائی ۱۳۹۳).....	۳۲
شکل ۴-۴) نمونه ای از داده های آزمایشی (حاتمی ۱۳۹۱).....	۳۲
شکل ۵-۴) نمونه ای از داده های آزمایشی (مسعود ۱۳۹۲).....	۳۳
شکل ۱-۵) میانگین مقادیر دقت (آبی)، بازخوانی (قرمز) و معیار F_1 برای ۲۰ برجسب برتر در ۵ روش معرفی شده:	
الف) tfidf ب) idf پ) tfidf ت) شباهت کسینوسی ث) tf-idf و شباهت کسینوسی.....	۳۹
شکل ۲-۵) مقایسه سه روش برتر استخراج برجسب. عبارات درشت نمایی شده در هر روش توسط خبرگان هم	
انتخاب شده است. (حاتمی ۱۳۹۱).....	۴۴
شکل ۳-۵) مقایسه سه روش برتر استخراج برجسب. عبارات درشت نمایی شده در هر روش توسط خبرگان هم	
انتخاب شده است. (اسلامی ۱۳۹۱).....	۴۵
شکل ۴-۵) مقایسه سه روش برتر استخراج برجسب. عبارات درشت نمایی شده در هر روش توسط خبرگان هم	
انتخاب شده است. (افیونی اکبری ۱۳۹۱).....	۴۶

فصل اول

مقدمه و اهداف

۱-۱- مقدمه

محققان در مقالات علمی از نمودارها برای خلاصه کردن نتایج و از فلوچارت‌ها برای بیان گام‌های الگوریتم خود استفاده می‌کنند. دسترسی به این عناصر از یک سند علمی در بسیاری از موارد می‌تواند دورنمایی از ایده مقاله و نتایج کلی آن ارائه کند. برای بازیابی اطلاعات از تصاویر موجود در اسناد، ابتدا باید یک پایگاه داده از تصاویر ایجاد کرد. در بعضی از مجلات علمی از کاربر می‌خواهند که تصاویر را با زیرنویس آنها به صورت جداگانه ارسال کنند. اما در گنج چنین مکانیزمی وجود ندارد. فخرزاده و زالی (فخرزاده و زالی ۱۳۹۸) سیستمی طراحی کردند که می‌تواند از فایل‌های ورد اسناد گنج که فرمت استاندارد دارند، به طور خودکار تصاویر را به همراه زیرنویس آنها استخراج کند و یک پایگاه داده از این تصاویر ایجاد کند. برای اینکه تصاویر موجود در این پایگاه داده بتوانند به طور موثر بازیابی شوند، نیاز است برای هر تصویر در پایگاه اطلاعات تصاویر تعدادی برچسب که توصیف‌کننده آن باشد در نظر گرفته شود.

بررسی دقیق روش‌های برچسب زدن تصاویر و ارائه یک الگوریتم برای استخراج برچسب مناسب برای تصاویر موجود در پایگاه داده گنج از اهداف این طرح می‌باشد. برچسب زدن تصاویر به صورت خودکار از موضوعات جذاب حوزه بازیابی تصاویر است (Mayhew et al. Murthy et al. 2015, 2015, Yang et al. 2015, Wu et. al. 2015; Sohmen et al. 2018; Jobin et al. 2019; Morris et al. 2020). هدف نهایی تحقیقات در این زمینه، ایجاد یک سیستم است که با استفاده از محتوای خود تصویر و روش‌های پردازش تصویر بتواند کلمات توصیف‌کننده آن را استخراج کنند. داده‌های بررسی شده در روش‌های فوق تصاویر طبیعی است. داده‌های موجود در گنج اسناد علمی می‌باشند و تصاویر مورد نظر در این تحقیق تصاویر موجود در اسناد علمی است. این نوع تصاویر پیچیدگی‌های بیشتری نسبت به تصاویر طبیعی دارند. به عنوان مثال تصاویر یک سند علمی ممکن است همه از نوع نمودار خطی باشند که اگرچه از نظر ظاهر بسیار شبیه هستند اما موضوعات متفاوتی را نشان می‌دهند و باید برچسب‌های متفاوتی داشته باشند. از آنجایی که ما در اینجا علاوه بر خود

تصویر به سندی که تصویر را شامل می‌شود هم دسترسی داریم، می‌توانیم از متن سند برای استخراج برچسب تصویر استفاده کنیم. برای این منظور زیرنویس و پاراگراف‌هایی از متن که در آن به تصویر اشاره شده است را در نظر می‌گیریم. سپس با استفاده از روش‌های استخراج عبارت‌های کلیدی در پردازش زبان طبیعی برچسب مناسب تصویر را استخراج می‌کنیم. البته این روش‌ها هم چالش‌های خود را دارد. اینکه کلمات کلیدی زیرنویس و پاراگرافی که به تصویر اشاره می‌کند چقدر می‌تواند تصویر مورد نظر را توصیف کند پرسشی است که در این طرح به دنبال پاسخ آن هستیم.

۱-۲-اهداف طرح

هدف نهایی این طرح، طراحی یک روش برای برچسب‌گذاری تصاویر استخراج شده از متون علمی گنج است. از این برچسب‌ها برای بازیابی تصاویر استفاده می‌شود. اهداف فرعی این طرح به شرح زیر است:

- بررسی روش‌های موجود برای استخراج کلمات کلیدی از متن و به کار بردن آنها در توضیحات متنی تصویر.
- استخراج برچسب‌های توصیف‌کننده تصویر با تحلیل و بررسی قسمتی از متن سند که به تصویر اشاره می‌کند. پرسش‌ها/فرضیه‌هایی که برای رسیدن به هدف‌های طرح پژوهشی باید پاسخ داده یا بررسی شوند به شرح زیر است:
- روش‌های موجود در استخراج کلمات کلیدی از متن چیست؟
- آیا کلمات کلیدی موجود در زیرنویس تصویر و متنی که به تصویر اشاره می‌کند می‌تواند توصیفی مناسبی برای آن تصویر باشد؟

۱-۳-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، و خطای احتمالی آن بررسی گردد و در صورت نیاز بهبودهایی روی آن صورت گیرد. بنابراین، آماده‌سازی این

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۴

مجموعه داده برای انجام آزمایش و اندازه گیری دقت روش هم قسمتی از فعالیت های این پژوهش محسوب می شود.

گام های اصلی این پژوهش به شرح زیر است:

۱. مطالعه روش های موجود در پیشینه و آماده سازی داده آزمایشی

- بررسی روش های موجود در پیشینه برای استخراج برچسب تصاویر و چالش های موجود

- بررسی روش های استخراج کلمات کلیدی متن

- آماده سازی داده آزمایشی برای انجام آزمایشات

۲. طراحی یک روش برای برچسب زدن تصاویر استخراج شده از اسناد علمی موجود در گنج

- استخراج برچسب های تصویر با استفاده از زیرنویس آن و متن سند مربوطه

- آزمایش و ارزیابی روش پیشنهادی روی مجموعه داده های آزمایشی

۴-۱- پیوند با اساس نامه 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 در حوزه اطلاعاتی جدید و تولید ارزش افزوده است.

فصل دوم

پیشینه پژوهش و پیش نیازها

۲-۱- مقدمه

نویسندگان مقالات علمی معمولاً از نمودار یا تصاویر برای انتقال نتایج کمی آزمایش‌های خود و یا فراهم کردن کمک بصری برای توضیح بهتر و یا خلاصه کردن روش پیشنهادی خود بهره می‌برند. 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 نموده است (Jeon et al. 2003; Barnard and Forsyth 2001; Wang et al. 2009; Murthy et al. 2015; Johnson et al. 2015; Yang et al. 2015, Wu et. al. 2015; Sohmen et al. 2018; Jobin et al. 2019; Morris et al. 2020). برای بازیابی نمایش تصاویر موجود در اسناد، ابتدا لازم است یک پایگاه داده از تصاویر و توضیح متنی آنها فراهم شود. سپس باید به هر تصویر برچسب‌هایی که توصیف‌کننده آن تصویر است، اختصاص داده شود. موتور جستجو از این کلمات برای بازیابی تصاویر استفاده می‌کند. موضوع برچسب‌گذاری خودکار تصاویر، از موضوعات جذابی است که اخیراً به آن توجه زیادی شده است. اکثر روش‌های موجود در این زمینه به دنبال استخراج برچسب‌های مناسب با توجه به محتوای خود تصویر هستند (Barnard and Forsyth 2001; Jeon et al. 2003; Wang et al. 2009). در این تحقیقات سعی بر آن است که با استفاده از ویژگی‌های خود تصویر، الگویی را پیدا کنند که ویژگی‌های بصری را به برچسب‌های از پیش تعیین شده توسط خبرگان، مربوط کند. این الگو می‌تواند برای برچسب‌گذاری تصاویر جدید استفاده شود. با به وجود آمدن داده‌های آزمایشی عظیم روش‌های مبتنی بر یادگیری عمیق در این زمینه پیشرفت‌های زیادی داشته‌اند (Murthy et al. 2015; Mayhew et al. 2015; Johanson et al. 2015; Yang et al. 2015; Wu et. al. 2015). اکثر این روش‌ها در تصاویری

که شامل یک شیء برجسته^۱ هستند، کارایی خوبی دارند. اما در تصاویر موجود در مقالات علمی که در آن تصاویر با ساختار بسیار مشابه، مفاهیم کاملاً متفاوتی را انتقال می‌دهند عملکرد ضعیفی دارند. تحقیقات محدودی انجام شده است که در آن‌ها از محتوای تصویر استفاده می‌کنند تا گروه تصویر (مثلاً تصاویر طبیعی، نمودار دایره‌ای، نمودار خطی، جدول‌ها، نمودار پراکندگی و غیره) را مشخص کنند و آن را به عنوان یکی از برجسب‌های تصویر در نظر می‌گیرند (Savva et al. 2011; Tang et al. 2016; Siegel et al. 2016; Zhou and Tan 2000; Prasad et al 2007). البته همین گروه‌بندی کلی تصاویر هم به دلیل تنوع زیاد تصاویر موجود در یک گروه و همچنین شباهت یک گروه به گروه دیگر کار پیچیده‌ای است. اخیراً جوین و همکاران (Jobin et al. 2019) بیست و هشت گروه برای تصاویر در مقالات علمی معرفی کرده‌اند، و با استفاده از روش‌های یادگیری عمیق با دقت خوبی توانسته‌اند تصاویر را تقسیم‌بندی کنند. تا آنجایی که نویسندگان این تحقیق اطلاع دارند، روش معتبری جهت برجسب‌گذاری تصاویر علمی با استفاده از محتوای خود تصویر (فراتر از گروه تصویر)، وجود ندارد.

در خیلی از موارد، تصویر در یک سند همراه با یک متن توصیفی می‌آید. لانگ و همکاران (Leong et al. 2010) با استفاده از زیرنویس و توضیح متنی تصاویر و به کارگیری تکنیک‌های پردازش زبان طبیعی، کلمات توصیف‌کننده تصاویر را استخراج کرده‌اند. آن‌ها از دو مجموعه داده استفاده کرده‌اند و سه روش متفاوت برای رتبه‌بندی کلمات در زیرنویس و متن مربوط به تصویر را در نظر گرفته‌اند. در یکی از روش‌ها از برجسب‌های موجود در انباره فلیکر^۲ استفاده کرده‌اند. در این روش، رتبه هر کلمه در متن مربوط به تصویر با توجه به تکرار آن کلمه در انباره فلیکر تعیین می‌شود. در روش بعدی از داده‌های موجود در ویکی‌پدیا و روش Textrank (Mihalcea and Tarau, 2006) استفاده شده است. روش Textrank یک روش استخراج کلمات کلیدی بر اساس گراف^۳ است. در این روش، هر کلمه در زیرنویس بر اساس تحلیل معنایی صریح^۴ تبدیل به یک بردار می‌شود و به عنوان راس گراف در نظر گرفته می‌شود سپس به بقیه کلمات در زیرنویس با یک یال جهت‌دار وصل می‌شود. وزن هر یال با توجه به شباهت استنباطی^۵ هر کلمه با کلمه بعدی تعیین می‌شود. وزن هر یال با نسبت

¹ Salient object

² www.flickr.com

³ Graph-based keyword extraction

⁴ Explicit semantic analysis

⁵ inferential similarity

تعداد هم رخدادی^۱ هر کلمه با کلمه بعدی به تعداد تکرار آن کلمه، در مقالات ویکی پدیا نرمال می شود. سپس با استفاده از وزن های محاسبه شده^۲ رتبه هر کلمه تعیین می شود. در روش سوم، از روش مدل کردن موضوع متن استفاده کرده اند (McCallum and Li 2006). بدین ترتیب رتبه هر کلمه در زیرنویس با توجه به احتمال تعلق آن به مدل های موضوعی پیدا شده، تعیین می شود. رتبه نهایی هر برچسب (هر کلمه در زیرنویس) از ترکیب خطی رتبه آن در روش های معرفی شده به دست می آید. وزن رتبه هر روش در این ترکیب خطی با استفاده از داده آموزشی و رگرسیون تابع بازخوانی (یا دقت) براساس روش دستگاه بردار پشتیبان محاسبه می شود. ماسون و چارنیاک (Mason and Charniak 2012)، با استفاده از داده های لانگ و همکاران (Leong et al. 2010) و داده های فنگ و لاپاتا (Feng and Lapata 2008) که شامل تصاویر و زیرنویس آنها است، نشان داده اند که روش های آماری ساده مثل فراوانی عبارت^۳، ضریب فراوانی و معکوس فراوانی سند^۴، فراوانی در پیکره، در نظر گرفتن کلمات سرنسخه اصلی و جمله اشاره شده به تصویر در متن همراه تصویر، نتایج قابل قبولی در مقایسه با روش های پیچیده تر به دست می دهند.

می توان از دو مجموعه داده، کلمات موجود در متن و ویژگی های تصویری، برای استخراج برچسب تصویر استفاده کرد. از روش های اصلی برای پیدا کردن یک مدل که اطلاعات موجود در زیرنویس را به اطلاعات تصویر مربوط می کند و بر این اساس برچسب توصیف کننده برای تصویر استخراج می کند، روش تخصیص پنهان دیریکله^۵ (Blei et al. 2003) است. تخصیص پنهان دیریکله، یک روش آماری است که در آن هر مدرک در پیکره بر اساس تعداد محدودی موضوع خلاصه می شود. در این روش کیسه ای از کلمات موجود در مدرک تشکیل می شود. تعداد N موضوع در نظر گرفته می شود و میزان نزدیکی هر کلمه به هر یک از این موضوعات تخمین زده می شود. بعد از تعیین احتمال تعلق کلمات یک سند به هر موضوع، می توان احتمال نزدیکی هر یک از N موضوع را به سند تخمین زد. برای استفاده از مدل تخصیص پنهان دیریکله برای برچسب زدن تصویر، ابتدا با استفاده از یکی از روش های تبدیل ویژگی مقیاس نابسته (SIFT) در پردازش تصویر، بردارهایی از تصویر که توصیف کننده ویژگی های موضعی تصویر است، تولید می شود. این بردارها به عنوان کلمات تصویری شناخته می شود. در این اینجا دو مجموعه اطلاعات داریم، مجموعه اول مربوط به کلمات زیرنویس است و مجموعه دوم مربوط به تصاویر مربوطه است. جهت اختصاص دادن یک موضوع به

¹ cooccurrence

² Graph iteration

³ Term frequency

⁴ Inverse document frequency

⁵ Latent Dirichlet Allocation

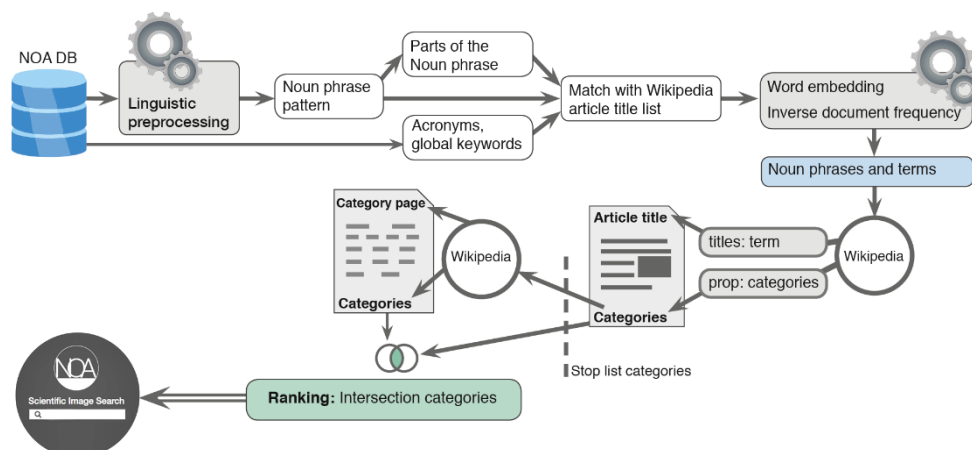
تصویر، یک چارچوب احتمالاتی این است که فرض کنیم تعداد محدودی از متغیرهای پنهان وجود دارند که همبستگی بین دو مجموعه داده را ایجاد می‌کند. در (Blei and Jordan 2003) فرض بر آن است که نسبت نزدیکی دو مجموعه داده، داده‌های مربوط به زیرنویس و تصویر، به هر موضوعی یکسان باشد. مشکل این نگاه این است که بردارهای معادل یک تصویر ویژگی‌های موضعی را توصیف می‌کند در حالیکه زیرنویس تصویر ممکن است یک توصیف کلی از تصویر داشته باشد و در نتیجه سهم یکسانی از یک موضوع نداشته باشند. پوتهیدی و همکاران (Putthividhy et al. 2010) یک مدل برگشت‌پذیر معرفی کرده‌اند که برای داده‌های تصویر و زیرنویس، به جای یک موضوع، دو موضوع که با هم همبستگی داشته باشند را استخراج می‌کند.

جوزی و همکاران (Josi et al. 2018) برای استخراج زیرنویس تصاویر از سند مربوط به تصویر استفاده کرده‌اند. در شکل ۱-۲ بلوک دیاگرام روش آنها مشاهده می‌شود. مقالات، تصاویر، زیرنویس آنها و قسمتی از متن که به تصویر اشاره شده است به عنوان ورودی به سیستم داده می‌شود. ابتدا با استفاده از پردازش‌های ابتدایی زبان‌شناختی عبارات اسمی از زیرنویس و متن اشاره شده به تصویر استخراج می‌شود. سپس در عنوان‌های مقالات ویکی‌پدیا این عبارات اسمی جستجو می‌شوند. اگر یک عبارت اسمی در ویکی‌پدیا پیدا نشود کلمه ابتدایی آن عبارت، حذف شده و ادامه آن در ویکی‌پدیا جستجو می‌شود. تا اینکه در نهایت عبارات اسمی معتبر انتخاب شود. با استفاده از روش معکوس فراوانی سند، عباراتی که معکوس فراوانی سند بالاتری دارند، در نظر گرفته می‌شوند. سپس با استفاده از روش تعبیه کلمه^۱، معادل برداری عبارت استخراج شده با معکوس فراوانی سند بالاتر، محاسبه می‌شود. عباراتی که معادل برداری آنها شباهت بیشتری به عبارات اسمی زیرنویس داشته باشند، به عنوان برجسب انتخاب می‌شوند. در نهایت با استفاده از اطلاعات فراداده مقاله، و برجسب‌های تصویر، برای هر تصویر یکمقوله^۲ از مقوله‌های ویکی‌پدیا اختصاص داده می‌شود (شکل ۱-۲).

¹ Word embedding

² Category

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۱۰



شکل ۲-۱) بلوک دیاگرام اختصاص یک ردیف به هر شکل در ویکی پدیا (Josi et al 2018)

در برچسب زدن تصویر انتخاب قسمت مناسبی از متنی که همراه تصویر است، نقش مهمی در دقت سیستم دارد. اطلاعات توصیف کننده تصویر معمولاً در نزدیکی تصویر قرار دارد، گاهی هم اطلاعات محتوایی تصاویر با هم همپوشانی دارند. چن و همکاران (Chan et al. 2013) پیشنهاد می کنند متنی را که برچسب مشترک با تصویر مورد نظر در HTML دارد و شامل برچسب است، در نظر گرفته شود.

با توجه به پیچیدگی های پردازش تصاویر در متون علمی، در این تحقیق از متن سند همراه تصویر برای برچسب زدن تصاویر استفاده می شود. در ادامه به روش های پایه پردازش متن، که شامل استخراج کلمات کلیدی متن و درون سازی کلمات است، می پردازیم. این روشها در روش پیشنهادی در فصل بعد به کار می رود.

۲-۲- روشهای استخراج کلمات کلیدی یک متن

کلمات کلیدی یک سند می تواند به عنوان خلاصه ای از آن سند، در سازماندهی و بازیابی اسناد به کار برود. به همین دلیل استخراج کلمات کلیدی به صورت خودکار، از مباحث مهم پردازش زبان های طبیعی است و تحقیقات زیادی در این زمینه انجام شده است. (Siddiqi and Sharan 2015; Biswas et al. 2018; Onan et al. 2016; Campos et al. 2020). صدیقی و شارن چهار گروه اصلی را برای استخراج کلمات کلیدی معرفی کرده اند:

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۱۱

۱) روش‌های زبان‌شناختی مبتنی بر دستور^۱: این روش‌ها عموماً مبتنی بر دستور و بر اساس ویژگی‌های زبان‌شناختی است. این روش‌ها بر اساس تحلیل واژگانی (Barzilay and Elhadad 1997، تحلیل نحوی (Hulth 2003) و تحلیل گفتمان (Salton et al. 1997) و غیره است. این روش‌ها، در خیلی از موارد دقت قابل قبولی دارند اما از نظر محاسباتی زمان‌بر هستند و نیاز به اطلاعات کافی از ساختار زبانی و دامنه زبان پیکره دارند.

۲) روش‌های آماری: این روش‌ها بر اساس اطلاعات آماری پیکره است. روش‌های آماری، روش‌های ساده‌ای هستند که نیازی به داده آموزشی ندارند. مزیت این روش‌ها، عدم وابستگی آنها به زبان داده و سرعت محاسبه پایین است. این روش‌ها شاید دقت روش‌های زبان‌شناختی را نداشته باشند اما در صورت دسترسی به یک پیکره عظیم نتایج خوبی به دست می‌دهند (Ramos 2003; Matsuo and Ishizuka 2004; Chien 1997).

۳) روش‌های مبتنی بر یادگیری ماشین: در این رویکرد از روش‌های یادگیری تحت نظارت استفاده می‌شود. در این روش‌ها با استفاده از یک داده آموزشی یک مدل برای استخراج کلمات کلیدی متن آموزش داده می‌شود. با استفاده از داده‌های آزمایشی مدل ارائه شده آزمایش می‌شود. روش‌های یادگیری با نظارت نیازمند یک داده آموزشی برچسب زده شده غنی است. فراهم کردن این مجموعه داده کار ساده‌ای نیست. در نبود داده آموزشی مکفی از روش‌های یادگیری بدون نظارت و یا نیمه نظارت شده استفاده می‌شود (Jo 2010; De Souza et al. 2009).

