کار با کتابخانه هضم

گروه ۵

شیوا رادمنش، سحر شیخ‌الاسلامی، سیدکسری کرمی‌نژاد، علی نصیری سروی، حامد واعظی

اطلاعات گزارش

تاریخ: ۱۳۹۹/۷/۱۹

واژگان کلیدی:
توکن سازی
نرمال سازی
ریشه یابی کلمات

چکیده

در این تکلیف به نحوه کار با کتابخانه هضم(hazm) در زبان برنامه نویسی Python پرداخته شد و با استفاده از آن تعدادی سطر مشخص شده را در ابتدا با استفاده از کلاس Normalizer که مربوط به نرمال سازی متون می‌شود، نرمال سازی کردیم و سپس متن نرمال شده را با استفاده از کلاس های توکن سازی، توکن های آن را جدا نموده و متن را پردازش کردیم. در نهایت برای ۵ جمله تصادفی از متن، کلمات موجود در آن را ریشه یابی کردیم.

۱- مقدمه

بازیابی اطلاعات (IR) پیدا کردن مواد (معمولا اسناد و داکیومنت ها) یک گونه غیرساختیافته (معمولا متن) که یک نیاز اطلاعاتی را از میان مجموعه های بزرگی از داده (که معمولا روی کامپیوترها ذخیره می‌شوند) تامین می کند. عموما این داده‌های غیرساختیافته به فرمت‌های مختلفی ذخیره شده‌اند و این می‌تواند باعث ایجاد مشکل در هنگام بازیابی شود. از این رو لازم است پیش پردازش‌هایی انجام شود تا اسناد آماده‌ی استفاده گردند. از انواع این پیش پردازش‌ها می‌توان به نرمال سازی، توکن سازی و ریشه یابی کلمات اشاره کرد.

این تکلیف به چهار بخش تقسیم می‌شود:

در قسمت اول یک فایل با فرمت docx که برای پردازش مناسب نیست را به فرمت txt در می‌آوریم تا برای پردازش‌های مراحل بعد مشکلی نداشته باشیم.

در قسمت دوم ابتدا متن مربوط به گروه خود را جدا می‌کنیم. سپس پیش پردازش‌های لازم برای متن را اعمال کرده و متن را نرمال سازی می‌کنیم.

در قسمت سوم ابتدا متن مربوط به گروه خود را به کلمات مجزا شکسته و در نهایت نوع هر کلمه (اسم،فعل و...) را استخراج می‌کنیم.

در قسمت چهارم ۵ جمله تصادفی از متن مربوط به گروه خود را انتخاب کرده و ریشه کلمات موجود در آن جملات را استخراج می‌کنیم.

۲- شرح تکنیکال

شرح تکنیکال +کد های گزارش

قسمت اول:

با استفاده از کتابخانه‌ی "docx2txt" فایل paragrapgs.docx را به فایل paragraphs.txt تبدیل کردیم.

قسمت دوم:

در فایلی که در اختیار ما قرار گرفته شده بود، در ابتدای متن هر گروه کلید واژه‌ی «گروه» به همراه شماره‌ی گروه نوشته شده بود. برای جدا کردن متن گروه خود(گروه ۵)، متن داده شده را با استفاده از کلید واژه‌ی «گروه» جداسازی می‌کنیم.

ماژول "Normalizer" در کتابخانه‌ی هضم، پیش پردازش‌هایی را در متن اعمال می‌کند. این پیش پردازش‌ها شامل موارد زیر می‌باشد:

- تبدیل فاصله به نیم فاصله (به وسیله‌ی تابع "affix_spacing")
- حذف کاف و یای عربی(به وسیله‌ی تابع "character_refinement")
- ایجاد فاصله‌ی مناسب بعد (یا قبل) علائم نگارشی(به وسیله‌ی تابع "punctuation_spacing")

این ماژول شامل تابع "normalize" نیز می‌باشد که همه‌ی پیش پردازش‌های ذکر شده را انجام می‌دهد. در این پروژه برای اعمال پیش پردازش‌های لازم از این تابع استفاده شده است.