۴) روش‌هایی ترکیبی که مخصوص یک حوزه هستند. اطلاعات هستی‌شناسی و ساختار پیکره یک حوزه خاص، در نظر گرفته می‌شود و روش مناسب جهت استخراج کلمات کلیدی با ترکیب روش‌های بالا را طراحی می‌کنند.

در روش‌های مبتنی بر یادگیری ماشین و روش‌های آماری با استفاده از پیکره داده آموزشی، یک مدل عددی برای داده‌ها به دست می‌آید و از این مدل برای استخراج کلمات کلیدی متن استفاده می‌شود. در ادامه با چند روش محاسبه مدل عددی از کلمات آشنا می‌شویم.

¹ Rule Based Linguistic approaches

۳-۲- درون‌سازی یا تعبیه کلمه

لغت‌نامه‌های انسانی بصورت متن ساده هستند اما برای اینکه یک مدل یادگیری ماشینی قادر به فهم و پردازش زبان طبیعی باشد، ما نیازمند تبدیل کلمات از متن ساده به مقادیر عددی هستیم. یکی از ساده‌ترین روش‌های تبدیل سازی، انجام فرایند کد گذاری one-hot است که در آن هر کلمه دارای یک جایگاه خاص در فضای برداری حاصل است و مقادیر دودویی صفر و یک نشانگر عدم وجود یا وجود یک کلمه در یک بردار می‌باشند. کدگذاری one-hot در زمانی که با تمام کلمات لغت نامه مواجه باشیم از نظر محاسباتی سربار بسیار زیادی بر ما تحمیل خواهد کرد و در نتیجه فرایندی غیرعملی خواهد بود چرا که بازنمایی حاصل در اینجا نیازمند صدها هزار بُعد خواهد بود. ما نیازمند روشهایی هستیم که سند و کلمات را به بردارهای عددی با ابعاد کمتر نگاشته کند. تعبیه کلمه یا اصطلاحاً word embedding نامی تجمعی برای روشهایی است که با استفاده از تعبیه سازی ریاضی، کلمات از فضایی با ابعاد زیاد به فضای برداری پیوسته با ابعاد بسیار کمتر نگاشته می‌شوند. این روشها براساس روشهای آماری و شبکه‌های عصبی هستند.

۲-۳-۱- روش آماری tf-idf

این روش آماری، یک روش ساده و محبوب در پردازش متن است که هدف آن وزن دهی کلمات بر اساس اهمیت آنها در یک سند و یا پیکره است. tf مخفف term frequency و به معنای فراوانی یک کلمه است، idf مخفف Inverse document frequency و به معنای معکوس فراوانی سند است. tf-idf به صورت زیر تعریف می‌شود:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (\text{فرمول ۱-۲})$$

$$tf(t, d) = f_{t,d} \quad (\text{فرمول ۲-۲})$$

$$idf(t, D) = \log \frac{N}{|d \in D: t \in d|} \quad (\text{فرمول ۳-۲})$$

t کلمه مورد نظر، d سند مربوطه، $f_{t,d}$ فراوانی واژه t در سند d است. N تعداد کل اسناد در پیکره D است. معکوس فراوانی سند، لگاریتم معکوس کسر اسنادی است که حاوی کلمه است و نشان دهنده اطلاعاتی است که اون کلمه شامل می شود (Spärck 1972). خروجی این روش برای یک پیکره، یک ماتریس عددی است که در آن هر سند به صورت برداری از وزنهای tf-idf متناظر با کلماتش. مشخص می شود. این ماتریس می تواند به عنوان مدل عددی در رتبه بندی کردن کلمات، خلاصه سازی متن، و استخراج کلمات کلیدی بکار برود. در بررسی که کورینا و همکاران (Corinna et al. 2015) انجام داده اند، حدود هشتاد درصد از سیستمهای پیشنهاد دهنده از tf-idf استفاده می کنند.

در tf-idf همبستگی بین کلمات در نظر گرفته نمی شود و همچنین کلمات هم معنی، مستقل در نظر گرفته می شود. در روش نمایه سازی معنایی نهفته، با به کار بردن روش تجزیه مقدارهای منفرد، ترکیب خطی ای از ویژگی های tf-idf در راستایی که بیشترین پراکندگی را در کل پیکره دارند، بدست می آید (Bellegarda 2000). به این ترتیب کلمات هم معنی مستقل در نظر گرفته نمی شود. البته با این روش، فضای برداری جدیدی به دست می آید که در آن قابلیت تفسیرپذیری اثر هر کلمه در مراحل بعدی از بین می رود.

۲-۳-۲ مدل N-grams

در ساختن مدل های زبانی، ایده ال این است که کلمات را در قالب جملات مدل سازی کنیم. با دسترسی به چنین مدلی سیستمهای پیشنهاد دهنده می توانند با دانستن تعدادی از کلمات، جملات معنی دار را پیشبینی کنند. به عبارت دیگر یک مدل خوب باید بتواند با دانستن $w_1, w_2, w_3, \dots, w_m$ احتمال هر ترکیبی از این کلمات را محاسبه کند $p(w_1, w_2, w_3, \dots, w_m)$ و ترکیب محتملتر را ارائه دهد. ما می دانیم که هر کلمه در جمله وابسته به توالی کلمات قبل از آن است. بنابراین به جای محاسبه همه ترکیبات کلمات می توانیم $p(w_1|h)$ ، احتمال یک کلمه را با دانستن کلمات قبل از آن (h) ، را محاسبه کنیم. به عنوان مثال احتمال اینکه بعد از "بازیکنان تیم ملی"، کلمه (جوانان) بیاید را محاسبه می کنیم:

$$(فرمول ۲-۴) \quad p(\text{بازیکنان تیم ملی} | \text{جوانان})$$

یک راه تخمین زدن این احتمال ب شمردن فراوانی است. به این ترتیب که از بین همه "بازیکنان تیم ملی" که در پیکره آمده است، چه تعداد از آنها کلمه جوانان بعد از آن آمده است:

$$(فرمول ۲-۵) \quad p(\text{بازیکنان تیم ملی} | \text{جوانان}) = \frac{C(\text{بازیکنان تیم ملی جوانان})}{C(\text{بازیکنان تیم ملی})}$$

برای چنین تخمینی ما نیاز به یک داده عظیم داریم. ولی حتی در یک داده عظیم مثل وب، جملات جدیدی مرتب ساخته می شود و امکان شمردن همه جملات نیست. بر اساس تئوری مارکوف می توان احتمال آمدن یک کلمه در جمله بعد از تعداد شناخته ای از کلمات، را با احتمال آمدن آن کلمه را بعد از تعداد محدودی از کلمات قبل از آن تخمین زد. به این ترتیب احتمال بالا به صورت زیر تخمین زده می شود:

$$(فرمول ۲-۶) \quad p(\text{بازیکنان تیم ملی} | \text{جوانان}) = p(\text{ملی} | \text{جوانان})$$

به بیان دیگر احتمال اینکه کلمه جوانان بعد از "بازیکنان تیم ملی" بیاید با احتمال اینکه کلمه جوانان بعد از ملی بیاید تخمین زده می شود به این احتمال 2-gram یا Bigram گفته می شود و به صورت زیر محاسبه می شود:

$$(فرمول ۲-۷) \quad p(\text{ملی} | \text{جوانان}) = \frac{C(\text{ملی جوانان})}{C(\text{جوانان})}$$

تعداد ترکیبات ملی جوانان به تعداد کل جوانان در پیکره تقسیم می شود. مدل 1-gram یا unigram همان احتمال رخداد یک کلمه در سند است، در مدل 3-gram دو کلمه قبل را در احتمال شرطی در نظر می گیریم و در مدل های 4-gram و 5-gram به ترتیب چهار و پنج کلمه قبل را در نظر می گیریم.

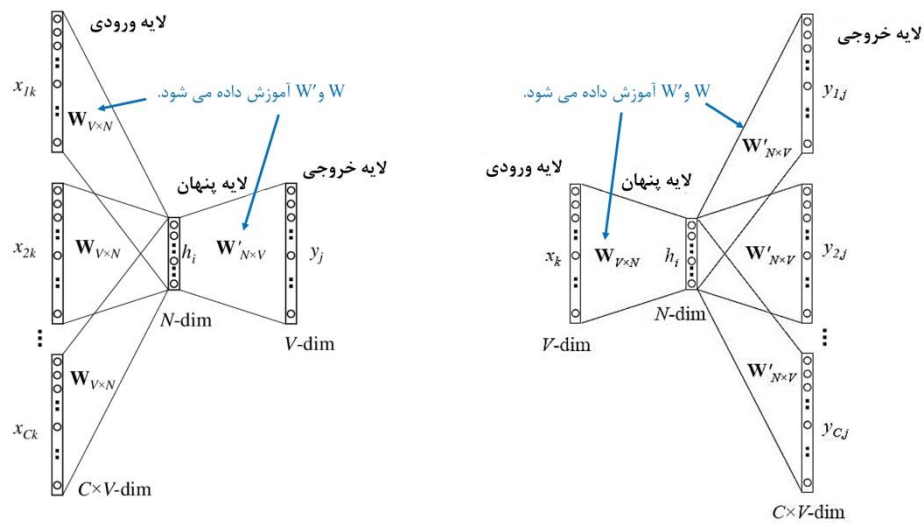
طراحی یک روش برای برجسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۱۵

روشهایی که معرفی شد روشهای ساده‌ای هستند و در زمانی که داده در دسترس اندازه کوچکی داشته باشد عملکرد بهتری نسبت به الگوریتمهای پیچیده دارند. یک مشکل عمده که در این روشها دیده می‌شود این است که شباهت معنایی و ظاهری بین کلمات در طراحی مدل در نظر گرفته نمی‌شود. یک فرض شهودی در رابطه با تعبیه سازی کلمه مناسب این است که آنها قادر به تشخیص کلمات هم معنی باشند مثلاً کلمات "خانه" و "منزل" کلماتی هم معنی هستند و در نتیجه انتظار می‌رود که در فضای برداری نیز بازنمایی های این دو کلمه نزدیک به یکدیگر باشند. همچنین یک مدل ایده آل می‌تواند روابط معنایی مخفی بین کلمات را آشکار ساری کنند. مثلاً رابطه بین "خانه" و "منزل" مشابه رابطه بین "ماشین" و "خودرو" باشد. میکولوف و همکاران (Mikolov et al. 2013) روشی را برای تعبیه سازی ارائه داده اند که محتوای کلمات هم در نظر گرفته می‌شود و مشکلات روشهای قبلی مرتفع شده است. آنها از دو مدل شبکه عصبی برای تعبیه سازی کلمات استفاده کرده‌اند که در زیر به آنها می‌پردازیم. در شکل ۲-۲ هم نمای کلی دو شبکه نشان داده شده است.

یک روش این است که مجموعه کلمات {پسر، بچه، با، در، پارک، بازی، می‌کرد} را به عنوان محتوا و یا ورودی در نظر گرفته می‌شود، و کلمه مرکزی "توپ" را پیشینی می‌شود. به مدلی که با این روش به دست می‌آید کیسه کلمات پیوسته^۱ (CBOW)، گفته می‌شود. ورودی الگوریتم، x^c است و خروجی y همان کلمه مرکزی است که می‌خواهیم پیشینی کنیم. ورودی و خروجی به صورت بردار one-hot هستند. دو ماتریس $v \in R^{|V| \times n}$ و $U \in R^{n \times |V|}$ را ایجاد می‌شود. n یک عدد دلخواه است که اندازه فضای تعبیه را مشخص می‌کند و $|V|$ تعداد کلمات موجود در پیکره است. v_i ستون i از ماتریس v ، بردار تعبیه n بعدی متناظر با کلمه ورودی w_i است. u_j ردیف j از ماتریس خروجی U ، بردار تعبیه n بعدی متناظر با کلمه مرکزی w_j است. خلاصه ای از الگوریتم در زیر آمده است.

¹ Continuous Bag of Words

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۱۶



شکل ۲-۲) شبکه عصبی CBOW در چپ و شبکه عصبی Skip-gram در راست نشان داده شده است.

- برای هر کلمه محتوا، در مجموعه کلمات محتوا با طول C ، یک بردار one-hot تولید می‌شود:

$$(x_{1k}, x_{2k}, x_{3k}, \dots, x_{Ck} \in R^{|V|}) \quad (\text{فرمول ۲-۸})$$

- بردارهای تعبیه را برای کلمات محتوا محاسبه می‌شود:

$$(v_{1k} = vx_{1k}, v_{2k} = vx_{2k}, \dots, v_{Ck} = vx_{Ck} \in R) \quad (\text{فرمول ۲-۹})$$

- میانگین بردارهای تعبیه به دست می‌آید:

$$\hat{v} = \frac{v_{1k} + v_{2k} + \dots + v_{Ck}}{C} \quad (\text{فرمول ۲-۱۰})$$

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۱۷

- بردار امتیازها محاسبه می‌شود، $z = U\hat{v} \in R^{|V|}$. از آنجایی که ضرب داخلی بردارهای مشابه بالاتر است این باعث می‌شود که کلمات مشابه نزدیک هم قرار بگیرند. تابع $softmax$ بردار امتیازها به احتمال می‌شود

$$\hat{y} = softmax(z) \in R^{|V|} \quad (\text{فرمول ۲-۱۱})$$

- مطلوب آن است که بردار احتمالاتی که توسط شبکه عصبی تولید می‌شود $\hat{y} \in R^{|V|}$ با بردار صحیح احتمال $y \in R^{|V|}$ تطابق داشته باشد. بردار صحیح احتمال همان بردار one-hot، معادل کلمه مرکزی است. تابع هدف با استفاده از آنتروپی متقاطع بین $\hat{y}$ و y به صورت زیر محاسبه می‌شود:

$$minimize J = -u_c^T \hat{v} + \log |V| \sum_{j=1}^{|V|} \exp(u_j^T \hat{v}) \quad (\text{فرمول ۲-۱۲})$$

با بهینه سازی تابع هدف با استفاده از داده آزمایشی بردارهای تعبیه کلمه مرکزی و کلمات محتوا به دست می‌آید.

روش دیگر $skip-gram$ است که در آن کلمه مرکزی مثل "توپ" را بعنوان ورودی شبکه در نظر بگیریم و کلمات اطراف آن را به عنوان خروجی $\{ \text{پسر، بچه، با، در، پارک، بازی، می‌کرد} \}$ پیش بینی کنیم. در این شبکه بردار امتیازها به صورت $z = Uv_c \in R^{|V|}$ محاسبه می‌شود و خروجی به صورت $\hat{y}_c, \dots, \hat{y}_{c+1}, \dots, \hat{y}_{c-1}, \dots, \hat{y}_{c-m}$ است که در واقع احتمال پیش بینی شده برای کلمات اطراف کلمه مرکزی است. تابع هدف در این مدل به صورت زیر محاسبه می‌شود.

$$minimize J = - \sum_{j=0, j \neq m}^{2m} u_{c-m}^T v_c + 2m \log \sum_{k=1}^{|V|} \exp(u_k^T v_c) \quad (\text{فرمول ۲-۱۳})$$

در این فصل ابتدا مروری بر ادبیات در زمینه برچسب زدن تصاویر با استفاده از روشهای پردازش زبان طبیعی صورت گرفت. سپس روشهای پایه پردازش زبان طبیعی که در استخراج کلمات کلیدی و برچسب زدن تصاویر کاربرد دارند معرفی شدند. روشهای معرفی شده برای برچسب زدن تصاویر با استفاده از متن همراه تصاویر در زبان انگلیسی کاربرد دارند و به دلیل تفاوت ساختاری زبان فارسی با انگلیسی، نمی‌توان مستقیماً در زبان فارسی به کار برد. زبان فارسی یکی از زبان‌های هندواروپایی است که توسط بیش از صد میلیون نفر از مردم جهان صحبت شده و بعنوان زبان رسمی سه کشور ایران، افغانستان (فارسی دری) و تاجیکستان (فارسی تاجیکی) است. بدلیل پیچیدگی‌های زبانی، کمبود منابع و مطالعات انجام شده در این زبان، از دیدگاه محاسباتی کمتر مورد توجه پژوهشگران قرار گرفته است. کاراکترهای یکسان با یونیکدهای گوناگون (مانند انواع میم)، وجود کاراکترهای بدون تاثیر در محتوا (مانند کاراکترهای کشسانی)، عدم یکسان نویسی فاصله و نیم فاصله و علائم نگارشی، کلمات چند دیگته‌ای، تفاوت ساختار زبان فارسی با لاتین، کلمات جمع مکسر، ریشه‌یابی پیچیده کلمات با توجه به ساختار زبان فارسی، نمونه‌هایی از چالش‌های پردازش زبان فارسی است که برای متون علمی دوچندان می‌شود. تا آنجایی که نویسندگان این مقاله اطلاع دارند تحقیقی در زمینه برچسب زدن تصاویر موجود در متون علمی فارسی صورت نگرفته است. در این تحقیق روشی برای برچسب زدن تصاویر در اسناد علمی معرفی می‌شود.

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۱۹

فصل سوم

روش پیشنهادی

۳-۱- مقدمه

در این فصل با استفاده از روش مطالعات اسنادی، سیستمی برای استخراج برچسب تصاویر موجود در اسناد علمی فارسی در گنج طراحی شده است. در «گنج» امکان جست و جو بر اساس یک عبارت متنی پرس و جو و بازیابی و نمایش نتایج جست و جو در قالب فراداده‌های متنی (عنوان، چکیده، کلیدواژه، پدیدآور، سال انتشار) وجود دارد. لیکن تصاویر موجود در اسناد «گنج» بازیابی نمی‌شود. فراهم کردن چنین امکانی در «گنج» به عنوان یک ارزش افزوده می‌تواند آن را از موتورهای جست و جوی مشابه متمایز سازد.

اولین قدم در بازیابی تصاویر ایجاد یک پایگاه داده از تصاویر موجود در گنج است. فخرزاده و صدیقی (فخرزاده و صدیقی، ۱۳۹۹) روشی ساختارمحور که مبتنی بر چیدمان و آرایش فایل وُرد سند است، برای ایجاد پایگاه داده از تصاویر مستخرج از اسناد علمی موجود در گنج ارائه داده اند. نرم افزار معرفی شده توسط آنها، قادر است تصاویر و زیرنویس آنها و قسمتی از متن سند را که شامل

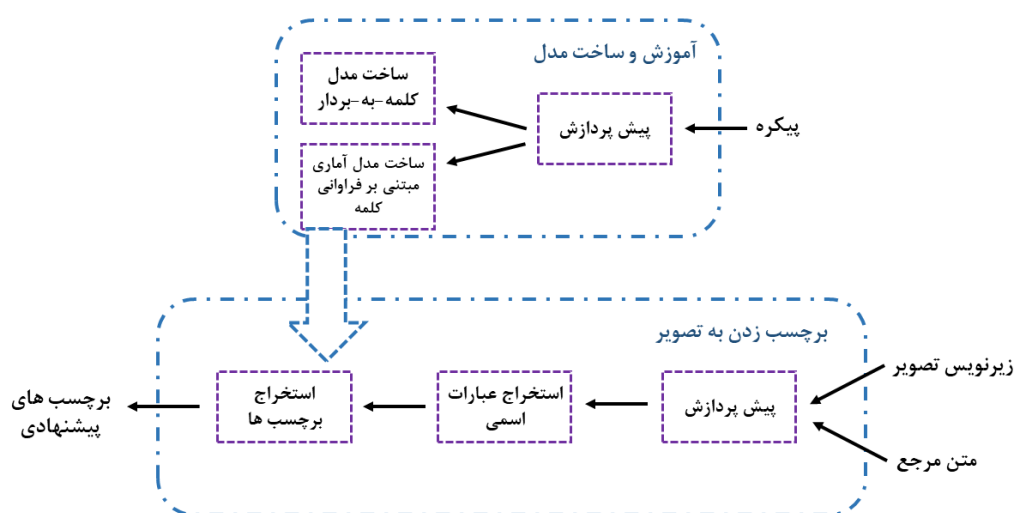
طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۲۲

اشاره به تصویر می‌شود، را استخراج کند. قدم بعدی در بازیابی تصاویر، استخراج برچسبهای توصیف کننده برای تصاویر است. . موتور جستجو از این کلمات برای بازیابی تصاویر استفاده می‌کند.

در این فصل، روش‌های استخراج خودکار برچسب تصاویر موجود در اسناد علمی فارسی، بر اساس پردازش متن همراه تصویر معرفی می‌شود. در فصل پنجم روشهای معرفی شده ارزیابی می‌شوند بهترین روش معرفی می‌شود.

۲-۳- روش پیشنهادی

شکل ۱-۳، واحدهای مختلف سیستم طراحی شده را نشان می‌دهد. این سیستم با تجزیه و تحلیل متن همراه تصویر با استفاده از روش‌های پردازش زبان طبیعی، برچسب‌های توصیف کننده تصویر را استخراج می‌کند. روش پیشنهادی از دو بخش تشکیل شده است. در بخش اول با استفاده از پیکره، مدل‌های آماری و کلمه-به-بردار ساخته می‌شود. در بخش دوم، متن مرجع که شامل اشاره به تصویر است به همراه زیرنویس تصویر تحلیل می‌شود و با استفاده از مدل‌های ساخته شده در بخش اول، برچسب‌های تصویر پیشنهاد می‌شود. جزئیات روش پیشنهادی در ادامه می‌آید.



شکل ۱-۳. روش پیشنهادی برای استخراج برچسب تصویر

۳-۲-۱- پیش پردازش

در مرحله اول نرمال سازی یا یکسان سازی انجام می شود. هدف از نرمال سازی یکسان کردن کلمات موجود در متون است. از آنجایی که اسناد مختلف از کدگذاری های مختلف استفاده می کنند، کلماتی که از نظر نوشتاری تفاوت دارند اما در واقع یک کلمه هستند با نرمال سازی یکسان می شوند در مرحله بعد کلمات ریشه یابی می شوند. در اینجا از یک ریشه یابی ساده استفاده شده است:

حذف شناسه های فعل فارسی با در نظر گرفتن زمان فعل

حذف پیشوندهای فعل فارسی. مانند: «می»، «ب» و ...

حذف پسوندهای مالکیت: «م»، «ت»، «ش»، «یم»، «یت»، «یش»، «یتان»، «یشان»، «یمان»، «مان»، «تان»، «شان» و «ان».

حذف نشانه های جمع فارسی و عربی: «ها»، «ات»، «های»، «ان»، «هایی» و «ین».

حذف «ی میانجی»

پس از آن با استفاده از واحد ساز، متن را به واحدهای تشکیل دهنده آن یعنی کلمات تبدیل می شود. مراحل پیش پردازش روی تمام متون موجود در پیکره اعمال می شود. در جدول ۳-۱ نمونه ای از متن همراه یک تصویر و متن پردازش شده پس از اعمال نرمال سازی و ریشه یابی مشاهده می شود.

۳-۲-۲- استخراج عبارات اسمی

پس از پیرایش اولیه، عبارات اسمی یک کلمه ای (unigram)، دو کلمه ای (bigram) و سه کلمه ای (trigram) متن تمام اسناد موجود در پیکره استخراج می شود. در متن اسناد کلمات رایجی وجود دارند که بار معنایی کمی دارند. این کلمات ایست واژه نامیده می شوند. ایست واژه های فارسی فهرستی از افعال فارسی، مصدر، حروف ربط، حروف اضافه و انواع قیده های زبان فارسی هستند. علاوه بر آن کلمات رایجی که ممکن است در توضیحات تصاویر ذکر شوند به این فهرست اضافه شده اند. واژه هایی چون «شکل»، «نقشه»، «نمودار»، «تصویر» و یا لغاتی که در حوزه مورد بررسی رایج

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۲۴

تلفی می‌شوند همچون کلمات «مگا پاسکال» یا «نرم افزار متلب» به این فهرست افزوده شده‌اند. عبارات اسمی که شامل ایست واژه‌ها هستند حذف می‌شوند.

جدول ۳-۱ نمونه‌ای از یک متن ورودی و متن پیش پردازش شده

متن ورودی	پیش پردازش
شکل 14- نرخ خطای بیت برای رشته با طول 10 شکل 12- سیگنال اصلی در رشته شبه تصادفی حال با استفاده از مدولاسیون PSK باینری سیگنال را مدوله کرده و به یک کانال با نویز سفید گوسی جمع شونده ارسال می‌کنیم سیگنال به نویز کانال را 5dB در نظر می‌گیریم. همان طور که در شکل 13 مشاهده می‌کنید، نمودار قرمز سیگنال رسیده به گیرنده می‌باشد که با نویز بسیار زیادی جمع شده است و سیگنال آبی سیگنال قبل از ارسال به کانال است. سپس با استفاده از PSK باینری سیگنال ضرب شده در رشته شبه تصادفی را آشکار می‌کنیم. اگر طول سیگنال اصلی را T (در اینجا 10) در نظر بگیریم، برای آشکارسازی بعد از ضرب دوباره سیگنال شبه تصادفی در سیگنال دمدوله شده انتگرالی روی بازه T میگیریم. اگر مقدار انتگرال مثبت باشد یک و اگر منفی باشد صفر را آشکار میکنیم برای این روش BER را محاسبه کرده و می‌بینیم که با SNR در حدود 5dB خطای صفر را داریم اگر SNR را 2dB در نظر بگیریم، خطای آشکارسازی در حدود 5 تا 10 درصد میشود شکل 13- سیگنال مدوله شده (آبی)، سیگنال مدوله شده و عبور کرده از کانال (قرمز) در شکل 14 نمودار نرخ خطای بیت را برای این روش مشاهده می‌کنید که در این شبیه سازی طول رشته شکل 14- نرخ خطای بیت برای رشته با طول 10 شبه تصادفی ضرب شده در سیگنال اصلی برابر با 10 می‌باشد	شکل ۱۴ نرخ خطا بیت برای رشته با طول ۱۰ شکل ۱۲ سیگنال اصلی در رشته شبه تصادفی حال با استفاده از مدولاسیون PSK باینری سیگنال را مدوله کرده و به یک کانال با نویز سفید گوسی جمع شو ارسال کرد&کن سیگنال به نویز کانال را ۵dB در نظر گرفت&گیر همان طور که در شکل ۱۳ مشاهده کرد&کن نمودار قرمز سیگنال رسیده به گیرنده بود&باش که با نویز بسیار زیادی جمع شد&شو و سیگنال آبی سیگنال قبل از ارسال به کانال است. سپس با استفاده از دمدولاتور PSK باینری سیگنال ضرب شده در رشته شبه تصادفی را آشکار کرد&کن اگر طول سیگنال اصلی را T در اینجا ۱۰ در نظر گرفت&گیر برای آشکارسازی بعد از ضرب دوباره سیگنال شبه تصادفی در سیگنال دمدوله شده انتگرالی روی بازه T گرفت&گیر اگر مقدار انتگرال مثبت باشد یک و اگر منفی باشد صفر را آشکار کرد&کن برای این روش BER را محاسبه کرده و دید&بین که با SNR در حدود ۵dB خطا صفر را داشت&دار اگر SNR را ۲dB در نظر گرفت&گیر خطا آشکارسازی در حدود ۵ تا ۱۰ درصد شد&شو شکل ۱۳ سیگنال مدوله شده آبی سیگنال مدوله شده و عبور کرده از کانال قرمز در شکل ۱۴ نمودار نرخ خطا بیت را برای این روش مشاهده کرد&کن که در این شبیه ساخت&ساز طول رشته شکل ۱۴ نرخ خطا بیت برای رشته با طول ۱۰ شبه تصادفی ضرب شده در سیگنال اصلی برابر با ۱۰ بود&باش

۳-۲-۳- استخراج مدلهای کلمات از پیکره

طراحی یک روش برای برجسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۲۵

بعد از تشکیل عبارات اسمی، این عبارات رتبه بندی می‌شوند و عباراتی را که بهتر توصیف کننده تصویر هستند انتخاب می‌شوند. همانطور که در شکل ۳-۱ نشان داده شده است، برای این منظور ابتدا باید دو مدل فراوانی کلمه-معکوس فراوانی سند و کلمه-به-بردار از پیکره استخراج شود.

۳-۲-۳-۱- استخراج مدل‌های آماری

با استفاده از پیکره سه مدل آماری کلمات برای یوگرم‌ها، بایگرم‌ها و تریگرم‌های جداگانه استخراج می‌شود:

- مدل tf : ماتریسی تولید می‌شود که سطرهاى آن متناظر با تمام کلمات (یا عبارات اسمی) موجود در پیکره است و هر ستون آن نشان دهنده فراوانی هر کلمه در سند متناظر با ستون است.
- مدل idf : برداری تولید می‌شود که سطرهاى آن متناظر با تمام کلمات (یا عبارات اسمی) موجود در پیکره است و ستون آن لگاریتم معکوس کسر اسنادی است که حاوی هر کلمه است.
- مدل $tf-idf$: ماتریسی تولید می‌شود که سطرهاى آن متناظر با تمام کلمات (یا عبارات اسمی) موجود در پیکره است و هر ستون از حاصلضرب tf هر کلمه در idf آن به دست می‌آید.

۳-۲-۳-۲- استخراج مدل کلمه-به-بردار

عبارات کلیدی می‌توانند بر اساس شباهت آنها به زیرنویس هم رتبه‌بندی شوند. جهت محاسبه شباهت بین کلمات نیاز است که مدل عددی کلمات را با استفاده از روش‌های تعبیه کلمات استخراج کرد. مدل کلمه-به-بردار از روش‌های کارآمد تعبیه کلمات است که با استفاده از روش یادگیری عمیق و بدون نظارت محاسبه می‌شود (Mikolov et al. 2013). در این روش کلمات پیکره به عنوان ورودی دریافت می‌شود و ماتریسی تولید می‌شود که سطرهاى آن نشان دهنده بردار عددی متناظر با کلمات پیکره است. بردار متناظر کلمات با در نظر گرفتن شباهت معنایی و مفهومی آنها محاسبه می‌شود. روش کلمه-به-بردار از دو مدل معماری شبکه استفاده می‌کند. در مدل CBOW با دانستن مجموعه محدودی از کلمات اطراف y ، کلمه y پیش‌بینی می‌شود. در مدل skip-gram با دانستن کلمه

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۲۶

y کلمات اطراف آن پیش‌بینی می‌شود. در اینجا از مدل CBOW استفاده شده است. جزییات این

روش در فصل دوم بخش ۳-۳-۲ صفحه ۱۵ آمده است.

۳-۲-۵- استخراج برچسب‌ها

با استفاده از نرم افزار استخراج تصاویر (فخرزاده و صدیقی ۱۳۹۹) فایل XML ورد تجزیه می‌شود و تصاویر، زیرنویس و قسمتی از متن سند که در آن به تصویر مورد نظر اشاره شده است (متن مرجع) استخراج می‌شود. پس از اعمال پیش‌پردازش unigram و ترکیبهای bigram و trigram کلمات زیرنویس و متن مرجع تشکیل می‌شود. با استفاده از پیکره، مدل‌های tf، idf، tf-idf و کلمه-به-بردار محاسبه می‌شود. بر اساس مدل‌های آماری به دست آمده ۵ روش استخراج برچسبها معرفی می‌شود:

- در روش tf، مجموعه unigrams، bigrams و trigrams متن مرجع را بر اساس فراوانی آنها در سند مربوطه رتبه‌بندی می‌کنیم.
- در روش idf، مجموعه unigrams، bigrams و trigrams متن مرجع را بر اساس معکوس فراوانی آنها در پیکره رتبه‌بندی می‌کنیم.
- در روش tf-idf مجموعه unigrams، bigrams و trigrams متن مرجع را بر اساس نسبت فراوانی آنها در سند مربوطه به معکوس فراوانی آنها در پیکره رتبه‌بندی می‌کنیم.
- در روش شباهت کسینوسی، مجموعه unigrams، bigrams و trigrams متن مرجع را بر اساس شباهت کسینوسی با زیرنویس رتبه‌بندی می‌کنیم. با استفاده از مدل کلمه-به-بردار که از پیکره ساخته شده است، معادل برداری زیرنویس و unigrams، bigrams و trigrams موجود در متن مرجع را به دست می‌آوریم. معادل برداری عبارات چند کلمه‌ای، میانگین بردارهای متناظر با کلمات آنها است. شباهت کسینوسی بین بردار معادل زیرنویس V_c با بردار معادل عبارت استخراج شده از متن مرجع V_R ، یک معیار برای سنجش شباهت با زیرنویس محسوب می‌شود. شباهت کسینوسی در واقع نسبت ضرب داخلی دو بردار به حاصلضرب اندازه دو بردار است و به صورت زیر حساب می‌شود:

$$\text{cosine_similarity}(V_C, V_R) = \frac{V_C \cdot V_R}{|V_C| |V_R|} \quad (\text{فرمول ۱-۳})$$

- در روش شباهت کسینوسی و tf-idf، مجموعه unigrams، bigrams و trigrams را ابتدا بر اساس tf-idf رتبه بندی می کنیم. سپس k برچسب برتر را انتخاب می کنیم و دوباره بر اساس شباهت کسینوسی به زیرنویس رتبه بندی می کنیم.

تعداد برچسبهای خودکار پذیرفته شده در هر روش، با بهینه سازی میانگین معیار F_1 ، با استفاده از داده های آزمایشی تعیین می شود. به این ترتیب که در هر روش تعدادی از برچسبهای خودکار که به ازای آنها میانگین معیار F_1 بیشینه است، در نظر گرفته می شود. در ادامه، در فصل چهارم ویژگی های داده آزمایشی بررسی می شود. سپس در فصل پنجم روشهای معرفی شده با استفاده از داده آزمایشی بررسی شده و بهترین روش معرفی می شود.

فصل چهارم

آماده‌سازی داده آزمایشی

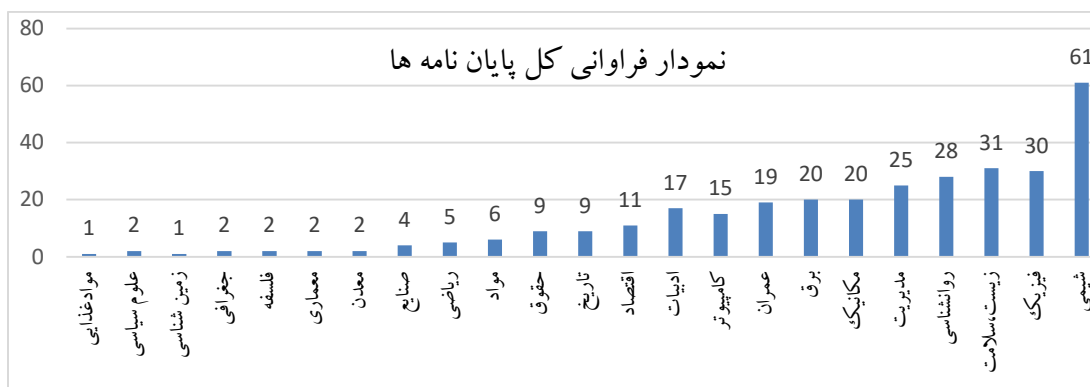
۴-۱- آماده سازی داده آزمایشی

در فصل پیش روشهای استخراج برچسب تصاویر علمی معرفی شد. برای ارزیابی روشهای پیشنهادی نیاز است که با استفاده از داده آزمایشی بررسی شوند. برای این منظور از اسناد موجود در گنج استفاده شده است. گنج با صدها هزار رکورد، که شامل اطلاعات پایان نامه ها و رساله ها و پیشنهادیه ها می شود، بزرگترین پایگاه اطلاعات علمی و فنی کشور است. این پایگاه هم اکنون مرجع بسیاری از پژوهشگران ایران و جهان است و روزانه بیش از ده هزار کاربر، ده ها هزار جست و جو در آن انجام می دهند. به همین دلیل یکی از بهترین پایگاههای اطلاعاتی داده برای بررسی و ارزیابی روشهای پردازش زبان طبیعی است. در ادامه مشخصات داده آزمایشی که در این طرح استفاده شده است، بررسی می شود.

۴-۲- پیکره

برای انجام این تحقیق، ۴۰۰ فایل ورد، پایان نامه و رساله به صورت تصادفی از پایگاه داده گنج انتخاب شده است. از این تعداد، ۳۲۲ فایل محتوای صحیح داشتند و برای انجام آزمایش در نظر گرفته شده اند. داده های درخواستی با دو برچسب کلی علوم فنی و علوم اجتماعی در اختیار ما قرار گرفت. از آنجایی که تنوع موضوعی فایلها بسیار زیاد بود، با استفاده از موضوع پارساها و رشته نویسنده ها برچسب های موضوعی اختصاصی تری را برای فایلها منظور کردیم که در شکل ۴-۱ نمایش داده شده است. مجموع این فایلها به عنوان پیکره مورد استفاده قرار گرفت. از پیکره برای ساختن مدل های آماری و ساختن مدل کلمه-به-بردار استفاده می شود.

طراحی یک روش برای برجسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۳۱

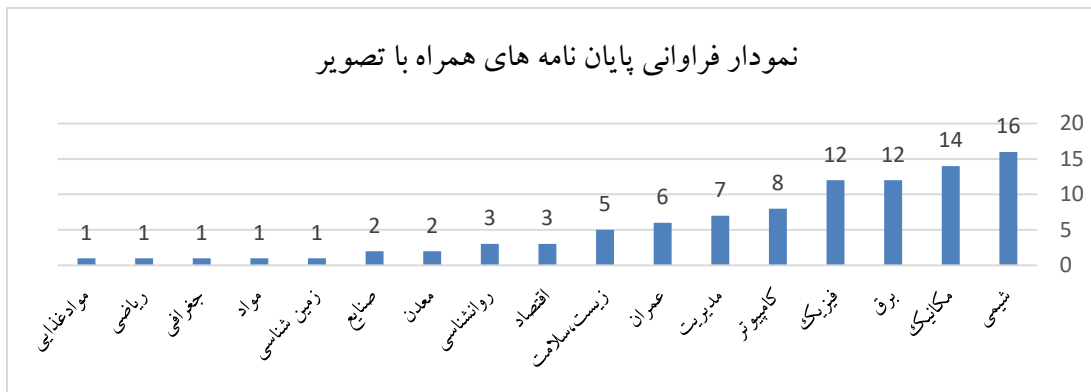


شکل ۴-۱ تقسیم بندی موضوعی پیکره

۴-۳- داده های آزمایشی

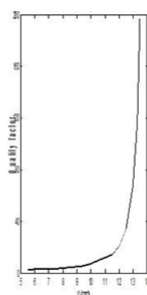
برای برجسب زدن تصویر، زیرنویس و قسمتی از متن سند که شامل اشاره به تصویر است به عنوان متن مرجع در نظر گرفته شد. درواقع، متن مرجع شامل پاراگراف حاوی اولین اشاره به تصویر، همراه با یک پاراگراف قبل و یک پاراگراف بعد از آن است. برای استخراج تصاویر و متن مزج از کد استخراج تصویر (فخرزاده، صدیقی ۱۳۹۸) استفاده شده است. کد استخراج تصویر قادر به استخراج تصاویر از فایل هایی است که فرمت استاندارد ورد را دارند. به دلیل اینکه تعدادی از فایل ها خالی از تصویر بودند و همچنین تعداد از آنها فرمت استاندارد نداشتند از مجموع ۳۲۲ فایل، از ۹۵ فایل تصاویر استخراج شد. جزییات موضوعی این ۹۵ فایل در شکل ۴-۲ نمایش داده شده است. همانطور که مشاهده می کنید، به غیر از هفت نمونه مدیریت، یک نمونه جغرافی و یک نمونه مواد غذایی بقیه فایل ها در حوزه فنی مهندسی و علوم پایه است. بنابراین به دلیل محدودیتهایی که در استخراج تصاویر و متن همراه وجود داشت بیشتر داده آزمایشی در حوزه فنی مهندسی و علوم پایه است. البته لازم به ذکر است روش های معرفی شده هیچ وابستگی ای به حوزه انتخابی ندارند. در مجموع ۱۶۵۰ تصویر استخراج شد و از این تعداد ۱۵۰ تصویر به صورت تصادفی انتخاب شده است و بر اساس زیرنویس و متنی که به تصویر در آن اشاره شده است برجسب موضوعی خورده اند. این ۱۵۰ تصویر برای محاسبه خطای سیستم استفاده می شود.

طراحی یک روش برای برجسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۳۲



شکل ۴-۲ تقسیم‌بندی موضوعی فایل‌هایی که تصویر از آنها استخراج شده است

نمونه‌هایی از داده آزمایشی در شکل ۴-۳، ۴-۴ و ۴-۵ آورده شده است. هر داده آزمایشی شامل زیرنویس و متن مرجع است. از محتوای تصویر در برجسب زدن استفاده نمی‌شود. داده‌های آزمایشی توسط سه خبره مستقل در حوزه مهندسی وفنی برجسب زده شده‌اند. به این ترتیب که خبرگان فقط زیرنویس و متن مرجع را مشاهده کردند و برجسب مناسب خود را انتخاب کردند. همانطور که در نمونه‌ها مشاهده می‌کنید انتخاب برجسب‌های مناسب فقط با در نظر گرفتن یک برش کوتاه از متن سند، بسیار چالش برانگیز است. در مواردی مثل شکل ۴-۳ پاراگراف‌های قبل و بعد از جمله حاوی اشاره به تصویر زیرنویس‌های جدول‌ها و تصاویر اطراف آن است که قابلیت توصیفی متن مرجع را کمتر می‌کند.



شکل ۴-۵: ضریب کیفیت خروجی شبیه سازی شده بر حسب فاصله

پیک نمودار تاخیر گروه نشان دهنده فرکانس مرکزی است باید در این فرکانس مقدار فاز یادداشت شود سپس باید فرکانس هایی که فاز آن نسبت به فاز فرکانس تشدید درجه اختلاف دارند مشخص شده و با استفاده از رابطه (۴-۳) مقدار ضریب کیفیت محاسبه شود.

با توجه به روند ذکر شده در بالا، برای ساختار فیلتر پیشنهادی نمودار ضریب کیفیت بر حسب فاصله در نمودار شکل (۴-۵) رسم شده است. نکته دیگری که در مورد ساختار تغذیه ورودی و خروجی می توان به آن اشاره کرد بحثی تحت عنوان ساختار تغذیه صفر درجه است (ب شکل ۴-۶: الف) ساختار تغذیه غیر ۰° (ب) ساختار تغذیه ۰° [۲۲]

برچسبها:

خبره اول: ضریب کیفیت، ضریب کیفیت خروجی، ساختار فیلتر، ضریب کیفیت، بر حسب فاصله، فرکانس تشدید، تغذیه صفر درجه

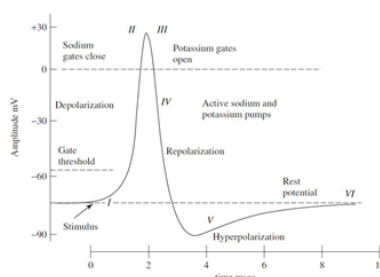
خبره دوم: ضریب کیفیت خروجی، ضریب کیفیت

خبره سوم: مقدار ضریب کیفیت، ضریب کیفیت، ساختار تغذیه

شکل ۴-۳ نمونه ای از داده های آزمایشی (ضیائی ۱۳۹۳)^۱

^۱ ضیائی بیدسکان مهری. ۱۳۹۳، طراحی شبیه سازی و ساخت فیلترهای قابل سوئیچ مایکروویو با خطوط ریزنواز، پایان نامه، دانشگاه فردوسی

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۳۴



شکل ۲-۱ تغییر ولتاژ پتانسیل عمل با زمان به هنگام رخداد تحریک [۸]

اطلاعات منتقل شده توسط یک عصب، پتانسیل عمل نامیده میشود پتانسیل عمل تغییر موقت در پتانسیل غشاء نرون است که در پاسخ به یک محرک، ابتدا در بدنه سلولی ایجاد و سپس در طول آکسون منتقل میشود پتانسیل های عمل با انواع مختلفی از محرک آغاز میشوند و به صورت معمول در یک جهت حرکت میکند اعصاب داخل سیستم عصبی مرکزی (مغز و نخاع) عمدتاً با فعالیت شیمیایی در سیناپس ها تحریک میشوند در شکل ۲-۱ یک نمونه پتانسیل عمل آورده شده است همانطور که در شکل مشاهده میشود، به محض اینکه قدرت محرک به بالای سطح آستانه میرسد، پتانسیل غشاء با دریافت یونهای سدیم، مثبت تر میشود و یک اسپایک تولید می کند پس از پیک زدن اسپایک پتانسیل غشاء منفی تر می شود پتانسیل نسبت به پتانسیل استراحت کمتر میشود و سپس به حالت عادی بر میگردد شکل ۲-۱ تغییر ولتاژ پتانسیل عمل با زمان به هنگام رخداد تحریک [۸] تولید اسپایک عمدتاً توسط باز کردن کانالهای سدیم ایجاد می شود هر دو یون سدیم و پتاسیم در تولید پتانسیل عمل نقش دارند سدیم در بیرون از سلول زیاد است و در درون سلول کم است سلول های قابل تحریک، کانالهای سدیم و پتاسیم درجه های مخصوصی دارند که در پاسخ به ولتاژ غشا باز و بسته میشوند باز کردن کانالهای سدیم اجازه میدهد که، سدیم به داخل سلول هجوم برده و بار مثبت را با خود حمل کند دامنه محدوده پتانسیل عمل برای انسان تقریباً بین ۶۰- میلی ولت تا ۱۰ میلی ولت می باشد [۸] ۱-۳-۲ تولید سیگنال EEG یک سیگنال EEG، از اندازه گیری جریان هایی حاصل میشود که از دندریت های تعداد زیادی از نورونهای مخروطی واقع در قشر مغز جاری میشود هنگامی که سلولهای مغزی فعال می شوند، جریانهای سیناپسی در دندریتها تولید می شوند این جریان یک میدان مغناطیسی قابل اندازه گیری با ماشین الکترومیوگرام و یک میدان الکتریکی روی پوست سر قابل اندازه گیری با سیستم های EEG تولید میکند سر انسان از لایه های مختلفی، شامل مغز، جمجمه و تعداد زیادی لایه های باریک بین آنها تشکیل شده است این لایهها سبب تضعیف سیگنالها، حدود صد برابر بیشتر نسبت به بافت نرم میشود بنابراین، تنها، تعداد جمعیت زیادی از نورونهای فعال، میتوانند پتانسیل کافی را تولید کنند که قابل ضبط توسط الکترودهای پوست سر باشد [۸]