قسمت سوم:

ابتدای هر پاراگراف، در فایلی که در اختیار ما قرار داده شده بود، کلید واژه‌ی «پاراگراف» به همراه شماره آن پاراگراف داده شده بود. به همین دلیل برای جداسازی هر پاراگراف، ابتدا کل متن گروه خود (گروه ۵) را

بر اساس واژه پاراگراف جداسازی می‌کنیم.

ماژول "SentenceTokenizer" در این کتابخانه، جملات را شناسایی و جدا می‌کند. با استفاده از این این ماژول، جمله های هر پاراگراف را استخراج می‌کنیم. با توجه به اینکه بعد از هر پاراگراف در متن، شماره آن پاراگراف آمده است،اولین جمله هر پاراگراف در واقع شماره همان پاراگراف است. به همین دلیل برای بررسی کلمات هر جمله از دومین جمله آن شروع می‌کنیم. خروجی این ماژول، لیستی از جمله های هر پاراگراف است و با توجه به توضیحات بالا، یکی کمتر از اندازه ی این لیست، تعداد جملات هر پاراگراف را نشان می‌دهد.

ماژول "WordTokenizer" در کتابخانه ی هضم، کلمات هر جمله را شناسایی می‌کند. ورودی این ماژول، جملات می‌باشد و در خروجی کلمه های آن جمله در لیست آمده است. با استفاده از این ماژول، کلمات متن را استخراج کردیم. همانند بالا، سایز این لیست تعداد کلمات را نشان می‌هد.

حال با استفاده از ماژول "POSTagger" نوع هر کلمه را مشخص می‌کنیم. این ماژول یک pattern یا مدل به عنوان ورودی می گیرد. ما از مدل "postagger.model" که مدل ارائه شده توسط خود کتابخانه است استفاده کردیم. ورودی این ماژول لیستی از کلمات است و در خروجی، کلمه و نوع آن کلمه(فعل، اسم، ....) را برمی‌گرداند.

پس از مشخص کردن نوع هر کلمه، با شمردن کلماتی که tag آنها برابر "V" می‌باشد، تعداد افعال پاراگراف را بدست می‌اوریم. با انجام همین عمل برای اسم(آنهایی که tag شان برابر"NE" یا "N" باشد)، تعداد اسم ها را بدست می آوریم.

لیست های "verbs_list" و "nouns_list" و "token_list"، به ترتیب شامل کل افعال، اسم ها و کلمات متن هستند. همچنین محتوای این لیست ها در فایل ذخیره شده است.

خروجی نهایی کد نیز، جدولی است که به ازای هر پاراگراف، تعداد جملات، تعداد کلمات، تعداد افعال و اسم را گزارش کرده است.

این جدول ۱ آورده شده است.

Par. No	Sent #	word #	verbs#	nouns#
1	2	77	8	11
2	4	122	9	19
3	3	116	15	22

جدول ۱ . تعداد جملات، کلمات، افعال و اسم های هر پاراگراف

قسمت چهارم:

در مرحله اول تمامی جملات سه پاراگراف در کنار هم قرار داده تا لیستی از ۹ جمله داشته باشیم. سپس ۵ جمله را به صورت رندوم از بین آن‌ها انتخاب می‌کنیم.

با استفاده از ماژول "WordTokenizer" که توضیح آن در مرحله قبل داده شده است، توکن‌های مربوط به هر جمله را استخراج می‌کنیم.

سپس به کمک ماژول "Lemmatizer" ریشه هر کلمه را استخراج می‌کنیم. این ماژول یک کلمه و نوع آن کلمه را گرفته و به کمک آن ریشه کلمه را پیدا می‌کند (نوع کلمه به عنوان پارامتر اختیاری می‌باشد). سپس هر کلمه و ریشه مرتبط را در یک دیکشنری ذخیره می‌کنیم.

برخی از کلمات و افعالی که توسط این ماژول به درستی ریشه یابی شده اند را با هم بررسی می‌کنیم:

کلمه اصلی	ریشه کلمه	نوع کلمه
نخواهند شد	شد#شو	فعل
بازاری	بازار	اسم
کنید	کرد#کن	فعل
دارد	داشت#دار	فعل
نیست	#هست	فعل
بازیهای	بازی	اسم

جدول ۲ .برخی از کلمات و ریشه مرتبط آن‌ها