برچسبها:

خبره اول: پتانسیل عمل، تغییر ولتاژ، پتانسیل غشاء

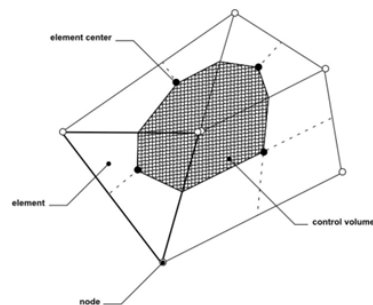
خبره دوم: پتانسیل غشاء، پتانسیل غشاء نرون، تغییر ولتاژ پتانسیل، قشر مغز، پتانسیل عمل، کانالهای سدیم

خبره سوم: تغییر ولتاژ، ولتاژ پتانسیل عمل

شکل ۴-۲ نمونه ای از داده های آزمایشی (حاتمی ۱۳۹۱)^۱

^۱ حاتمی رویا، ۱۳۹۱. طبقه بندی سیگنالهای EEG ناشی از تصور حرکتی به کمک تلفیق فیلترهای فضایی فرکانسی، پایان نامه، دانشگاه

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۳۵



شکل ۳-۵ حجم کنترل دو بعدی [۲۱]

۲-۴-۲ گسسته سازی معادلات پایه CFX از روش حجم محدود جهت گسسته سازی معادلات استفاده می‌کند که شامل گسسته سازی هندسه به کمک شبکه بندی است شبکه بندی جهت ساختن حجم‌های محدود که جهت نگهداری مقادیری مانند جرم، مومنتوم و انرژی استفاده می‌شوند به کار می‌رود جهت ساده سازی شبکه بندی را به صورت دو بعدی فرض می‌کنیم **شکل (۵-۳) نشان دهنده یک شبکه ۲ بعدی است** تمام متغیرهای حل و مشخصه‌های سیال بر روی گره‌های شبکه ذخیره می‌شوند شکل ۵-۳ حجم کنترل دو بعدی [۲۱] حجم کنترل (ناحیه سایه خورده) حول هر گره شبکه به وسیله خطوطی که مراکز لوله‌ها و مراکز المان‌ها را به هم وصل می‌کنند، ایجاد می‌شود جهت نشان دادن مدل‌سازی حجم محدود، معادلات بقا برای جرم و مومنتوم و متغیر اسکالر در مختصات دکارتی آورده شده است این معادلات بر روی هر حجم کنترل انتگرال گیری شده و با استفاده از قضیه دیورژانس به شکلهای زیر تبدیل می‌شوند که در آن‌ها S و V بیانگر سطح و حجم ناحیه انتگرال گیری و بردار نرمال خروجی از سطح استانتگرال‌های حجم بیانگر ترم‌های چشمه و انتگرال‌های سطح بیانگر مجموع شارها هستند مرحله بعدی در الگوریتم عددی گسسته سازی انتگرال‌های سطح و حجم است یک المان تنها مانند شکل (۶-۳) را در نظر می‌گیریم

برچسبها:

خبره اول: حجم کنترل، کنترل دو بعدی، حجم محدود، قضیه دیورژانس، معادلات پایه CFX
خبره دوم: کنترل دو بعدی، معادلات پایه CFX، گسسته سازی معادلات، گسسته سازی، معادلات بقا
خبره سوم: حجم کنترل دو بعدی، گسسته سازی معادلات، شبکه بندی

شکل ۴-۵ نمونه‌ای از داده‌های آزمایشی (مسعود ۱۳۹۲)^۱

^۱ جنتی مسعود، ۱۳۹۲. شبیه سازی عددی جریان عبوری از دریچه مصنوعی قلب، پایان نامه، دانشگاه یزد ..

فصل پنجم

پیاده‌سازی و ارزیابی نتایج

۵-۱- مقدمه

در این فصل ابتدا جزییات پیاده سازی روش پیشنهادی و ابزارهای مورد استفاده تشریح می شود. سپس معیارهای ارزیابی معرفی و روش پیشنهادی بر روی داده آزمایشی اعمال شده و خروجی سیستم بررسی می شود. در انتها بهترین روش برای برجسب زدن تصاویر در متون علمی فارسی معرفی می شود.

۵-۲- پیاده سازی

به منظور پیاده سازی از زبان برنامه نویسی پایتون نسخه ۳/۶ استفاده شده است. کتابخانه های NLTK و جنسیم^۱ (Rehurek, R. and P. Sojka. 2010) و پارسور^۲ (Salar et al 2018) مورد استفاده قرار گرفته است. کتابخانه NLTK یکی از جامع ترین و قدیمی ترین کتابخانه های پردازش زبان طبیعی در پایتون است. این کتابخانه پایه و استاندارد برای کتابخانه های پردازش متن محسوب می شود. تعداد معدودی کتابخانه مخصوص زبان فارسی وجود دارند و از معروف ترین آن ها هضم^۳ و پارسور هستند. ریشه یابی پارسور عملکرد بهتری از هضم داشت و به همین دلیل تمام مراحل پیش پردازش پیش پردازش و نرمال سازی و ریشه یابی با پارسور انجام شد. در جدول ۵-۱ خروجی ریشه یابی

¹ Gensim

² Parsivar

³ Hazm

طراحی یک روش برای برجسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۳۸

پارسیور و هضم مقایسه شده‌اند. همانطور که مشاهده می‌کنید به عنوان مثال عبارت تخصصی «نگاشت غیرخطی» در هضم تبدیل به «نگا غیر خط» شده است ولی در پارسیور تغییر پیدا نکرده است.

برای ساختن ngram ها و مدل‌های آماری مربوط به آنها از کتابخانه NLTK استفاده شد. با استفاده از مدل کلمه-به-بردار CBOW از کتابخانه جنسیم معادل برداری کلمات پیکره استخراج شد. در روش CBOW، طول بردار برابر با ۳۰۰ و پارامترهای اندازه‌ی پنجره و کمینه‌ی رخداد کلمات درسند هر دو برابر با ۵ در نظر گرفته شد.

جدول ۵-۲ مقایسه ریشه‌یابی هضم و پارسیور

متن اصلی	در حالت کلی هر نرون یک نگاشت غیرخطی از nIR به IR که معمولاً با توجه به تابع فعالسازی استفاده شده در بازه [۰،۱] یا [۱-۱] می‌باشد که شکل ۶ مدلی از این نگاشت را نشان می‌دهد. شکل ۶: مدل ریاضی یک نرون البته باید متذکر شد که در برخی نرونها مدل شده، از واحد ضرب کننده به جای واحد جمع‌کننده استفاده می‌کنند و هر ورودی به توان وزنش میرسد. واحدهای ضرب امکان ترکیبایی با درجه بالای ورودی را می‌دهد که دارای مزیت افزایش ظرفیت اطلاعات می‌باشد. همچنین در برخی از مدل‌ها یکی از ورودیهای نرون با وزن ۱- را به عنوان بایاس در نظر می‌گیرند. یک شبکه عصبی مصنوعی، یک شبکه لایه‌ای از نرونها مصنوعی می‌باشد. این شبکه شامل لایه ورودی، لایه‌های مخفی و لایه خروجی است که نرونها مصنوعی هر لایه بصورت کامل یا جزئی با نرونها مصنوعی لایه بعدی در ارتباطند.
هضم	در حال کل هر نرون یک نگا غیرخط از nIR به IR که معمولاً با توجه به تابع فعالساز استفاده شده در بازه [۰،۱] یا [۱-۱] می‌باشد که شکل ۶ مدل از این نگا را ن می‌دهد. شکل ۶: مدل ریاض یک نرون البته باید متذکر شد که در برخ نرون مدل شده، از واحد ضرب کننده به جا واحد جمع‌کننده استفاده می‌کنند و هر ورود به تو وزن میرسد. واحد ضرب امک ترکیب با درجه بالا ورود را می‌دهد که دارا مز افزا ظرف اطلاع می‌باشد. همچنین در برخ از مدل یک از ورودی نرون با وزن ۱- را به عنو بایاس در نظر می‌گیرند یک شبکه عصب مصنوعی، یک شبکه لای از نرون مصنوع می‌باشد. این شبکه شامل لایه ورودی، لایه مخف و لایه خروج اس که نرون مصنوع هر لایه بصور کامل یا جزئی با نرون مصنوع لایه بعد در ارتباطند
پارسیور	در حالت کلی هر نرون یک نگاشت غیرخطی از nIR به IR که معمولاً با توجه به تابع فعالسازی استفاده شده در بازه [۰،۱] یا [۱-۱] بود&باش که شکل ۶ مدل از این نگاشت را نشان داد&ده شکل ۶: مدل ریاضی یک نرون البته باید متذکر شد که در برخی نرون مدل شده، از واحد ضرب کننده به جای واحد جمع‌کننده استفاده کرد&کن و هر ورودی به توان وزن رسید&رس واحد ضرب امکان ترکیب با درجه بالای ورودی را داد&ده که دارای مزیت افزایش ظرفیت اطلاعات بود&باش

همچنین در برخی از مدل یکی از ورودی نرون با وزن ۱- را به عنوان بایاس در نظر گرفت & گیر یک شبکه عصبی مصنوعی، یک شبکه لای از نرون مصنوعی بود & باش این شبکه شامل لایه ورودی، لایه مخفی و لایه خروجی اس که نرون مصنوعی هر لایه بصورت کامل یا جزئی با نرون مصنوعی لایه بعدی در ارتباطند	
--	--

برای استخراج برجسبها نیاز است که تصاویر، زیرنویس، پاراگرافی که شامل اشاره به تصویر است به همراه یک پاراگراف قبل از آن و بعد از آن استخراج شود. برای این منظور از الگوریتم فخرزاده و صدیقی (فخرزاده و صدیقی ۱۳۹۹) استفاده شد. الگوریتم فخرزاده و صدیقی فقط تصاویر به همراه زیرنویس را استخراج می کرد. این الگوریتم جهت استخراج متن مرجع به روز رسانی شد. در زیر گامهای الگوریتم به روز شده مشاهده می شود، تغییرات اعمال شده درشت نمایی شده اند.

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 دربر گرفته اند موارد زیر را انجام بده:

گام ۳-۳-۱: برای تمام عناصر همبرده 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 برجسب 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: 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: 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.

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:

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۴۱

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

گام ۳-۴-۱-۲: 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 ۳-۴-۲-۲: 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 ۳-۴-۳: 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 و **شناسه تصویر 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 و **شناسه تصویر 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 ۳-۵-۱-۲: 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 و **شناسه تصویر را ذخیره کن.** در نهایت به گام ۳-۶ برو.

گام ۳-۵-۲: 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

و **شناسه تصویر** را ذخیره کن. در نهایت به گام ۳-۶ برو.

گام ۳-۵-۳: اگر عنصر بعدی پاراگراف و جدول نبود به گام ۳-۷ برو.

گام ۳-۶ **شناسه تصویر را جستجو کن**. پاراگرافی که شامل شناسه تصویر است

اگر تعداد کلماتش از آستانه بیشتر است، به همراه یک پاراگراف قبل از آن و

بعد از آن ذخیره کن و به گام ۳-۷ برو.

گام ۳-۷. اگر گره تصویر دیگری وجود ندارد توقف کن، در غیر این صورت به گام ۳-۱ برو.

۳-۵- معیارهای ارزیابی

برای ارزیابی بازخوانی از معیار مککارتی و نوئیگیلی (McCarthy and Navigli, 2007) استفاده

کردیم. برای محاسبه این معیار، تعدادی خبره مستقل تصاویر را برچسب می‌زنند. و هر برچسب

خودکاری که استخراج می‌شود براساس فراوانی آن در لیست برچسب خبرگان وزن‌دهی می‌شود.

اگر فرض کنیم $H = \{h_1, h_2, h_3\}$ مجموعه خبرگان باشد و $I = \{i_1, i_2, i_3, \dots\}$ مجموعه

تصاویری باشد که توسط خبرگان برچسب خورده‌اند، T_j مجموعه تمام برچسب‌هایی است که توسط

خبرگان، برای تصویر i_j در نظر گرفته شده و a_j مجموعه برچسب‌هایی باشد که توسط سیستم برای

تصویر i_j انتخاب شده‌است. بازخوانی به صورت زیر تعریف می‌شود:

$$R = \frac{\sum_{a_j: i_j \in I} \frac{\sum_{word \in a_j} freq_{word}}{|T_j|}}{|I|} \quad (\text{فرمول ۵-۱})$$

در معادله بالا $freq_{word}$ فراوانی کلمه $word$ از مجموعه برچسب‌های خودکار a_j در مجموعه

برچسب‌های دستی است. دقت سیستم به صورت زیر محاسبه می‌شود:

$$P = \frac{\sum_{a_j: i_j \in I} \frac{\sum_{word \in a_j} \{1 \text{ if } word \in T_j\}}{|a_j|}}{|I|} \quad (\text{فرمول ۵-۲})$$

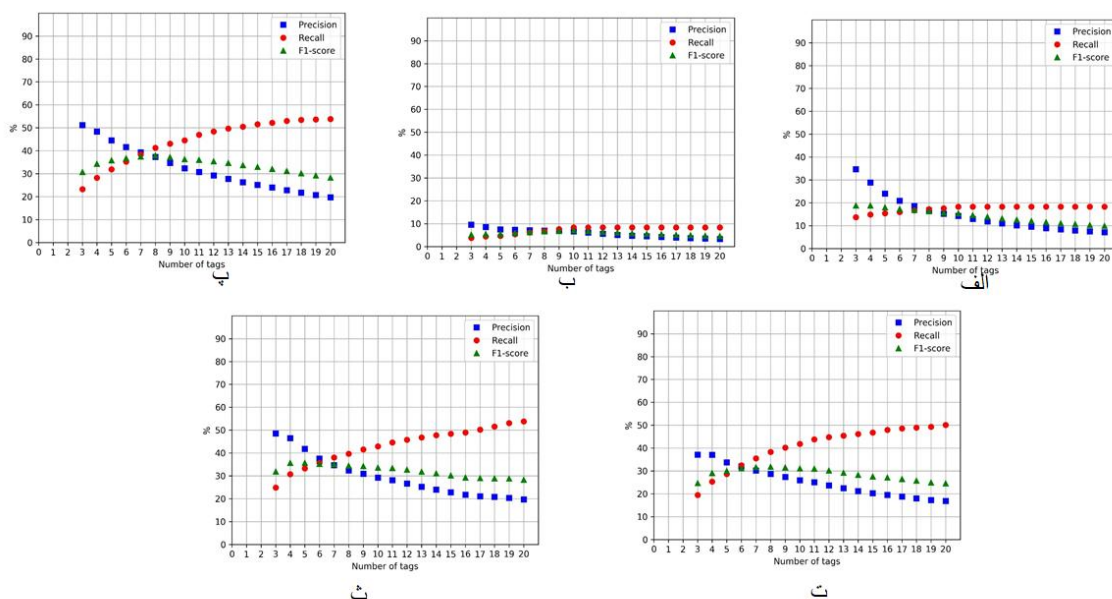
با استفاده از معیار بازخوانی و دقت معیار F_1 به صورت زیر محاسبه می‌شود:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \quad (\text{فرمول ۳-۵})$$

۴-۵- بررسی نتایج

همانطور که در فصل قبل توضیح داده شد، ۱۵۰ تصویر توسط سه خبره برچسب زده شده و برای ارزیابی روشهای معرفی شده استفاده شده است. از متن همراه تصویر که شامل زیرنویس و متن مرجع است، عبارات اسمی که شامل همه ترکیبات unigram، bigrams و trigram متن مرجع است، استخراج شد. سپس عبارات اسمی استخراج شده با استفاده از پنج روش فراوانی کلمه، معکوس فراوانی سند، فراوانی کلمه- معکوس فراوانی سند، شباهت کسینوسی عبارات با زیرنویس و ترکیب روش فراوانی کلمه- معکوس فراوانی سند و شباهت کسینوسی رتبه بندی شدند در مشاهدات اولیه با در نظر گرفتن unigrams عبارات نامرتب زیادی در خروجی مشاهده شد و تاثیر منفی در کارکرد الگوریتم داشت. همچنین در مجموعه برچسب‌هایی که توسط خبرگان انتخاب شده بود تعداد خیلی محدودی برچسب unigram وجود داشت. به همین دلیل برای این مساله تنها bigrams و trigrams در نظر گرفته شد.

در هر روش بیست برچسب برتر که رتبه بالاتری دارند برای هر تصویر در نظر گرفته شد. بر اساس مشاهدات انجام شده بعد از بیست برچسب، موارد مرتبط به ندرت وجود داشت. برای برچسبهای یک تا بیست، میانگین مقادیر دقت و بازخوانی و معیار F_1 برای همه تصاویر محاسبه شد. نمودار میانگین دقت و بازخوانی و معیار F_1 برای پنج روش معرفی شده در شکل ۵-۱ نمایش داده شده است.



شکل ۵-۱. میانگین مقادیر دقت (آبی)، بازخوانی (قرمز) و معیار F_1 برای ۲۰ برچسب برتر در ۵ روش معرفی شده: الف) tf (ب) idf (پ) tf-idf (ت) شباهت کسینوسی (ث) tf-idf و شباهت کسینوسی

همانطور که در نمودارهای شکل ۵-۱ مشاهده می‌کنیم، هرچه تعداد برچسب‌های انتخابی توسط سیستم را افزایش دهیم بازخوانی سیستم بهتر می‌شود اما به دلیل ظاهر شدن مثبت‌های اشتباه، دقت سیستم و به همان نسبت معیار F_1 کاهش پیدا می‌کند. در هر روش تعدادی از برچسب‌های برتر در نظر گرفته می‌شود که میانگین معیار F_1 به ازای آن بیشینه است. همانطور که در جدول ۵-۲ مشاهده می‌شود روش‌های tf و idf عملکرد ضعیفی دارند و روش‌های شباهت کسینوسی، tf-idf و ترکیب شباهت کسینوسی با tf-idf خروجی قابل مقایسه‌ای دارند. از بین همه روش‌ها tf-idf بیشترین معیار F_1 را دارد را به عنوان روش مطلوب معرفی می‌شود. نمودار ۵-۱ نشان می‌دهد که tf-idf با هفت برچسب انتخابی بهترین خروجی را دارد.

عباراتی که tf-idf بالاتری دارند در واقع کلمات کلیدی متن هستند. در راستا با تحقیقات جوسی و همکاران (Josi et al 2018)، انتظار اولیه ما این بود که کلمات موجود در زیرنویس بهترین توصیف کننده‌های تصویر باشند و بر این اساس با رتبه بندی مجدد کلمات کلیدی بر اساس شباهت

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۴۵

کسینوسی با زیرنویس برچسبهای بهتری برای تصویر به دست بیاید. اما نتایج آزمایشات نشان داد در نظر گرفتن شباهت کسینوسی با زیرنویس، اگر چه نتایج قابل مقایسه‌ای با tf-idf دارد، باعث بهبود خروجی آن نشد. دلیل اصلی این امر این است که در خیلی از موارد زیرنویسهای انتخاب شده توسط نویسنده سند دقیق نیستند

جدول ۵-۲ مقایسه ۵ روش پیشنهادی

Method	Precision	Recall	F_1
tf	٪۱۵	٪۱۷	٪۱۵
idf	٪۶	٪۸	٪۷
tf-idf	٪۳۷	٪۴۱	٪۳۸
Cosine similarity	٪۲۹	٪۳۸	٪۳۱
tf-idf + Cosine similarity	٪۴۲	٪۳۳	٪۳۶

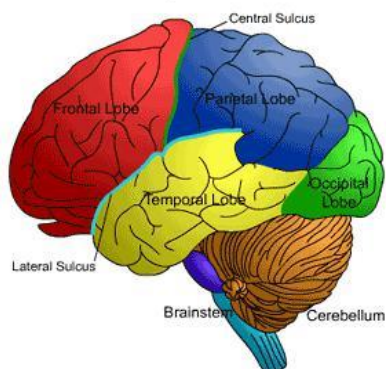
در شکل ۵-۲، ۵-۳ و ۵-۴ نمونه‌هایی از تصاویر استخراج شده، متن همراه و بیست برچسب انتخابی توسط سیستم با سه روش مختلف را نشان می‌دهد. اولین پاراگراف عباراتی که tf-idf بالاتری را دارد، نشان می‌دهد. این عبارات بار دیگر بر اساس شباهت کسینوسی رتبه‌بندی شده‌اند و در پاراگراف سوم نمایش داده شده‌است. پاراگراف دوم عبارات فقط بر اساس شباهت کسینوسی با زیرنویس رتبه‌بندی شده‌اند. کلماتی که درشت‌نمایی شده‌اند عباراتی هستند توسط خبرگان نیز به عنوان برچسب در نظر گرفته شده‌اند. در شکل ۵-۲ کلمه «نمای جانبی» به اشتباه در زیرنویس دوبار تکرار شده است و این باعث شده‌است که در شباهت کسینوسی با زیرنویس، عباراتی که شامل {نما و جانبی} هستند، رتبه بالاتری از عباراتی بگیرند که شامل کلمه «مغز» هستند. در حالیکه در این تصویر از نظر سه خبره «قشر مغز» و «لوب» کلمات کلیدی توصیف کننده تصویر است و «نما» و «جانبی» کلمات کم اهمیتی

هستند. همانطور که مشاهده می کنید با ترکیب شباهت کسینوسی با tf-idf نیز بهبودی در خروجی tf-idf ایجاد نمی شود.

گاهی، زیرنویس یک توضیح کلی از تصویر را ارائه می دهد و توضیح دقیق تر تصویر در متن مرجع می آید. مثلاً در شکل ۳-۵ زیرنویس «نتایج حاصل از ردیابی مسیر تحت نویز سفید» است و در توضیحات تکمیلی در متن مرجع همراه تصویر، چنین به نظر می آید که در این تصویر برای بازوی مکانیکی ماهر، خطای ردیابی مسیر در روش کنترلر PID تحت نویز سفید بررسی شده است. بنابراین خبرگان «بازوی مکانیکی ماهر» و «کنترلر PID» را به عنوان برچسب در نظر گرفته اند که با خروجی سیستم بر اساس روش tf-idf تطابق دارد. با در نظر گرفتن شباهت کسینوسی ترکیباتی که کلمه کم اهمیتی مثل «مسیر» را شامل میشوند رتبه بالاتری پیدا می کنند که مطلوب نیست. در شکل ۴-۵ زیرنویس کامل و گویا است ولی در این مورد هم خروجی روش tf-idf با شباهت کسینوسی خیلی به هم نزدیک است.

یکی از مشکلات روشهای معرفی شده ظاهر شدن ترکیبات بی معنی مثل «عادی بدن ایجاد»، «ناحیه اصلی شامل» در خروجی است، که دقت سیستم را پایین می آورد. البته اگر عبارات توصیف کننده مربوطه که احتمال جستجو آنها توسط کاربر زیاد است در برچسبهای پیشنهادی سیستم دیده شود، این دست از متهای اشتباه چندان مشکل ساز نیست.

اطلاعاتی که در دسترس خبرگان قرار گرفته است زیرنویس و متن مرجع است. اما سیستم علاوه بر این اطلاعات به مدلهای آماری پیکره هم دسترسی دارد به همین دلیل در بعضی موارد برچسبهای مناسبی انتخاب می کند که توسط خبرگان انتخاب نشده است. مثلاً در شکل ۲-۵ «تحقیقات BCI» برچسب مناسبی به نظر می رسد، اگر چه توسط خبرگان انتخاب نشده است. در بعضی موارد هم برچسبهای پیشنهادی توسط سیستم به دلیل اختلافات جزئی با برچسب خبرگان خطا در نظر گرفته شده است. مثلاً در شکل ۴-۵ برچسب «آرایه VCSEL» به عنوان خطا در نظر گرفته شده، اگر چه «VCSEL» توسط خبرگان انتخاب شده است.



شکل ۴-۱ نمای جانبی از نمای جانبی لوبهای قشر مغز

در تحقیقات BCI بسیار مهم است که بدانیم که کدام ناحیه از کورتکس مربوط به فعالیت مورد تحقیق است و چه تغییراتی در سیگنال نوروون های آن قسمت خاص نسبت به حالت عادی بدن ایجاد میشود. ۳-۱-۴ نواحی قشر مغز لوب های مخ را متناسب با عملکرد آن به چهار ناحیه اصلی شامل لوب پیشانی، لوب آهیانه ای، لوب گیجگاهی، لوب پس سری تقسیم بندی میشود. اسامی این لوب ها برگرفته از استخوان جمجمه است. شکل ۴-۱ نمای جانبی لوبهای قشر مغز را نشان میدهد. (شکل ۴-۱ نمای جانبی از نمای جانبی لوبهای قشر مغز) ۱-۴-۳-۱ لوب پیشانی یا قدامی این ناحیه در قسمت جلویی سر واقع شده است و مرتبط با فعالیتهای مغزی مربوط به تمرکز، چاره یابی، هیجانات، داوری، سخن گفتن و حرکات ارادی است [۹]. ۱-۴-۳-۲ لوب آهیانه ای

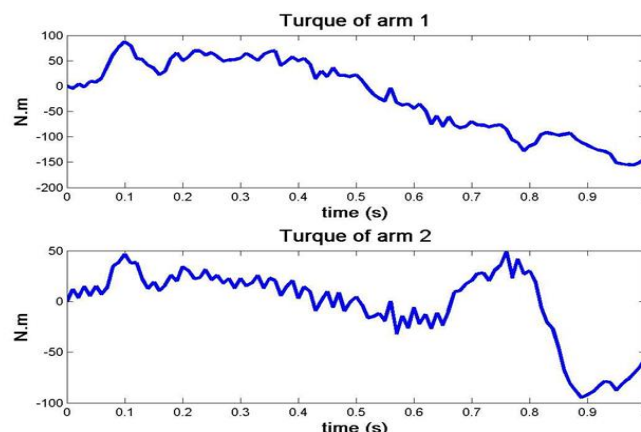
Tf-idf: قشر مغز، فعالیت مغزی، لوب پیشانی، لوب آهیانه ای، لوب گیجگاهی، نما جانبی، نما جانبی لوبهای، جانبی لوبهای قشر، لوبهای قشر مغز، جانبی لوبهای، لوبهای قشر، نواحی قشر مغز، تحقیقات BCI، عادی بدن ایجاد، لوب پیشانی لوب، قسمت خاص نسبت، ناحیه اصلی شامل، حالت عادی بدن، شامل لوب پیشانی، اصلی شامل لوب

Cos-similarity: جانبی لوبهای، - نما، نما جانبی، قشر مغز، نما جانبی لوبهای، جانبی لوبهای قشر، - نما جانبی، لوبهای قشر، لوبهای قشر مغز، لوب های مخ، قشر مغز لوب های مخ، استخوان جمجمه، لوب پیشانی، لوب، لوب گیجگاهی، گیجگاهی لوب، فعالیت مغزی، عادی بدن، لوب آهیانه ای، آهیانه ای لوب

Tf-idf+cos-similarity: جانبی لوبهای، قشر مغز، نما جانبی، نما جانبی لوبهای، جانبی لوبهای قشر، لوبهای قشر، لوبهای قشر مغز، لوب پیشانی، لوب گیجگاهی، فعالیت مغزی، لوب آهیانه ای، عادی بدن ایجاد، لوب پیشانی لوب، نواحی قشر مغز، قسمت خاص نسبت، ناحیه اصلی شامل، حالت عادی بدن، شامل لوب پیشانی، اصلی شامل لوب، تحقیقات BCI

برچسبهای خبرگان: قشر مغز (۳)، لوبهای قشر مغز (۳)، نما جانبی لوبهای، لوب پیشانی، لوب آهیانه ای، لوب گیجگاهی، لوب پس سری، لوب پیشانی

شکل ۲-۵ مقایسه سه روش برتر استخراج برچسب. عبارات درشت نمایشی شده در هر روش توسط خبرگان هم انتخاب شده است. (حاتمی ۱۳۹۱)



شکل ۱۶: نتایج حاصل از ردیابی مسیر تحت نویز سفید

شکل ۱۵: گشتاور حاصل برای مسیر موردنظر بوسیله روش ارائه شده در [۳۶] و WNN-CS در شکل ۱۵ نمودارهای گشتاور بدست آمده برای مسیر مرجع توسط روش کنترل بهینه حلقه باز ارائه شده در [۳۶] نشان داده میشود. جدول ۹: مقادیر پارامترهای یک بازوی مکانیکی ماهر دولینکه صفحه‌ای. همانگونه که مشاهده میشود نتایج حاصل از روش ارائه شده در پژوهش حاضر با استاندارد به روش ارائه شده در [۳۶] معتبر میباشد و خطای حاصل نیز مربوط به خطای ردیابی مسیر در روش کنترل PID مبتنی بر WNN-CS میباشد. حال همان مسیر را تحت نویز سفید قرار داده تا کنترلر پیشنهادی از نظر مقاوم بودن بررسی گردد. نتایج حاصل در شکل ۱۶ آمده‌است. همانطور که دیده میشود کنترلر پیشنهادی با بیشترین خطای، مقاومت خوبی در برابر نویز دارد

Tf-idf: بازو مکانیکی، بازو مکانیکی ماهر، مکانیکی ماهر، کنترلر PID، ردیابی مسیر، خطا ردیابی، مسیر موردنظر، مکانیکی ماهر
دولینکه، کنترلر پیشنهادی، ماهر دولینکه، PID مبتنی، گشتاور بدست، ماهر دولینکه صفحه‌ای، کنترلر PID مبتنی، مسیر موردنظر بوسیله، خطا ردیابی مسیر، کنترلر بهینه حلقه، نویز سفید قرار، حالت بعدی موقعیت، خطا مقاومت خوبی

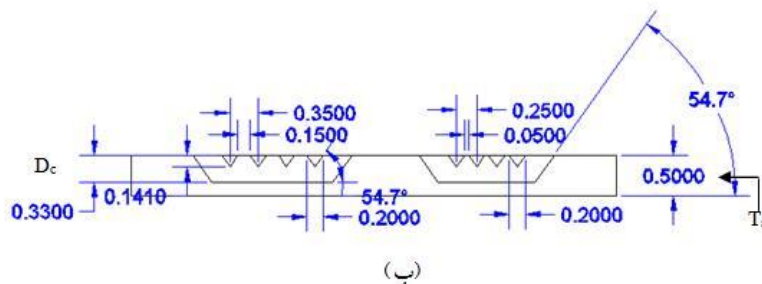
Cos-similarity: نتایج حاصل، ردیابی مسیر، نویز سفید، مسیر موردنظر، نتایج ردیابی، مسیر مرجع، اولیه مسیر، مسیر موردنظر بوسیله، خطا ردیابی مسیر، مسیر موردنظر شروع، اولیه مسیر موردنظر، خطا حاصل، حالت اولیه مسیر، خطا ردیابی، مسیر قرار، گشتاور حاصل، بهینه حلقه، کنترلر بهینه، کنترلر بهینه حلقه، موردنظر شروع

Tf-idf+cos-similarity: ردیابی مسیر، مسیر موردنظر، مسیر موردنظر بوسیله، خطا ردیابی مسیر، خطا ردیابی، کنترلر بهینه حلقه، نویز سفید قرار، حالت بعدی موقعیت، کنترلر پیشنهادی، خطا مقاومت خوبی، کنترلر PID، گشتاور بدست، بازو مکانیکی ماهر، بازو مکانیکی، ماهر دولینکه صفحه‌ای، ماهر دولینکه، کنترلر PID مبتنی، مکانیکی ماهر دولینکه، PID مبتنی، مکانیکی ماهر

برچسبهای خبرگان: بازوی مکانیکی (۲)، بازوی مکانیکی ماهر (۲)، نویز سفید (۳)، تحت نویز سفید، کنترلر PID

شکل ۵-۳. مقایسه سه روش برتر استخراج برچسب. عبارات درشت نمایی شده در هر روش توسط خبرگان هم انتخاب شده است. (اسلامی ۱۳۹۱)^۱

^۱ اسلامی، علی. ۱۳۹۱. طراحی کنترلر شبکه ویولتی برای بازوی مکانیکی متحرک. پایان نامه کارشناسی ارشد. دانشگاه تبریز



شکل ۳-۵: ابعاد میز نوری سیلیکونی الف) دید از بالای میز نوری ب) دید از جانب میز نوری (ابعاد در مقیاس میلی متر می باشند)

شکل ۳-۵: شماتیک میز نوری سیلیکونی به همراه المانهای نصب شده بر روی آن ابعاد VCSEL و دیود نوری معرفی شده در فصل دوم به طور خلاصه در جدول ۱-۵ مشخص شده اند. ابعاد حفره ها و شیارهای V شکل متناسب با ابعاد VCSEL، فوتودایود و فیبر نوری مشخص شده اند. به عنوان مثال طول حفره ی مربوط به آرایه ی VCSEL باید از $965 \mu\text{m}$ (طول VCSEL) و طول حفره ی گیرنده باید از $1350 \mu\text{m}$ (طول چهار دیود نوری با فاصله ی $50 \mu\text{m}$ از یکدیگر) بزرگتر باشد. شکل ۳-۵ دید از بالای میز نوری را نشان می دهد که در این شکل ابعاد حفره ها و شیارها، مشخص شده است. شکل ۳-۵ دید از کنار میز نوری را نشان می دهد که در این شکل زاویه ی دیوارهای حفره، عرض شیارها و عمق حفره ها و شیارها مشخص شده است. براساس این شکل فاصله ی بین شیارهای V شکل در طرف فرستنده برابر با $250 \mu\text{m}$ (فاصله ی بین دو VCSEL در آرایه ی VCSEL) و در طرف گیرنده برابر با $350 \mu\text{m}$ می باشد. ابعاد میز نوری طراحی شده در جدول ۲-۵ مشخص شده است. شکل ۳-۵: ابعاد میز نوری سیلیکونی الف) دید از بالای میز نوری ب) دید از جانب میز نوری (ابعاد در مقیاس میلی متر می باشند) به دلیل زاویه ی واگرایی کم VCSEL (حدود 14° درجه).

Tf-idf: میز نوری، میز نوری سیلیکونی، نوری سیلیکونی، دیود نوری، آرایه ی VCSEL، فیبر نوری، ضریب شکست، سطح زیرین فیبر، زیرین فیبر، شیار V، سطح زیرین، عرض حفره، میز نوری طراحی، ابعاد میز نوری، شماتیک میز نوری، محور نوری فیبر، میز نوری ابعاد، نوری سیلیکونی طراحی، دلیل زاویه واگرایی، همراه المانهای نصب

Cos-similarity: میز نوری، میلی متر، سیلیکونی الف)، نوری ابعاد، نوری سیلیکونی، الف) دید، جانب میز، مقیاس میلی، میز نوری سیلیکونی، ابعاد میز نوری، جانب میز نوری، میز نوری ابعاد، سیلیکونی الف) دید، مقیاس میلی متر، ابعاد میز، نوری سیلیکونی الف)، ابعاد VCSEL، ابعاد دیود نوری، ابعاد المانهای، - ابعاد

Tf-idf+cos-similarity: میز نوری، نوری سیلیکونی، میز نوری سیلیکونی، ابعاد میز نوری، میز نوری ابعاد، فیبر نوری، دیود نوری، شماتیک میز نوری، نوری سیلیکونی طراحی، میز نوری طراحی، محور نوری فیبر، آرایه ی VCSEL، زیرین فیبر، عرض حفره، سطح زیرین فیبر، همراه المانهای نصب، دلیل زاویه واگرایی، شیار V، ضریب شکست، سطح زیرین

برچسبهای خبرگان: میز نوری سیلیکونی (۳)، میز نوری (۳)، نوری سیلیکونی، دیود نوری (۲)، VCSEL، ابعاد میز نوری، فیبر نوری (۲)

شکل ۴-۵ مقایسه سه روش برتر استخراج برچسب. عبارات درشت نمایشی شده در هر روش توسط خبرگان هم انتخاب شده است. (افیونی اکبری ۱۳۹۱)^۱

^۱ افیونی اکبری، شیرین، ۱۳۹۱، طراحی و پیاده سازی میز نوری سیلیکونی جهت کوپلینگ VCSEL، پایان نامه کارشناسی ارشد، دانشگاه اصفهان

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۵۰

فصل ششم

نتیجه گیری و پیشنهادات آتی

۶-۱- نتیجه‌گیری و پیشنهادات آتی

روشی برای برچسب زدن تصاویر موجود در اسناد علمی معرفی شد. تصاویر حاوی اطلاعات مهمی از اسناد علمی هستند، به همین دلیل امروزه تحقیقات در زمینه بازیابی تصاویر از اسناد علمی، بسیار رو به رشد است. به دلیل پیچیدگی‌های محتوایی تصاویر علمی، در این تحقیق از زیرنویس و قسمتی از متن سند که شامل اشاره به تصویر است، برای برچسب زدن تصاویر استفاده شد. ابتدا با استفاده از الگوریتم پیشنهاد شده توسط فخرزاده و صدیقی (فخرزاده و صدیقی ۱۳۹۹) زیرنویس تصاویر و متن مرجع (قسمتی از متن سند که شامل اشاره به تصویر مورد نظر است) از اسناد استخراج شد.

برای استخراج برچسب تصاویر، پیش پردازشهای رایج متن (نرمال‌سازی، ریشه‌یابی) بر روی متن پیکره و متن همراه تصویر اعمال شد. در مرحله بعد عبارات اسمی متن مرجع، که شامل همه ترکیبات یک کلمه‌ای، دو کلمه‌ای، سه کلمه‌ای متن مرجع است، استخراج شد. مدل‌های آماری عبارات و مدل کلمه-به-بردار از پیکره استخراج شد. با استفاده از مدل‌های به دست آمده از پیکره، عبارات اسمی موجود در متن همراه تصویر، با استفاده از پنج روش فراوانی کلمه، معکوس فراوانی سند، فراوانی کلمه-معکوس فراوانی سند، شباهت کسینوسی عبارات با زیرنویس و ترکیب روش فراوانی کلمه-معکوس فراوانی سند و شباهت کسینوسی رتبه بندی شدند. تعدادی از عبارات اسمی با رتبه بالاتر که میانگین معیار F_1 به ازای آن تعداد بیشینه است به عنوان برچسب پیشنهادی در هر روش معرفی شد. آزمایشات نشان داد که روش فراوانی کلمه-معکوس فراوانی سند عملکرد بهتری نسبت سایر روشها دارد.

یکی از مشکلات روش پیشنهادی و البته تمام روش‌های مشابه که بر اساس مدل‌سازی آماری طراحی شده است این است که در خروجی نهایی عبارات مثبت اشتباهی مشاهده می‌شود که در خیلی موارد بارمعنایی کمی دارند. در صورت دسترسی به پیکره عظیم داده در حوزه تخصصی مورد نظر،

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۵۳

می‌توانیم مدل‌های کلماتی قابل اعتمادتری بسازیم. مدل کلماتی بهتر خطای سیستم را کاهش می‌دهد. همچنین دسترسی به یک بانک داده عظیم از واژه‌های کلیدی موجود در حوزه مربوطه می‌تواند در فیلتر کردن عبارات بی‌معنی کمک کننده باشد.

در روش ارائه شده، با بررسی متنی که به تصویر اشاره می‌کند برچسب‌های تصویر استخراج شده‌است و محتوای خود تصویر در نظر گرفته نشده‌است. از جمله پیشنهادها برای تحقیقات آتی این است که محتوای خود تصویر هم بررسی شود. از آنجایی که تنوع تصاویر در اسناد علمی زیاد است و ویژگی‌های یک گروه با گروه دیگر از تصاویر خیلی متفاوت است، بنابراین در قدم اول باید تصاویر را گروه‌بندی کرد و سپس با استفاده از روش‌های پردازش تصویر اطلاعات مناسب در خود تصاویر را برای برچسب زدن در نظر گرفت.

منابع

فخرزاده، آزاده و مجتبی زالی. ۱۳۹۸. ایجاد پایگاه داده از تصاویر موجود در پایگاه اطلاعات علمی ایران (گنج). طرح پژوهشی. تهران: پژوهشگاه علوم و فناوری اطلاعات ایران (ایرانداک).

فخرزاده، آزاده و امیرحسین صدیقی. ۱۳۹۹. ارائه روشی ساختارمحور برای ایجاد پایگاه داده از تصاویر مستخرج از اسناد علمی؛ مورد مطالعه: پایگاه اطلاعات علمی ایران (گنج). پژوهشنامه پردازش و مدیریت اطلاعات ۳۵ (۱۰۱): ۷۲۹-۷۵۴.

Barnard K. and D. Forsyth. 2001. Learning the semantics of words and pictures. *Proceedings Eighth IEEE International Conference on Computer Vision* 408-415. Vancouver, BC, Canada. doi: 10.1109/ICCV.2001.937654.

Barzilay R. and M. Elhadad. 1997. Using lexical chains for text summarization. *Intelligent Scalable Text Summarization*

Bellegarda J. R. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE* 88(8):1279-1296. doi: 10.1109/5.880084

Blei D. M., A.Y. Ng and M. I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3(4-5): 993-1022

Bratasanu D., I. Nedelcu and M. Datcu, Bridging. 2011. The semantic gap for satellite image annotation and automatic mapping applications. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 4 (1): 193-204.

Biswas S. Kr., M.Bordoloi and J.Shreya. 2018. A graph based keyword extraction model using collective node weight. *Expert Systems with Applications* 97 (51-59).

Chan Ch. Sh., A. Johar and J. Lang Hong, Contextual information for image retrieval systems. 2013. *2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)* 863-

طراحی یک روش برای برجسب زدن تصاویراستخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۵۶

867. Shenyang. doi: 10.1109/FSKD.2013.6816315.

Campos J, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes and A. Jatow. 2020. YAKE! Keyword extraction from single documents using multiple local feature. *information Sciences* 509 (257-289).

Chien L. F. 1997. Pat-tree-based keyword extraction for Chinese information retrieval. *ACM SIGIR Forum* 31(50-58).

Corinna B., G. Bela and L. Stefan. 2015. Research-paper recommender systems: a literature survey. In *International Journal on Digital Libraries* 17 (4): 305–338.

De Souza A.F., F Pedroni, E Oliveira, P.M. Ciarelli and W.F. Henrique. 2009. Automated multi-label text categorization with VG-RAM weightless neural networks. *Neurocomputing* 72 (10-12): 2209-2217.

Yansong F. and M. Lapata. 2008. Automatic image annotation using auxiliary text information. *Proceedings of ACL-08: HLT*. 272–280.

Hulth, A. 2003. Improved automatic keyword extraction given more linguistic knowledge. *Proceedings of the 2003 conference on Empirical methods in natural language processing* 216-223.

Jo T. 2003. Neural Based Approach to Keyword Extraction from Documents. *Kumar V., Gavrilova M.L., Tan C.J.K., L'Ecuyer P. (eds) Computational Science and Its Applications - ICCSA 2003. ICCSA 2003. Lecture Notes in Computer Science*. 2667. Berlin, Heidelberg. https://doi.org/10.1007/3-540-44839-X_49.

Johnson, L. Ballan and L. Fei-Fei. 2015. Love thy neighbors: Image annotation by exploiting image metadata. 2015. *IEEE International Conference on Computer Vision* 4624-4632, Santiago, Chile

Josi F., C. Wartena and J Charbonnier. 2018. Text-Based Annotation of Scientific Images Using Wikimedia Categories. *DEXA 2018. Communications in Computer and Information Science* 903.

Jobin K. V., A. Mondal, and C. V. Jawahar. 2019. Docfigure: A dataset for scientific document figure classification. *13th IAPR International Workshop on Graphics Recognition*, Sydney, NSW, Australia

Jeon J., V. Lavrenko, and R. Manmatha. 2003. Automatic image annotation and retrieval using cross-media relevance models. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. Toronto, Canada

Leong C. W and M. Rada and S. Hassan. 2010. Text Mining for Automatic Image Tagging. *Proceedings of the 23rd International Conference on Computational Linguistics* 23-27. Beijing, China

Murthy V.N., S. Maji and R. Manmatha. 2015. Automatic image annotation using deep learning representations. *ACM on International Conference on Multimedia Retrieval*. 603-606. New York, US

Mikolov T, K. Chen, G. Corrado and J. Dean, Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*

- Mayhew M.B., B. Chen, K.S. Ni. 2016. Assessing semantic information in convolutional neural network representations of images via image annotation. *IEEE International Conference on Image Processing (ICIP)* 2266–2270. Phoenix, AZ, US
- Mason, R. and E. Charniak. 2012. Apples to Oranges: Evaluating Image Annotations from Natural Language Processing Systems. 2012. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics ,Human Language Technologies*. Association for Computational Linguistics 172-181. Canada
- Mihalcea R. and P. Tarau. 2006. Textrank: Bringing order into texts. *Proceedings of Empirical Methods in Natural Language Processing*, Barcelona, Spain.
- Morris, D, E. Müller-Budack and R. Ewerth 2020. SlideImages: A Dataset for Educational Image Classification. *Jose J. et al. (eds) Advances in Information Retrieval. ECIR 2020. Lecture Notes in Computer Science*. 12036. https://doi.org/10.1007/978-3-030-45442-5_36
- Putthividhy D., H. T. Attias, and S. S. Nagarajan. 2010. Topic regression multi-modal latent dirichlet allocation for image annotation. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA.
- Prasad V. S. N., B. Siddiquie, J. Golbeck, and L. S. Davis. 2007. Classifying computer generated charts. *International Workshop on Content-Based Multimedia Indexing (CBMI)*. Talence, France.
- Rehurek, R. and P. Sojka. 2010. Software Framework for Topic Modelling with Large Corpora *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* 45-50. Valletta, Malta
- Salar, M., B. Roshanfekar, A. Zafarian and H. Asghari. 2018. A Language Processing Toolkit for Persian. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*. Miyazaki, Japan
- Savva M., N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. 2011. Revision: Automated classification, analysis and redesign of chart images. *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11)* 393–402. Association for Computing Machinery. DOI: <https://doi.org/10.1145/2047196.2047247>. New York, NY, USA
- Spärck Jones, K. 1972. A statistical interpretation of term specificity and its *Application in retrieval. Journal of Documentation* 28 (11-21)
- Siegel N., Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi. 2016. Figureseer: parsing result-figures in research papers. *ECCV, 2016. Lecture Notes in Computer Science*. 9911. Springer, Cham. https://doi.org/10.1007/978-3-319-46478-7_41
- Song L., M.Luo, J.Liu, L.Zhang, B.Qian, M.H.Li and Q.Zheng. 2016. Sparse multi-modal topical coding for image annotation, *Neurocomputing* 214 (2016): 162–174. <https://doi.org/10.1016/j.neucom.2016.06.005>.
- Sohmen J. Charbonnier, I. Blümel, C. Wartena, and L. Heller. 2018. *Figures in scientific open access publications. Digital Libraries for Open Knowledge, 22nd International Conference on Theory and Practice of Digital Libraries, TPD L . Proceedings. (E. Méndez, F. Crestani, C. Ribeiro, G. David, and J. C. Lopes, eds.)*. Lecture Notes in Computer Science 11057 (220–226), Springer. Porto,

Portugal

- McCarthy D. and N. Roberto. 2007. Evaluations (SemEval-2007 Task 10: English Lexical Substitution Task). *Proceedings of the Fourth International Workshop on Semantic Association for Computational Linguistics* 48-53. US.
- Matsuo Y. and M. Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools* 13 (01): 157-169.
- Onan A, S. Korukoğlu and H. Bulut. 2016. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications* 57 (232-247).
- Ramos J. 2003. Using tf-idf to determine word relevance in document queries. *Proceedings of the first instructional conference on machine learning* 1-4.
- Siddiqi S. and A. Sharan. 2015. Keyword and Keyphrase Extraction Techniques: A Literature Review. *International Journal of Computer Applications*. 109 (18-23).
- Salton G., A. Singhal, M. Mitra and C. Buckley. 1997. Automatic text structuring and summarization. *Information Processing & Management* 33(2):193-207.
- Tang B., X. Liu, J. Lei., M. Song, D. Tao, S. Sun, and F. Dong. 2016. Deepchart: Combining deep convolutional networks and deep belief networks in chart classification. *Signal Processing* 124 (156-161).
- Wang C., David Blei, and Li Fei-Fei. 2009. Simultaneous image classification and annotation. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, US.
- Wang C., L. Zhang and H.-J. Zhang. .2008. Learning to reduce the semantic gap in web image retrieval and annotation. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08*, ACM. 355–362. , New York, NY, US .<https://doi.org/10.1145/1390334.1390396>
- Wu J., Y. Yu, C. Huang and K. Yu. 2015. Deep multiple instance learning for image classification and auto-annotation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 3460–3469. IEEE. Boston, MA, USA
- Yang Y., W. Zhang and Y. Xie. 2015, Image automatic annotation via multi-view deep representation, *Journal of Visual Communication Image Representation* 368–377
- Zhou Y. P. and C. L. Tan. 2000. Hough technique for bar charts detection and recognition in document images. *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101) (ICIP)*, Vancouver, BC, Canada
- Wei Li and Andrew McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. *Proceedings of the International Conference on Machine learning* 577-584. Pittsburgh, Pennsylvania, US

طراحی یک روش برای برچسب زدن تصاویر استخراج شده از پارساهای موجود در پایگاه اطلاعات علمی ایران (گنج) □ ۵۹

پیوستها


```

36. # print(stem)
37. if "&" not in stem and stem.strip() != "":
38.     stemming.append(stem)
39. return stemming
40. def RemovePunc(token):
41.     punctuation = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~«»؟،'
42.     result=[]
43.     for word in token:
44.         for s in punctuation:
45.             if s in word:
46.                 word=word.replace(s,"")
47.             if word!="":
48.                 result.append(word)
49.     return result
50. def iter_block_items(parent):
51.     if isinstance(parent, _Document):
52.         parent_elm = parent.element.body
53.     elif isinstance(parent, _Cell):
54.         parent_elm = parent._tc
55.     elif isinstance(parent, _Row):
56.         parent_elm = parent._tr
57.     else:
58.         raise ValueError("something's not right")
59.     for child in parent_elm.iterchildren():
60.         if isinstance(child, CT_P):
61.             yield Paragraph(child, parent)
62.         elif isinstance(child, CT_Tbl):
63.             yield Table(child, parent)
64. def getDOCXparagraphs(path):
65.     document=Document(path)
66.     count=1
67.     clause=[]
68.     for block in iter_block_items(document):
69.         if isinstance(block, Paragraph):
70.             if block.text.strip() != "":
71.                 clause.append(block.text)
72.                 count+=1
73.         elif isinstance(block, Table):
74.             for row in block.rows:
75.                 row_cells = [_Cell(tc, block) for tc in row._tr._tblt]
76.                 for cell in row_cells:
77.                     for paragraph in cell.paragraphs:
78.                         if paragraph.text.strip() != "":
79.                             clause.append(paragraph.text)
80.                 count += 1
81.     return clause
82. def get_tokens(path):
83.     if "docx" in path:
84.         paragraphs = getDOCXparagraphs(path)
85.     elif "txt" in path:
86.         with open(path, "r", encoding="utf-8") as txtfile:
87.             paragraphs = txtfile.read().split("\n")
88.     else:
89.         return "NonText"
90.     Token=[]
91.     for item in paragraphs:
92.         # remove half-space
93.         item=re.sub(r"[\u200c-\u200f]", " ", item)
94.         normalized = Normalizer(item)
95.         split = SentenceSplitter_Tokenizer(normalized)
96.         sent_tokens=[]
97.         for sent in split:
98.             tokens = word_tokenize(sent)

```

```
99. clean=RemovePunc(tokens)
100. # print("clean:",clean)
101. if len(clean)!=0:
102.     sent_tokens.append(Stemmer(clean))
103. Token.extend(sent_tokens)
104. def save_tokens(path_src):
105.     print("In", path_src, "directory...")
106.     for file_name in os.listdir(path_src):
107.         # start=time.time()
108.
109.         fname = file_name.split(".")
110.         name = fname[0]
111.         # print(name)
112.         if len(fname) == 1: ## This is not a file... It can be skipped...
113.             continue
114.         if not os.path.exists(path_src + "/tokens/"):
115.             os.makedirs(path_src + "/tokens/")
116.
117.         if os.path.exists(path_src + "/tokens/" + name + ".txt"):
118.             continue
119.
120.         print("File " + file_name + " is processed.")
121.         try:
122.             token = get_tokens(path_src + "/" + file_name)
123.         except Exception as exp:
124.             print("Error:", exp)
125.             continue
126.         if token != "NonText":
127.             with open(path_src + "/tokens/" + name + ".txt", 'a', encoding="utf-8") as file:
128.                 file.write(str(token))
129. ##### Get corpous tekens
130. # Please set the data_path_src as corpus directory.
131. data_path_src="/home/myubuntu/Documents/ImageTagging/Corpus/fani_new/"
132.
133. # A directory, named "tokens" is made to save tokens of corpus
134. save_tokens(data_path_src)
135. # #####
```

۲- بخش دوم، تولید مدل آماری:

NgramModeling.py

```
1.  # -*- coding: utf-8 -*-
2.  Created on 1/30/21 6:46 PM
3.  # Creating ngram models...
4.  from nltk.util import ngrams
5.  import ast
6.  from nltk.text import TextCollection
7.  import pickle
8.  import os
9.  def generating_model(path,Text):
10.     if not os.path.exists(path + "models/"):
11.         os.makedirs(path + "models/")
12.     # unigram = []
13.     # for doc in Text:
14.     #     temp=[]
15.     #     for sent in doc:
16.     #         temp.extend(list(ngrams(sent, 1)))
17.     #     unigram.append(temp)
18.     # unigram_model = TextCollection(unigram)
19.     # with open(path + "models/" + 'uni_model.pkl', 'wb') as f:
20.     #     pickle.dump(unigram_model, f)
21.     # del unigram_model
22.     # del unigram
23.     bigram=[]
24.     for doc in Text:
25.         temp=[]
26.         for sent in doc:
27.             temp.extend(list(ngrams(sent, 2)))
28.         bigram.append(temp)
29.     bigram_model = TextCollection(bigram)
30.     with open(path + "models/" + 'bi_model.pkl', 'wb') as f:
31.         pickle.dump(bigram_model, f)
32.     del bigram_model
33.     del bigram
34.     trigram = []
35.     for doc in Text:
36.         temp=[]
37.         for sent in doc:
38.             temp.extend(list(ngrams(sent, 3)))
39.         trigram.append(temp)
40.     trigram_model = TextCollection(trigram)
41.     with open(path + "models/" + 'tri_model.pkl', 'wb') as f:
42.         pickle.dump(trigram_model, f)
43.
44.
45. # Please set the path_src as "tokens" directory (tokens of the corpus).
46. path="/home/myubuntu/Documents/ImageTagging/Corpus/Approach2/tokens/"
47. Text=[]
48.
49. # For each Doc, all paragraphs are read and a list variable (named 'content') extended with them.
50. # Another list variable (Text) contains all paragraphs of documents.
51. for idx, file in enumerate(sorted(os.listdir(path))):
```



```

50.     return result
51.
52. def iter_block_items(parent):
53.     if isinstance(parent, _Document):
54.         parent_elm = parent.element.body
55.     elif isinstance(parent, _Cell):
56.         parent_elm = parent._tc
57.     elif isinstance(parent, _Row):
58.         parent_elm = parent._tr
59.     else:
60.         raise ValueError("something's not right")
61.     for child in parent_elm.iterchildren():
62.         if isinstance(child, CT_P):
63.             yield Paragraph(child, parent)
64.         elif isinstance(child, CT_Tbl):
65.             yield Table(child, parent)
66. def getDOCXparagraphs(path):
67.     document=Document(path)
68.     count=1
69.     clause=[]
70.     for block in iter_block_items(document):
71.         if isinstance(block, Paragraph):
72.             if block.text.strip() != "":
73.                 clause.append(block.text)
74.                 count+=1
75.         elif isinstance(block, Table):
76.             for row in block.rows:
77.                 row_cells = [_Cell(tc, block) for tc in row._tr.tc_lst]
78.                 for cell in row_cells:
79.                     for paragraph in cell.paragraphs:
80.                         if paragraph.text.strip() != "":
81.                             clause.append(paragraph.text)
82.                             count += 1
83.     return clause
84. def get_tokens(path):
85.     if "docx" in path:
86.         paragraphs = getDOCXparagraphs(path)
87.     elif "txt" in path:
88.         with open(path, "r", encoding="utf-8") as txtfile:
89.             paragraphs = txtfile.read().split("\n")
90.     else:
91.         return "NonText"
92.     Token=[]
93.     for item in paragraphs:
94.         # remove half-space
95.         item=re.sub(r"[\u200c-\u200f]", " ", item)
96.         normalized = Normalizer(item)
97.         split = SentenceSplitter_Tokenizer(normalized)
98.         sent_tokens=[]
99.         for sent in split:
100.             tokens = word_tokenize(sent)
101.             clean=RemovePunc(tokens)
102.             # print("clean:",clean)
103.             if len(clean)!=0:
104.                 sent_tokens.append(Stemmer(clean))
105.             Token.extend(sent_tokens)
106.     return Token
107. def save_tokens(path_src):
108.     print("In", path_src, "directory...")
109.     for file_name in os.listdir(path_src):
110.         fname = file_name.split(".")
111.         name = fname[0]
112.         if len(fname) == 1: ## This is not file... It can be skipped...

```

```
113.         continue
114.     if not os.path.exists(path_src + "/tokens/"):
115.         os.makedirs(path_src + "/tokens/")
116.     if os.path.exists(path_src + "/tokens/" + name + ".txt"):
117.         continue
118.     print("File " + file_name + " is processed.")
119.     try:
120.         token = get_tokens(path_src + "/" + file_name)
121.     except Exception as exp:
122.         print("Error:", exp)
123.         continue
124.     if token != "NonText":
125.         with open(path_src + "/tokens/" + name + ".txt", 'a', encoding="utf-8") as file:
126.             file.write(str(token))
127. ##### Get tokens of each doc with caption
128. # Please set the cap_src as the root directory...
129. cap_path_src="/home/myubuntu/Documents/ImageTagging/Corpus/Approach2/images/"
130. for file in os.listdir(cap_path_src):
131.     # A directory, named "tokens" is made to save tokens of captions
132.     save_tokens(cap_path_src+file+"/")
```

۴- بخش چهارم، استخراج برجسبها:

AutoImageTagger.py

```
1.  # -*- coding: utf-8 -*-
2.
3.  # This code is written to read each image's info from an excel file.
4.  # Based on image's caption and ref paragraphs, automatic tags are extracted.
5.  # !!!Important: The stop words are selected based on our stop words list.
6.  import pickle
7.  from nltk.util import ngrams
8.  import operator
9.  import gensim
10. import numpy
11. from sklearn.metrics.pairwise import cosine_similarity
12. import os
13. from shutil import copy
14. import csv
15. import ast
16. import openpyxl
17. import pandas
18. import math
19. import re
20. def GetStopWord(sentences,doc_token,model_path):
21.     with open('stopwords') as f:
22.         sw = [re.sub(r"[\u200c-\u200f]", "", line.rstrip()).replace(" ", "") for line in f]
23.
24.     # print(sw)
25.     return sw
26.
27. def tf_idf_filtering(sentences,doc_token,model_path,sw):
28.
29.     content=sentences[1:]
30.
31.     tags = []
32.     ### n-gram (n=range(2,4)=2,3) are created...
33.     for i in range(2, 4):
34.
35.         if i==2:
36.             with open(model_path + 'bi_model.pkl', 'rb') as f:
37.                 model = pickle.load(f)
38.
39.         elif i==3:
40.
41.             with open(model_path + 'tri_model.pkl', 'rb') as f:
42.                 model = pickle.load(f)
43.
```

```

44. doc_ngram = []
45. for sent in doc_token:
46.     doc_ngram.extend(list(ngrams(sent, i)))
47.
48. tf_idf_list = []
49.
50. ng_list=[]
51. ngram = []
52.
53. for sent in content:
54.     ngram.extend(list(ngrams(sent, i)))
55. for ng in ngram:
56.     ListOfng=list(ng)
57.     ng_len=len(ng)
58.     valid=1
59.     numbers="۰۱۲۳۴۵۶۷۸۹"
60.     for k in range(ng_len):
61.
62.         for s in numbers:
63.             if s in ListOfng[k]:
64.                 ListOfng[k]=ListOfng[k].replace(s, "")
65.             if ListOfng[k] == "":
66.                 valid=0
67.                 break
68.             if re.sub(r"[\u200c-\u200f]", "", ListOfng[k].replace(" ", "")) in sw:
69.                 valid=0
70.                 break
71.             if valid==0:
72.                 continue
73.             ng=tuple(ListOfng)
74.             if ng not in ng_list:
75.                 ng_list.append(ng)
76.                 tf_idf = model.tf_idf(ng, doc_ngram)#In Doc
77.                 tf_idf_list.append((ng, tf_idf))
78. sorted_tfidf = sorted(tf_idf_list, key=operator.itemgetter(1), reverse=True)
79. # print(sorted_tfidf)
80. try:
81.     for j in range(10):# The top 20 (10 bigram + 10 trigram) phrases according to tf-idf
82.         tf_idf_words=[]
83.         for idx in range(i):
84.             ### Appending the ngrams to write results in a txt file...
85.             tf_idf_words.append(sorted_tfidf[j][0][idx])
86.             value = sorted_tfidf[j][1]
87.             tags.append((tf_idf_words, value))
88. except:
89.     pass
90. # filtered_tags[str(i)] = temp
91. del model
92. sorted_tags = sorted(tags, key=operator.itemgetter(1), reverse=True)
93. return sorted_tags
94. #####Initial path...
95. # Please set the doc_token_path as the tokens of the corpus directory.
96. doc_token_path="/home/myubuntu/Documents/ImageTagging/Corpus/Approach2/tokens/"
97. # Please set the img_path_src as the tokens of the images directory.
98. img_path_src="/home/myubuntu/Documents/ImageTagging/Corpus/Approach2/images/"
99. # Please set the article_id as the name of the article (thesis).
100. article_id="w189150"
101. # Please set the pic_id as the name of the image.
102. pic_id="شکل ۹"
103. # Please set the ngram_model_path as the path of the ngram model.
104. ngram_model_path = "/home/myubuntu/Documents/ImageTagging/Corpus/Approach2/tokens/models/"
105. #####Tagging
106. Text = []

```

```
107. # For each Doc, all paragraphs are read and a list variable (content) extended with them.
108. # Another list variable (Text) contains all paragraphs of documents.
109. if os.path.isfile(doc_token_path+article_id+".txt"):
110.     with open(doc_token_path+article_id+".txt" , 'r', encoding="utf-8") as f:
111.         doc_token=ast.literal_eval(f.read())
112.         # print(ast.literal_eval(item))
113.     with open(img_path_src+article_id+"/tokens/" + pic_id+".txt", 'r', encoding="utf-8") as f:
114.         sentences = ast.literal_eval(f.read())
115.     try:
116.         sw = GetStopWord(sentences, doc_token, ngram_model_path)
117.         print("tf_idf filtering...")
118.         tf_result = tf_idf_filtering(sentences, doc_token, ngram_model_path, sw)
119.         # print(tf_result)
120.
121.         tags_str = ""
122.         for item in tf_result:
123.             tags_str += "," + (" ".join(item[0]))
124.         print(tags_str)
125.     except Exception as exp:
126.         print("Error:",exp)
```

کد استخراج تصاویر از فایل ورد. در این کد تصاویر، زیرنویس و متن مرجع استخراج می شود.

```
136.
137. # -*- coding: utf-8 -*-
138. Created on Wed Jan 23 19:51:01 2019
139.
140. @author: Seddighi
141. @modified by: Fakhrzadeh
142. import uuid
143. import zipfile
144. import os
145. from lxml import etree
146. import re
147. from openpyxl import Workbook
148. from openpyxl import load_workbook
149. import datetime
150. from PIL import Image
151. import csv
152. import shutil
153. #from hazm import * #added by AF
154. #import re #added by AF
155. import platform
156. import subprocess
157. import win32com.client
158.
159. # XML Namespaces
160. nsmap = {
161.     'a': 'http://schemas.openxmlformats.org/drawingml/2006/main',
162.     'a14': 'http://schemas.microsoft.com/office/drawing/2010/main',
163.     'm': 'http://schemas.openxmlformats.org/officeDocument/2006/math',
164.     'mc': 'http://schemas.openxmlformats.org/markup-compatibility/2006',
165.     'o': 'urn:schemas-microsoft-com:office:office',
166.     'pic': 'http://schemas.openxmlformats.org/drawingml/2006/picture',
167.     'r': 'http://schemas.openxmlformats.org/officeDocument/2006/relationships',
168.     'v': 'urn:schemas-microsoft-com:vml',
169.     'w': 'http://schemas.openxmlformats.org/wordprocessingml/2006/main',
170.     'w10': 'urn:schemas-microsoft-com:office:word',
171.     'w14': 'http://schemas.microsoft.com/office/word/2010/wordml',
172.     'wne': 'http://schemas.microsoft.com/office/word/2006/wordml',
173.     'wp': 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing',
174.     'wp14': 'http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing',
175.     'wpc': 'http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas',
176.     'wpg': 'http://schemas.microsoft.com/office/word/2010/wordprocessingGroup',
177.     'wpi': 'http://schemas.microsoft.com/office/word/2010/wordprocessingInk',
178.     'wps': 'http://schemas.microsoft.com/office/word/2010/wordprocessingShape',
179. }
180. def doc2docx(src_path, dst_path):
181.     '''
182.     Convert a doc file to a docx file using Microsoft Word 2007 or higher in Windows
183.     and LibreOffice in Linux platform.
184.     src_path: relative path to the source file (path/*.doc)
185.     dst_path: relative path to the destination file (path/*.docx)
186.     '''
187.     status = 1 # Normal status
188.     current_platform = platform.system()
189.     if(current_platform == 'Linux'):
```

```

190.     if src_path.endswith('.doc'):
191.         try:
192.             dst_folder = os.path.dirname(dst_path)
193.             subprocess.call(['soffice', '--headless', '--convert-to', 'docx', '--outdir', dst_folder, src_path])
194.         except:
195.             status = 0 # required software has not installed
196.     elif(current_platform == 'Windows'):
197.         if src_path.endswith('.doc'):
198.             try:
199.                 word = win32com.client.Dispatch("Word.Application")
200.                 src_abs_path = os.path.abspath(src_path)
201.                 dst_abs_path = os.path.abspath(dst_path)
202.                 doc = word.Documents.Open(src_abs_path)
203.                 doc.SaveAs2(dst_abs_path, FileFormat=16) # file format for docx
204.                 doc.Close()
205.                 word.Quit()
206.             except:
207.                 status = -1 # required software has not installed
208.         else:
209.             # Mac: Darwin
210.             status = -2 # platform is not supported
211.
212.     return status
213.
214. def qn(tag):
215.     '''
216.     Stands for 'qualified name', a utility function to turn a namespace
217.     prefixed tag name into a Clark-notation qualified tag name for lxml. For
218.     example, ``qn('p:cSld')`` returns ``{http://schemas.../main}cSld``.
219.     Source: https://github.com/python-openxml/python-docx/
220.     '''
221.     prefix, tagroot = tag.split(':')
222.     uri = nsmmap[prefix]
223.     return '{{{}}}'.format(uri, tagroot)
224. def get_word_xml(docx_filename, xml_name = 'word/document.xml'):
225.     '''
226.     Reads a docx file and extracts the embedded xml file from it.
227.     Then it returns the xml tree of the target xml file.
228.     The target xml file is defined by xml_name parameter, which is
229.     document.xml (main xml file in docx format) by default.
230.     '''
231.     with open(docx_filename, 'rb') as f:
232.         ziph = zipfile.ZipFile(f)
233.         xml_content = ziph.read(xml_name)
234.         xml_tree = etree.fromstring(xml_content)
235.         ziph.close()
236.     return xml_tree
237. def update_xml_nsmmap(xml_tree):
238.     '''
239.     Updates the default xml namespaces using a new xml tree.
240.     It is used to include the new namespaces, which are not
241.     in our default list.
242.     '''
243.     for child in xml_tree.iter():
244.         nsmmap.update(child.nsmmap)
245.     return nsmmap
246. def none_handling(text, replace=u''):
247.     '''
248.     Replaces input (text) that is none with empty string (replace parameter).
249.     '''
250.     result = text if text is not None else replace
251.     return result
252. def elm2text(xml_element):

```

```
253. ''' A string representing the textual content of a xml element, with content
254. child elements like ``<w:tab/>`` translated to their Python
255. equivalent.
256. Adapted from: https://github.com/python-openxml/python-docx/
257. '''
258. text = u''
259. for child in xml_element.iterdescendants():
260.     if child.tag == qn('w:t'):
261.         t_text = child.text
262.         text += none_handling(t_text)
263.     elif child.tag == qn('w:noBreakHyphen'):
264.         text += '\u2011'
265.     elif child.tag == qn('w:tab'):
266.         text += '\t'
267.     elif child.tag in (qn('w:br'), qn('w:cr')):
268.         text += '\n'
269. return text
270.
271. def p_is_blank(paragraph):
272.     '''
273.     Checks whether a paragraph is blank or not. Ruterns true or false.
274.     '''
275.     result = False
276.
277.     # Matches any non-whitespace character; this is equivalent to the class [^\t\n\r\f\v]
278.     pattern = re.compile('\S')
279.
280.     if(etree.iselement(paragraph) and paragraph.tag == qn('w:p')):
281.         if(len(paragraph)):
282.             for child in paragraph.iterdescendants():
283.                 if child.tag == qn('w:drawing'):
284.                     result = False
285.                     break
286.                 elif child.tag == qn('w:noBreakHyphen'):
287.                     result = False
288.                     break
289.                 elif child.tag == qn('w:softHyphen'):
290.                     result = False
291.                     break
292.                 elif child.tag == qn('w:sym'):
293.                     result = False
294.                     break
295.                 elif child.tag == qn('w:bookmarkStart'):
296.                     result = True
297.                 elif child.tag == qn('w:bookmarkEnd'):
298.                     result = True
299.                 elif child.tag == qn('w:t'):
300.                     t_text = child.text
301.                     t_text = none_handling(t_text)
302.
303.                     # search for non-whitespace characters
304.                     if pattern.search(t_text) is not None:
305.                         result = False
306.                         break
307.                     else:
308.                         result = True
309.                 else:
310.                     result = True
311.
312.             else:
313.                 result = True
314.         else:
315.             result = None
```

```
316.
317.     return result
318.
319. def is_blank(element):
320.     '''
321.     Checks whether an element is blank or not. Ruterns true or false.
322.     '''
323.     result = False
324.
325.     if(etree.iselement(element)):
326.         # self and children which are paragraph
327.         paragraphs = list(element.iter(qn('w:p')))
328.         if(len(paragraphs)):
329.             for child in element.iter(qn('w:p')):
330.                 result = p_is_blank(child)
331.                 if(result == False):
332.                     break
333.         else:
334.             result = True
335.     else:
336.         result = None
337.
338.     return result
339.
340. def is_within_table(element):
341.     '''
342.     Checks whether an element is within table or not. Ruterns true or false.
343.     '''
344.     result = False
345.     if(etree.iselement(element)):
346.         tables = list(element.iterancestors(qn('w:tbl')))
347.         if(len(tables)):
348.             result = True
349.     else:
350.         result = None
351.
352.     return result
353.
354. def has_element(element, target_element):
355.     '''
356.     Checks whether an element has a terget element in its descendants or not.
357.     Ruterns true or false.
358.     '''
359.     result = False
360.     if(etree.iselement(element)):
361.         target_list = list(element.iterdescendants(qn(target_element)))
362.         if(len(target_list)):
363.             result = True
364.     else:
365.         result = None
366.
367.     return result
368.
369. def has_picture(element):
370.     '''
371.     Checks whether an element has a picture or not. Ruterns true or false.
372.     '''
373.     return has_element(element, 'a:blip')
374.
375. def has_caption(element):
376.     '''
377.     Checks whether an element has a caption or not. Ruterns true or false.
378.     '''
```

```
379. result = False
380. if(etree.iselement(element)):
381.     for caption in element.iterdescendants(qn('w:pStyle')):
382.         if(caption.get(qn('w:val')) == "Caption"):
383.             result = True
384.             break
385. else:
386.     result = None
387.
388. return result
389.
390. def get_cell_width(cell):
391.     '''
392.     Ruterns the width of a cell.
393.     '''
394.     width = 0
395.     if(etree.iselement(cell) and cell.tag == qn('w:tc')):
396.         for child in cell.iterdescendants(qn('w:tcW')):
397.             width = int(child.get(qn('w:w')))
398.             width = none_handling(width, 0)
399.     else:
400.         width = None
401.
402. return width
403.
404. def cells_intersection(dis, width, w_sum, tc_width):
405.     '''
406.     Computes the intersection of two cells, which are in two consecutive rows.
407.     Returns the start point (distance from left) and width of intersection
408.     part.
409.     inputs
410.     dis: cell1 distance from left
411.     width: cell1 width
412.     w_sum: cell2 distance from left plus its width
413.     tc_width: cell2 width
414.     outputs
415.     new_dis: intersection distance from left
416.     new_width: intersection width
417.     '''
418.     tc_left = w_sum - tc_width
419.     tc_right = w_sum
420.     if(tc_left < dis):
421.         new_dis = dis
422.         if(tc_right < dis + width):
423.             new_width = w_sum - dis
424.         else:
425.             new_width = width
426.     else:
427.         new_dis = w_sum - tc_width
428.         if(tc_right < dis + width):
429.             new_width = tc_width
430.         else:
431.             new_width = tc_width - w_sum + dis + width
432.
433.     return new_dis, new_width
434.
435. def add_text(info, text, delimiter=' | ', before=False):
436.     '''
437.     Adds a text to an info using a delimiter.
438.     before: add text before info
439.     '''
440.     if(info == u''):
441.         result = text
```

```

442. elif(text == u''):
443.     result = info
444. else:
445.     if(before):
446.         result = text + delimiter + info
447.     else:
448.         result = info + delimiter + text
449. return result
450.
451. def intercell_info(table, tr, dis, width, text_state = False):
452.     '''
453.     Extracts the information that is in the next row of a table and under a
454.     cell determined with dis and width.
455.     table: table element
456.     tr: row element which includes the target cell
457.     dis: cell distance from left
458.     width: cell width
459.     text_state: if cell has innertext then text_state will be true.
460.     '''
461.     info = u''
462.     caption_state = False
463.     if (etree.iselement(table) and table.tag == qn('w:tbl') and etree.iselement(tr) and tr.tag == qn('w:tr')):
464.         next_tr = tr.getnext()
465.         if(next_tr is not None):
466.             w_sum = 0
467.             for tc in next_tr.iterdescendants(qn('w:tc')):
468.                 tc_width = get_cell_width(tc)
469.                 w_sum += tc_width
470.                 if(w_sum > dis):
471.                     if(not has_picture(tc)):
472.                         if(not is_blank(tc)):
473.                             info = add_text(info, elm2text(tc))
474.                             if(has_caption(tc)):
475.                                 caption_state = True
476.                     else:
477.                         # check next row for blank cells
478.                         new_dis, new_width = cells_intersection(dis, width, w_sum, tc_width)
479.
480.                         new_info, temp_state = intercell_info(table, next_tr, new_dis, new_width)
481.                         new_info = none_handling(new_info)
482.                         info = add_text(info, new_info)
483.                         if(temp_state == True):
484.                             caption_state = True
485.                     else:
486.                         if(not text_state):
487.                             temp = elm2text(tc)
488.                             if(temp == u''):
489.                                 # check next row for picture cells which doesn't have text
490.                                 new_dis, new_width = cells_intersection(dis, width, w_sum, tc_width)
491.
492.                                 new_info, temp_state = intercell_info(table, next_tr, new_dis, new_width)
493.                                 new_info = none_handling(new_info)
494.                                 info = add_text(info, new_info)
495.                                 if(temp_state == True):
496.                                     caption_state = True
497.                             else:
498.                                 info = add_text(info, temp)
499.                                 if(has_caption(tc)):
500.                                     caption_state = True
501.
502.             if(w_sum < dis + width):
503.                 # more cells must processed
504.                 continue

```

```

505.         else:
506.             break
507.     else:
508.         info = None
509.         caption_state = None
510.
511.     return info, caption_state
512.
513. def is_cell_vertical(cell):
514.     '''
515.     Checks whether a cell is vertical or not. Returns true or false and the
516.     state of the vertical cell.
517.     '''
518.     result = False
519.     state = None
520.     if(etree.iselement(cell) and cell.tag == qn('w:tc')):
521.         vertical= list(cell.iterdescendants(qn('w:vMerge')))
522.         if(len(vertical)):
523.             result = True
524.             for child in cell.iterdescendants(qn('w:vMerge')):
525.                 value = child.get(qn('w:val'))
526.                 if(value == 'restart'):
527.                     state = 'start'
528.                 else:
529.                     state = 'continue'
530.         else:
531.             result = None
532.
533.     return result, state
534.
535. def get_next_info(element):
536.     '''
537.     Returns the text information that comes after an element.
538.     '''
539.     # element can be either paragraph or table
540.     info = u''
541.     caption_state = False
542.     if(etree.iselement(element)):
543.         for next_element in element.itsiblings():
544.             if(next_element.tag == qn('w:p')):
545.                 paragraph = next_element
546.                 if(not has_picture(paragraph)):
547.                     if(not is_blank(paragraph)):
548.                         info = elm2text(paragraph)
549.                         if(has_caption(paragraph)):
550.                             caption_state = True
551.                         break
552.             else:
553.                 continue
554.         else:
555.             continue
556.     elif(next_element.tag == qn('w:tbl')):
557.         table = next_element
558.         for tr in table.iterdescendants(qn('w:tr')):
559.             if(not has_picture(tr)):
560.                 if(not is_blank(tr)):
561.                     info = elm2text(tr)
562.                     if(has_caption(tr)):
563.                         caption_state = True
564.                     break
565.             else:
566.                 continue
567.         else:

```

```

568.         temp = elm2text(tr)
569.         if (temp == u''):
570.             continue
571.         else:
572.             info = temp
573.             if(has_caption(tr)):
574.                 caption_state = True
575.                 break
576.         else:
577.             break
578.
579.     else:
580.         info = None
581.         caption_state = None
582.
583.     return info, caption_state
584.
585. def get_next_caption(element):
586.     '''
587.     Returns the caption information that comes after an element.
588.     '''
589.     # element can be either paragraph or table
590.     info = u''
591.     caption_state = False
592.     if(etree.iselement(element)):
593.         for next_element in element.itorsiblings():
594.             if(next_element.tag == qn('w:p')):
595.                 paragraph = next_element
596.                 if(not has_picture(paragraph)):
597.                     if(not is_blank(paragraph)):
598.                         caption = list(paragraph.iterdescendants(qn('w:pStyle')))
599.                         if(len(caption)):
600.                             caption = caption[0]
601.                             if(caption.get(qn('w:val')) == "Caption"):
602.                                 info = elm2text(paragraph)
603.                                 caption_state = True
604.                                 break
605.                         else:
606.                             break
607.                     else:
608.                         break
609.                 else:
610.                     continue
611.             else:
612.                 continue
613.         elif(next_element.tag == qn('w:tbl')):
614.             table = next_element
615.             for tr in table.iterdescendants(qn('w:tr')):
616.                 if(not has_picture(tr)):
617.                     if(not is_blank(tr)):
618.                         for paragraph in tr.iterdescendants(qn('w:p')):
619.                             caption = list(paragraph.iterdescendants(qn('w:pStyle')))
620.                             if(len(caption)):
621.                                 caption = caption[0]
622.                                 if(caption.get(qn('w:val')) == "Caption"):
623.                                     info = elm2text(paragraph)
624.                                     caption_state = True
625.                                     break
626.                             else:
627.                                 continue
628.                         else:
629.                             continue
630.                     break

```



```

694.         w_sum += tc_width
695.         if(w_sum > dis):
696.             if(tc_width == width):
697.                 tc_result, tc_state = is_cell_vertical(tc)
698.                 if(tc_result and tc_state == 'continue'):
699.                     tr = next_tr
700.                 else:
701.                     result = False
702.                     break
703.             else:
704.                 result = False
705.                 break
706.         else:
707.             result = False
708.             break
709.
710.         if(intrainfo == u''):
711.             text_state = False
712.         else:
713.             text_state = True
714.         interinfo, inter_caption_state = intercell_info(table, tr, dis, width, text_state)
715.         #-----
716.         # extract table next info
717.         #-----
718.         tableinfo, table_caption_state = get_next_caption(table)
719.     else:
720.         intrainfo = None
721.         interinfo = None
722.         tableinfo = None
723.         intra_caption_state = None
724.         inter_caption_state = None
725.         table_caption_state = None
726.
727.     return intrainfo, interinfo, tableinfo, intra_caption_state, inter_caption_state, table_caption_state
728.
729. def detect_language(text):
730.     '''
731.     Detects the text language.
732.     '''
733.     lang = 'English'
734.
735.     # Persian alphabet
736.     persian = re.compile('[\u0621-\u0628\u062A-\u063A\u0641-\u0642\u0644-\u0648\u064E-\u0651\u0655\u067E\u0686\u0698\u06A9\u06AF\u06BE\u06CC]')
737.
738.     # search for persian characters
739.     if persian.search(text) is not None:
740.         lang = 'Persian'
741.
742.     return lang
743.
744. def get_file_list(path, file_type='docx'):
745.     '''
746.     Returns a list of files in a certain path, which have a common file type.
747.     '''
748.     file_list = []
749.     for file in os.listdir(path):
750.         if file.endswith("." + str(file_type)):
751.             file_list.append(file)
752.     return file_list
753.
754. def desc2vec(pic_desc, delimiter=' | '):
755.     '''

```

```

756. Combines the text of pic_desc using a delimiter to vectorize the information.
757. '''
758. pic_direct_desc = pic_desc['pic_direct_desc']
759. pic_intermediate_desc = pic_desc['pic_intermediate_desc']
760. pic_general_desc = pic_desc['pic_general_desc']
761. desc = u''
762. for i in reversed(range(len(pic_general_desc))):
763.     desc = add_text(desc, pic_general_desc[i], delimiter)
764.     desc = add_text(desc, pic_intermediate_desc[i], delimiter)
765.     desc = add_text(desc, pic_direct_desc[i], delimiter)
766. return desc
767.
768. def unique(item):
769.     '''
770.     Returns a list of unique itmes.
771.     '''
772.     result = []
773.     length = len(item)
774.     for i in range(length):
775.         state = 1 # item is unique
776.         for j in range(i+1,length):
777.             if item[i] == item[j]:
778.                 state = 0
779.                 break
780.         if state:
781.             result.append(item[i])
782.     return result
783.
784. def img_extract(docx_filename, img_dir='media', save_policy='normal', file_per_folder=1000):
785.     '''
786.     Extracts the images and their corresponding captions (main function).
787.     docx_filename: document filename
788.     img_dir: image directory that will be used to store extracted images.
789.     save_policy: policy for saving images (normal or multiple folder)
790.     file_per_folder: number of files in each folder if save_policy is multiple
791.     '''
792.     if not os.path.exists(docx_filename):
793.         print('File {} does not exist.'.format(docx_filename))
794.     else:
795.         # read document file
796.         xml_tree = get_word_xml(docx_filename)
797.
798.         # update namespaces
799.         global nsmap
800.         nsmap = update_xml_nsmap(xml_tree)
801.
802.         # find id and description of pictures which have caption
803.         pics = []
804.         lang = ''
805.         for paragraph in xml_tree.iter(qn('w:p')):
806.             for child in paragraph.iterdescendants(qn('a:blip')):
807.                 pic_id = child.get(qn('r:embed'))
808.                 pic_desc = u''
809.
810.                 # paragraph is within a table
811.                 if(is_within_table(paragraph)):
812.                     pic_direct_desc = []
813.                     pic_intermediate_desc = []
814.                     pic_general_desc = []
815.                     empty = []
816.                     cell = paragraph.getparent()
817.                     element = paragraph
818.                     lang = 'English'

```

```

819.         caption_state = False
820.         for table in paragraph.iterancestors(qn('w:tbl')):
821.             intrainfo, interinfo, tableinfo, intra_caption_state, inter_caption_state, table_caption_state =
cell_info(table, cell, element)
822.             intrainfo = none_handling(intrainfo)
823.             interinfo = none_handling(interinfo)
824.             tableinfo = none_handling(tableinfo)
825.             if(detect_language(intrainfo) == 'Persian' or detect_language(interinfo) == 'Persian' or
detect_language(tableinfo) == 'Persian'):
826.                 lang = 'Persian'
827.                 if(intra_caption_state or inter_caption_state or table_caption_state):
828.                     caption_state = True
829.                     pic_direct_desc.append(intrainfo)
830.                     pic_intermediate_desc.append(interinfo)
831.                     pic_general_desc.append(tableinfo)
832.                     empty.append(u'')
833.                     cell = table.getparent()
834.                     element = table
835.                     if(pic_direct_desc != empty or pic_intermediate_desc != empty or pic_general_desc != empty):
836.                         if(caption_state):
837.                             pic_desc = {'pic_direct_desc': pic_direct_desc, 'pic_intermediate_desc':
pic_intermediate_desc, 'pic_general_desc': pic_general_desc}
838.                             else:
839.                                 pic_desc = u''
840.                             else:
841.                                 pic_desc = u''
842.                             else:
843.                                 # paragraph is within body
844.                                 pic_desc, _ = get_next_caption(paragraph)
845.                                 pic_desc = none_handling(pic_desc)
846.                                 lang = detect_language(pic_desc)
847.
848.                                 if(pic_desc != u''):
849.                                     pics.append([pic_id, pic_desc, lang])
850.
851. # remove duplicates
852. pics = unique(pics)
853.
854. # find filenames of pictures
855. xml_name = 'word/_rels/document.xml.rels'
856. rels_tree = get_word_xml(docx_filename, xml_name = xml_name)
857. pics_info = []
858. for pic in pics:
859.     pid = pic[0]
860.     pdesc = pic[1]
861.     lang = pic[2]
862.     puid = str(uuid.uuid4()).replace('-', '')
863.     for child in rels_tree.iter():
864.         if child.get('Id') == pid:
865.             path = child.get('Target')
866.             pics_info.append({'path': path, 'id': pid, 'desc': pdesc, 'lang': lang, 'uid': puid})
867.
868. # create img_dir
869. if save_policy == 'multiple':
870.     # computing number of folders for saving images under multiple folder policy
871.     num_pics = len(pics_info)
872.     num_folder = 0
873.     if num_pics % file_per_folder > 0:
874.         num_folder = round(num_pics / file_per_folder) + 1
875.     else:
876.         num_folder = round(num_pics / file_per_folder)
877.     # create folders
878.     img_dir_base = img_dir

```

```

879.     for idx in range(1,num_folder + 1):
880.         img_dir = img_dir_base + '_' + str(idx)
881.         if not os.path.exists(img_dir):
882.             try:
883.                 os.makedirs(img_dir)
884.             except OSError:
885.                 print("Unable to create image directory {}".format(img_dir))
886.                 return None
887.
888.         # extract pictures and write them to folders
889.         with open(docx_filename, 'rb') as f:
890.             ziph = zipfile.ZipFile(f)
891.             end_range = idx*file_per_folder
892.             if idx == num_folder:
893.                 end_range = num_pics
894.             for pic in pics_info[(idx-1)*file_per_folder:end_range]:
895.                 src_path = "word/" + pic["path"]
896.                 dst_path = os.path.join(img_dir, os.path.basename(pic["path"]))
897.
898.                 with open(dst_path, "wb") as dst_f:
899.                     dst_f.write(ziph.read(src_path))
900.
901.             ziph.close()
902.         else:
903.             if not os.path.exists(img_dir):
904.                 try:
905.                     os.makedirs(img_dir)
906.                 except OSError:
907.                     print("Unable to create image directory {}".format(img_dir))
908.                     return None
909.
910.             # extract pictures and write them to img_dir
911.             with open(docx_filename, 'rb') as f:
912.                 ziph = zipfile.ZipFile(f)
913.                 for pic in pics_info:
914.                     src_path = "word/" + pic["path"]
915.                     dst_path = os.path.join(img_dir, os.path.basename(pic["path"]))
916.
917.                     with open(dst_path, "wb") as dst_f:
918.                         dst_f.write(ziph.read(src_path))
919.
920.                 ziph.close()
921.
922.         return pics_info
923.
924. def add_text_to_array(array, text):
925.     """
926.     Add a text to all elements of an array
927.     """
928.     new_array = array.copy()
929.     for idx in range(len(new_array)):
930.         new_array[idx] += text
931.
932.     return new_array
933.
934. def is_special(char):
935.     """
936.     Check if char is special (hyphens, period, underscore and space) or not
937.     """
938.     special_chars = [' ', '_', '.', '\u002d', '\u005be', '\u005c', '\u0020', '\u0021', '\u0022', '\u0023', '\u0024', '\u0025', '\u0026', '\u0027', '\u0028', '\u0029', '\u002a', '\u002b', '\u002c', '\u002d', '\u002e', '\u002f', '\u003a', '\u003b', '\u003c', '\u003e', '\u003f', '\u0040', '\u0041', '\u0042', '\u0043', '\u0044', '\u0045', '\u0046', '\u0047', '\u0048', '\u0049', '\u004a', '\u004b', '\u004c', '\u004d', '\u004e', '\u004f', '\u0050', '\u0051', '\u0052', '\u0053', '\u0054', '\u0055', '\u0056', '\u0057', '\u0058', '\u0059', '\u005a', '\u005b', '\u005c', '\u005d', '\u005e', '\u005f', '\u0060', '\u0061', '\u0062', '\u0063', '\u0064', '\u0065', '\u0066', '\u0067', '\u0068', '\u0069', '\u006a', '\u006b', '\u006c', '\u006d', '\u006e', '\u006f', '\u0070', '\u0071', '\u0072', '\u0073', '\u0074', '\u0075', '\u0076', '\u0077', '\u0078', '\u0079', '\u007a', '\u007b', '\u007c', '\u007d', '\u007e', '\u007f', '\u0080', '\u0081', '\u0082', '\u0083', '\u0084', '\u0085', '\u0086', '\u0087', '\u0088', '\u0089', '\u008a', '\u008b', '\u008c', '\u008d', '\u008e', '\u008f', '\u0090', '\u0091', '\u0092', '\u0093', '\u0094', '\u0095', '\u0096', '\u0097', '\u0098', '\u0099', '\u009a', '\u009b', '\u009c', '\u009d', '\u009e', '\u009f', '\u00a0', '\u00a1', '\u00a2', '\u00a3', '\u00a4', '\u00a5', '\u00a6', '\u00a7', '\u00a8', '\u00a9', '\u00aa', '\u00ab', '\u00ac', '\u00ad', '\u00ae', '\u00af', '\u00b0', '\u00b1', '\u00b2', '\u00b3', '\u00b4', '\u00b5', '\u00b6', '\u00b7', '\u00b8', '\u00b9', '\u00ba', '\u00bb', '\u00bc', '\u00bd', '\u00be', '\u00bf', '\u00c0', '\u00c1', '\u00c2', '\u00c3', '\u00c4', '\u00c5', '\u00c6', '\u00c7', '\u00c8', '\u00c9', '\u00ca', '\u00cb', '\u00cc', '\u00cd', '\u00ce', '\u00cf', '\u00d0', '\u00d1', '\u00d2', '\u00d3', '\u00d4', '\u00d5', '\u00d6', '\u00d7', '\u00d8', '\u00d9', '\u00da', '\u00db', '\u00dc', '\u00dd', '\u00de', '\u00df', '\u00e0', '\u00e1', '\u00e2', '\u00e3', '\u00e4', '\u00e5', '\u00e6', '\u00e7', '\u00e8', '\u00e9', '\u00ea', '\u00eb', '\u00ec', '\u00ed', '\u00ee', '\u00ef', '\u00f0', '\u00f1', '\u00f2', '\u00f3', '\u00f4', '\u00f5', '\u00f6', '\u00f7', '\u00f8', '\u00f9', '\u00fa', '\u00fb', '\u00fc', '\u00fd', '\u00fe', '\u00ff']
939.     # source: https://www.compart.com/en/unicode/category/Pd
940.

```

```
941. for special_char in special_chars:
942.     if(char == special_char):
943.         return True
944.
945. return False
946.
947. def strip_special(chars):
948.     '''
949.     Strip special charactrs
950.     '''
951.     new_chars = chars
952.     for i in range(len(chars)):
953.         if(is_special(chars[i])):
954.             new_chars = new_chars[1:]
955.         else:
956.             break
957.
958.     for i in range(len(chars)):
959.         if(is_special(chars[-(i+1)])):
960.             new_chars = new_chars[:-1]
961.         else:
962.             break
963.
964.     return new_chars
965.
966. def find_text(docx_filename, text):
967.     '''
968.     Find a text in a document and return all paragraphs which contain the text.
969.     result: retrived paragraphs attached using "|" delimiter.
970.     '''
971.     result = u''
972.     if not os.path.exists(docx_filename):
973.         print('File {} does not exist.'.format(docx_filename))
974.     else:
975.         # read document file
976.         xml_tree = get_word_xml(docx_filename)
977.
978.         # update namespaces
979.         global nsmap
980.         nsmap = update_xml_nsmap(xml_tree)
981.
982.         # find text
983.         for paragraph in xml_tree.iter(qn('w:p')):
984.             p_text = elm2text(paragraph)
985.             if(text in p_text):
986.                 result = add_text(result, p_text)
987.
988.     return result
989.
990. def find_exact_text(docx_filename, text, max_num = 0):
991.     '''
992.     Find exact text in a document and return all paragraphs which contain the text.
993.     max_num: maximum number of paragraphs that will be extracted before and after the text.
994.     result: retrived paragraphs attached using "|" delimiter.
995.     '''
996.     result = u''
997.     if not os.path.exists(docx_filename):
998.         print('File {} does not exist.'.format(docx_filename))
999.     else:
1000.         # read document file
1001.         xml_tree = get_word_xml(docx_filename)
1002.
1003.         # update namespaces
```

```

1004. global nsmap
1005. nsmap = update_xml_nsmap(xml_tree)
1006.
1007. # find exact text
1008. for paragraph in xml_tree.iter(qn('w:p')):
1009.     p_text = elm2text(paragraph)
1010.
1011.     # find all occurrences in a paragraph
1012.     start = 0 # Starting position for search
1013.     index = p_text.find(text, start)
1014.     text_length = len(text)
1015.     p_length = len(p_text)
1016.     while(index != -1):
1017.         # if exists next characters
1018.         if(index + text_length < p_length):
1019.             # check next characters
1020.             for next_idx in range(index + text_length, p_length):
1021.                 if(is_special(p_text[next_idx]) or (not p_text[next_idx].isprintable())):
1022.                     continue
1023.                 elif(not p_text[next_idx].isdigit()):
1024.                     # extract previous paragraphs
1025.                     before_p_text = u''
1026.                     iter = 0
1027.                     for prev_paragraph in paragraph.itorsiblings(tag=qn('w:p'), preceding=True):
1028.                         if(iter < max_num):
1029.                             if(not is_blank(prev_paragraph) and not has_picture(prev_paragraph)):
1030.                                 prev_p_text = elm2text(prev_paragraph)
1031.                                 before_p_text = add_text(before_p_text, prev_p_text, delimiter=', ', before=True)
1032.                                 iter += 1
1033.                         else:
1034.                             break
1035.
1036.                     # extract next paragraphs
1037.                     after_p_text = u''
1038.                     iter = 0
1039.                     for next_paragraph in paragraph.itorsiblings(tag=qn('w:p')):
1040.                         if(iter < max_num):
1041.                             if(not is_blank(next_paragraph) and not has_picture(next_paragraph)):
1042.                                 next_p_text = elm2text(next_paragraph)
1043.                                 after_p_text = add_text(after_p_text, next_p_text, delimiter=', ')
1044.                                 iter += 1
1045.                         else:
1046.                             break
1047.
1048.                     all_p_text = before_p_text + ' ' + p_text + ' ' + after_p_text
1049.                     result = add_text(result, all_p_text)
1050.                     break
1051.                 elif(p_text[next_idx].isdigit()):
1052.                     break
1053.
1054.             start = index + text_length
1055.             index = p_text.find(text, start)
1056.         else:
1057.             index = -1
1058.
1059.     return result
1060.
1061. def extract_related_pic_text(docx_filename, pic_title, extra_paragraph = 0):
1062.     '''
1063.     Extract all texts related to a picture title in a document
1064.     '''
1065.     title = pic_title.strip()
1066.     start_texts = ['اشکل', 'تصویر', 'نمودار', 'دیاگرام', 'نقشه', 'فلوچارت']

```

```

1067. pic_numbers = []
1068. result = u''
1069. for start_text in start_texts:
1070.     if(title.startswith(start_text)):
1071.         length = len(start_text)
1072.         # remove start text
1073.         headless_title = title[length:]
1074.         # remove extra spaces form right and left
1075.         headless_title = headless_title.strip()
1076.
1077.         # if text starts with letter
1078.         if(headless_title[0].isalpha()):
1079.             text_array = headless_title.split()
1080.             pic_numbers.append(start_text + ' ' + text_array[0])
1081.             pic_numbers.append(start_text + text_array[0])
1082.         # if text starts with something else
1083.         else:
1084.             pic_number = u''
1085.             for i in range(len(headless_title)):
1086.                 if(headless_title[i].isdigit() or is_special(headless_title[i]) or (not headless_title[i].isprintable())):
1087.                     pic_number += headless_title[i]
1088.                 else:
1089.                     break
1090.
1091.             # delete extra spaces (TODO: delete other extra special characters)
1092.             while(pic_number.find(' ') != -1):
1093.                 pic_number = pic_number.replace(' ', '')
1094.
1095.             # strip special characters
1096.             pic_number = strip_special(pic_number)
1097.
1098.             pic_numbers.append(pic_number[0])
1099.             # each character in pic_number is digit or special (space, hyphen, ...) or nonprintable (\u200a-\u200f,
1100.             ...)
1101.             for i in range(1, len(pic_number)):
1102.                 if(pic_number[i] == ' '):
1103.                     # print('space')
1104.                     if(pic_number[i - 1].isdigit() and pic_number[i + 1].isdigit()):
1105.                         pic_numbers = add_text_to_array(pic_numbers, pic_number[i])
1106.                     else:
1107.                         branch = add_text_to_array(pic_numbers, pic_number[i])
1108.                         pic_numbers.extend(branch)
1109.
1110.                     # print('pic-numbers:', pic_numbers)
1111.
1112.                 elif(is_special(pic_number[i])):
1113.                     # print('spacial')
1114.                     if(pic_number[i - 1] != ' '):
1115.                         # print('char is special with a non space back:', pic_numbers)
1116.                         branch1 = add_text_to_array(pic_numbers, pic_number[i])
1117.                         branch2 = add_text_to_array(pic_numbers, ' ' + pic_number[i])
1118.                         branch1.extend(branch2)
1119.                         pic_numbers = branch1.copy()
1120.                     else:
1121.                         pic_numbers = add_text_to_array(pic_numbers, pic_number[i])
1122.
1123.                     # print('pic-numbers:', pic_numbers)
1124.
1125.                 elif(pic_number[i].isdigit()):
1126.                     # print('digit')
1127.                     if(pic_number[i - 1] != ' ' and (is_special(pic_number[i - 1]) or not pic_number[i -
1128.                     1].isprintable())):
1129.                         branch1 = add_text_to_array(pic_numbers, pic_number[i])

```

```

1128.         branch2 = add_text_to_array(pic_numbers, ' ' + pic_number[i])
1129.         branch1.extend(branch2)
1130.         pic_numbers = branch1.copy()
1131.     else:
1132.         pic_numbers = add_text_to_array(pic_numbers, pic_number[i])
1133.
1134.         # print('pic-numbers:', pic_numbers)
1135.     # non printables
1136.     else:
1137.         # print('non printables')
1138.         branch = add_text_to_array(pic_numbers, pic_number[i])
1139.         pic_numbers.extend(branch)
1140.         # print('pic-numbers:', pic_numbers)
1141.
1142.     all_pic_numbers = []
1143.     for elm in pic_numbers:
1144.         all_pic_numbers.append(start_text + ' ' + elm)
1145.         all_pic_numbers.append(start_text + elm)
1146.     pic_numbers = all_pic_numbers.copy()
1147.     #print(pic_numbers)
1148.
1149.     for text in pic_numbers:
1150.         # retrieved_texts = find_text(docx_filename, text)
1151.         retrieved_texts = find_exact_text(docx_filename, text , extra_paragraph)
1152.         result = add_text(result, retrieved_texts)
1153.         result=add_text(result,pic_numbers[1]) #added by AF july/05
1154.     return result
1155. return result
1156.
1157.def create_image_db(path_src, path_dst, db_title='ImageDB', thumbnail_size=(128,128),
    save_policy='normal', file_per_folder=1000 , extra_paragraph = 0):
1158.    '''
1159.    Creates an image database in excel and csv format. Extracts all images,
1160.    their corresponding information, and image thumbnails.
1161.    '''
1162.    # Creating an excel file for saving information
1163.    wb = Workbook()
1164.    ws = wb.active
1165.    ws.title = db_title
1166.
1167.    # information header
1168.    row = 1
1169.    # the last element fig number is added by AF
1170.    header = ['id', 'article_id', 'uid', 'title_fa', 'title_en', 'description', 'size', 'height', 'width', 'created_at',
    'updated_at', 'retrieved_texts']
1171.    for j in range(len(header)):
1172.        ws.cell(row=row, column=j+1, value=header[j])
1173.
1174.    # Converting doc files to docx files
1175.    doc_files = get_file_list(path_src, 'doc')
1176.    state = 1 # normal convert
1177.    for doc in doc_files:
1178.        file_type = 'doc'
1179.        doc_name_only = doc[:-(len(file_type)+1)]
1180.        doc_src_path = path_src + '/' + doc
1181.        docx_dst_path = path_src + '/' + doc_name_only + '.docx'
1182.        state = doc2docx(doc_src_path, docx_dst_path)
1183.
1184.    if(state == 0):
1185.        print("LibreOffice must be installed on this machine.")
1186.    elif(state == -1):
1187.        print("Microsoft Word 2007 or higher must be installed on this machine.")
1188.    elif(state == -2):

```

```

1189.     print("Platform is not supported.")
1190.
1191.     path_dst_temp = path_dst+'/'+temp'
1192.     file_type = 'docx'
1193.
1194.     filenames = get_file_list(path_src, 'docx')
1195.     counter = 0
1196.     withoutC = 0 #AF
1197.
1198.     # processing each file
1199.     for file in filenames:
1200.         try:
1201.             print('file {} is processed'.format(file)) #AF
1202.             docx_name_only = file[:-(len(file_type)+1)]
1203.             docx_filename = path_src + '/' + file
1204.             img_dir = path_dst + '/' + docx_name_only
1205.             img_dir_temp = path_dst_temp + '/' + docx_name_only
1206.             pics_info = img_extract(docx_filename, img_dir_temp, save_policy, file_per_folder)
1207.
1208.             pics_info = unique(pics_info)
1209.
1210.             if not pics_info: #AF
1211.                 withoutC = withoutC + 1 #AF
1212.
1213.             if save_policy == 'multiple':
1214.                 # computing number of folders for saving images under multiple folder policy
1215.                 num_pics = len(pics_info)
1216.                 num_folder = 0
1217.                 if num_pics % file_per_folder > 0:
1218.                     num_folder = round(num_pics / file_per_folder) + 1
1219.                 else:
1220.                     num_folder = round(num_pics / file_per_folder)
1221.
1222.                 img_dir_base = img_dir
1223.                 img_dir_temp_base = img_dir_temp
1224.                 for idx in range(1, num_folder + 1):
1225.                     img_dir = img_dir_base + '_' + str(idx)
1226.                     img_dir_temp = img_dir_temp_base + '_' + str(idx)
1227.                     end_range = idx*file_per_folder
1228.                     if idx == num_folder:
1229.                         end_range = num_pics
1230.                     for pic in pics_info[(idx-1)*file_per_folder:end_range]:
1231.                         row += 1
1232.                         desc = pic['desc']
1233.                         lang = pic['lang']
1234.                         title_fa = ''
1235.                         title_en = ''
1236.                         if (type(desc) is str):
1237.                             if lang == 'Persian':
1238.                                 title_fa = desc
1239.                                 title_en = ''
1240.                             else:
1241.                                 title_fa = ''
1242.                                 title_en = desc
1243.                         elif (type(desc) is dict):
1244.                             if lang == 'Persian':
1245.                                 title_fa = desc2vec(desc)
1246.                                 title_en = ''
1247.                             else:
1248.                                 title_fa = ''
1249.                                 title_en = desc2vec(desc)
1250.
1251.                     title_fa = title_fa.strip()

```

```

1252.         title_en = title_en.strip()
1253.         # pic_number = extract_pic_number(title_fa)
1254.         retrieved_texts = extract_related_pic_text(docx_filename, title_fa, extra_paragraph)
1255.         rt = retrieved_texts.split(' | ')
1256.         picId=rt[-1] #added by AF july/05
1257.         #print( picId)
1258.
1259.
1260.         # *** retrieve only the first mention (after "List of Figures") of the figure title in document
1261.         if(not retrieved_texts == u''):
1262.             retrieved_texts = retrieved_texts.split(' | ')
1263.             if(len(retrieved_texts)>1):
1264.                 retrieved_texts = retrieved_texts[1]
1265.             else:
1266.                 retrieved_texts = retrieved_texts[0]
1267.
1268.
1269.
1270.         pic_name = pic['path'][6:]
1271.         img_filename = img_dir_temp + '/' + pic_name
1272.
1273.         # image file size in KB
1274.         img_size = round(os.path.getsize(img_filename)/1024,1)
1275.
1276.         # create img_dir
1277.         if not os.path.exists(img_dir):
1278.             try:
1279.                 os.makedirs(img_dir)
1280.             except OSError:
1281.                 print("Unable to create image directory {}".format(img_dir))
1282.
1283.         # image change type to jpeg and create thumbnails
1284.         # image dimensions in pixel
1285.         with Image.open(img_filename) as image:
1286.             width, height = image.size
1287.             #pic_name = pic["uid"] + '.jpg'
1288.             pic_name = picId + '.jpg' #Added by Af
1289.             img_filename = img_dir + '/' + pic_name
1290.             image.convert('RGB').save(os.path.join(img_dir, pic_name), 'JPEG')
1291.             image.thumbnail(thumbnail_size)
1292.             #image.convert('RGB').save(os.path.join(img_dir, "thumbnail" + pic_name), 'JPEG')
1293.
1294.
1295.         related_txt=picId+'.txt' #added by AF july/05
1296.         imageTxtName=img_dir + '/' + related_txt #added by AF july/05
1297.         with open(imageTxtName,"wb") as txtfile:
1298.             txtfile.write( title_fa.encode('utf8')+"\n".encode('utf8'))
1299.             lines=retrieved_texts.split('.')
1300.             for index, line in enumerate(lines):
1301.                 #print(line)
1302.                 txtfile.write(line.encode('utf8')+"\n".encode('utf8'))
1303.
1304.         record = [row - 1, docx_name_only, pic["uid"], title_fa, title_en, "",
1305.                  img_size, height, width, datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
1306.                  datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"), retrieved_texts]
1307.
1308.         for j in range(len(header)):
1309.             ws.cell(row=row, column=j+1, value=record[j])
1310.
1311.     else:
1312.         for pic in pics_info:
1313.             row += 1
1314.             desc = pic['desc']

```

```

1315.     lang = pic['lang']
1316.     title_fa = ''
1317.     title_en = ''
1318.     if(type(desc) is str):
1319.         if lang == 'Persian':
1320.             title_fa = desc
1321.             title_en = ''
1322.         else:
1323.             title_fa = ''
1324.             title_en = desc
1325.     elif(type(desc) is dict):
1326.         if lang == 'Persian':
1327.             title_fa = desc2vec(desc)
1328.             title_en = ''
1329.         else:
1330.             title_fa = ''
1331.             title_en = desc2vec(desc)
1332.
1333.     title_fa = title_fa.strip()
1334.     title_en = title_en.strip()
1335.     # pic_number = extract_pic_number(title_fa)
1336.     retrieved_texts = extract_related_pic_text(docx_filename, title_fa, extra_paragraph)
1337.     rt = retrieved_texts.split(' | ')
1338.     picId=rt[-1] #added by AF july/05
1339.     # *** retrieve only the first mention (after "List of Figures") of the figure title in document
1340.     if(not retrieved_texts == u' '):
1341.         retrieved_texts = retrieved_texts.split(' | ')
1342.         if(len(retrieved_texts)>1):
1343.             retrieved_texts = retrieved_texts[1]
1344.         else:
1345.             retrieved_texts = retrieved_texts[0]
1346.
1347.     pic_name = pic['path'][6:]
1348.     img_filename = img_dir_temp + '/' + pic_name
1349.
1350.     # image file size in KB
1351.     img_size = round(os.path.getsize(img_filename)/1024,1)
1352.
1353.     # create img_dir
1354.     if not os.path.exists(img_dir):
1355.         try:
1356.             os.makedirs(img_dir)
1357.         except OSError:
1358.             print("Unable to create image directory {}".format(img_dir))
1359.
1360.     # image change type to jpeg and create thumbnails
1361.     # image dimensions in pixel
1362.     with Image.open(img_filename) as image:
1363.         width, height = image.size
1364.         #pic_name = pic["uid"] + '.jpg'
1365.         pic_name = picId + '.jpg'
1366.         img_filename = img_dir + '/' + pic_name
1367.         image.convert('RGB').save(os.path.join(img_dir, pic_name), 'JPEG')
1368.         image.thumbnail(thumbnail_size)
1369.         #image.convert('RGB').save(os.path.join(img_dir, "thumbnail" + pic_name), 'JPEG')
1370.
1371.
1372.     related_txt=picId+'.txt' #added by AF july/05
1373.     imageTxtName=img_dir + '/' + related_txt #added by AF july/05
1374.     with open(imageTxtName,"wb") as txtfile:
1375.         #txtfile.write( title_fa.encode('utf8')+"\n".encode('utf8'))
1376.         txtfile.write( title_fa.encode('utf8')+"\n".encode('utf8'))
1377.         txtfile.write( picId .encode('utf8')+"\n".encode('utf8'))

```

```

1378.         lines=retrieved_texts.split('.')
1379.         for index, line in enumerate(lines):
1380.             #print(line)
1381.             txtfile.write(line.encode('utf8')+"\n".encode('utf8'))
1382.
1383.         record = [row - 1, docx_name_only, pic["uid"], title_fa, title_en, "",
1384.                  img_size, height, width, datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
1385.                  datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"), retrieved_texts]
1386.
1387.         for j in range(len(header)):
1388.             ws.cell(row=row, column=j+1, value=record[j])
1389.
1390.         counter += 1
1391.
1392.     except Exception as exp:
1393.         print('something is wrong about file {}. Following exception occurred during image extraction:
1394.               \n{}'.format(file, exp))
1395.
1396.     # removing temporary folder
1397.     if os.path.exists(path_dst_temp):
1398.         shutil.rmtree(path_dst_temp) # remove dir and all contains
1399.
1400.     # saving information in excel file
1401.     try:
1402.         ('{} files were processed.'.format(counter))
1403.         print('{} files did not have caption.'.format(withoutC)) #added by AF
1404.         outputpath=path_src+'/'+ 'ImageDB.xlsx' ##AF
1405.         wb.save( outputpath)
1406.         wb = load_workbook(filename = outputpath)
1407.         sh = wb.active
1408.         cvs_path=path_src+'/'+ 'ImageDB.csv'
1409.         with open(cvs_path, 'w', newline="", encoding='utf-8') as csv_file:
1410.             c = csv.writer(csv_file)
1411.             for r in sh.rows:
1412.                 c.writerow([cell.value for cell in r])
1413.
1414.     except Exception as exp:
1415.         print('something is wrong. Following exception occurred during saving Excel file: \n{}'.format(exp))
1416.
1417. # In[]:
1418.
1419.
1420.
1421. # information about files and paths
1422. #path_src = 'C://Users//lenovo//Documents//proposal99//codes//word_fani' ##AF
1423. path_src='C://Users//lenovo//Documents//proposal99//codes//test'
1424. path_src='C://Users//lenovo//Documents//proposal99//irandocdata//all data//fani//fani_new'
1425. #path_src = 'words'
1426. path_dst = path_src+'/'+ 'images' ##AF
1427. #path_dst='C://Users//lenovo//Documents//proposal99//irandocdata//برقی/images1para"
1428. db_title='ImageDB'
1429.
1430. # thumbnail size
1431. thumbnail_size = (128,128)
1432.
1433. # save policy
1434. save_policy = 'normal' # 'multiple' or 'normal'
1435. file_per_folder = 100
1436.
1437. # number of extracted paragraphs before and after the picture number in the document
1438. extra_paragraph = 3
1439. create_image_db(path_src, path_dst, db_title, thumbnail_size, save_policy, file_per_folder, extra_paragraph)

```


Abstract

In this research, a new method for annotating images in Persian scientific documents is suggested. Images in scientific documents contain valuable information. In many cases, by analyzing images one can understand the main idea and important results of the document. Due to the explosive growth of image data, many search engines are looking for methods to retrieve images from their databases, so that users can have access to the figures related to their query.

Ganj is Iran's scientific information database and one of the main resources of scientific Persian documents; it contains a large number of dissertations, theses, and research proposals. In Ganj, there is the possibility of a search based on a text inquiry and the results are represented in text metadata such as subject, summary, authors, and printed year. However, figures in Ganj are not retrieved. Providing such possibility in Ganj is a value-added that distinguishes it from similar search engines. To this end, it is needed to extract tags for figures in the Ganj database; the search engine can use these tags to retrieve figures containing the searched words in their tags list.

Here, for image annotation, the associated text is used. The caption and the part of the document that includes the reference to the image are considered. Noun phrases in the associated text are ranked based on five different methods; term frequency, inverse document frequency, term frequency–inverse document frequency, cosine similarity between word embedding of noun phrases in the text and the caption, and using both term frequency–inverse document frequency and cosine similarity methods. Image tags in every method are the noun phrases with the highest rank. Proposed methods are evaluated using a test data from Ganj. Based on the results in this research term frequency–inverse document frequency method with F_1 measure of 38% is the best method and after that is term frequency–inverse document frequency and cosine similarity method with F_1 measure of 36%. Since in some cases, the caption of the image does not represent a complete description of the image, considering the similarity with the caption results in a lower F_1 measure.

Keywords: Image tagging, text analysis, keyword extraction, Natural language processing.



**Iranian Research Institute for Information Science and Technology
(Irandoct)**

Research Project

**Developing a method for tagging figures
extracted from scientific documents in Iran
scientific information database (Ganj)**

By:

Azadeh Fakhrazadeh

Co-authors:

AmirHosein Seddighi

July 2020

