# گزارش کار تکلیف شماره 2

در این تکلیف، با استفاده از کتابخانه hazm، سعی در انجام پردازش‌هایی ساده بر روی متون هستیم.

* **گروه 3:**
  + دينا اميدوار طهراني
  + پرستو باقرپور
  + نسيم فاني
  + محسن محمودزاده
  + مريم واقعي

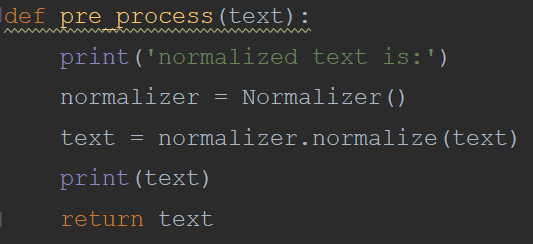
## خواندن ورودی

در ابتدای امر، با استفاده از کتابخانه docx2txt، محتویات فایل ورودی با پسوند .docx را در متغیری به نام MY\_TEXT ذخیره می‌کنیم.

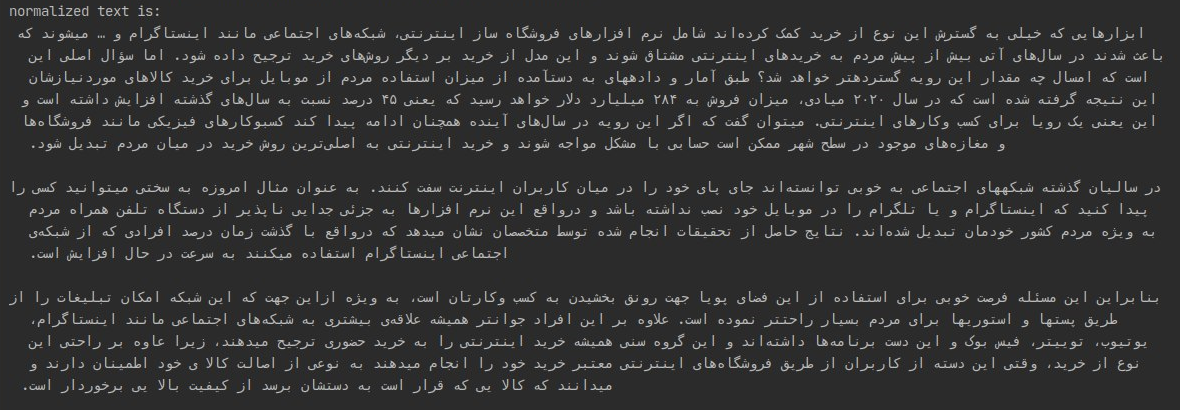


## پيش‌پردازش ورودي

در ادامه، با فراخواني تابع pre\_process، پيش‌پردازش‌هاي لازم بر روي متغير MY\_TEXT را انجام مي‌دهيم. انجام اين كار شامل نمونه‌سازي از ماژول Normalizer كتابخانه‌ي hazm و فراخواني متد normalize اين نمونه است. اين پيش‌پردازش‌ها شامل پردازش‌هايي از جمله تبديل فاصله‌ها به نيم‌فاصله‌ها، حذف كشيدگي‌هاي كلمات و ... است. مقدار بازگشتي متد normalize، متن نرماليزه شده‌ي موردنظر است.



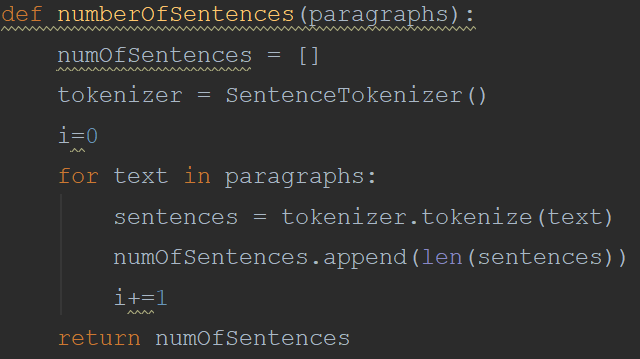




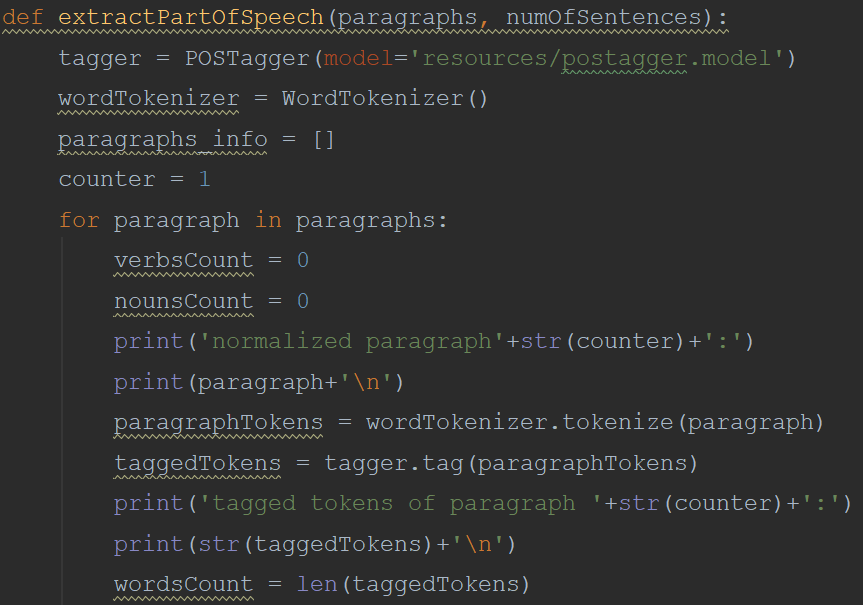
## پردازش پاراگراف‌ها

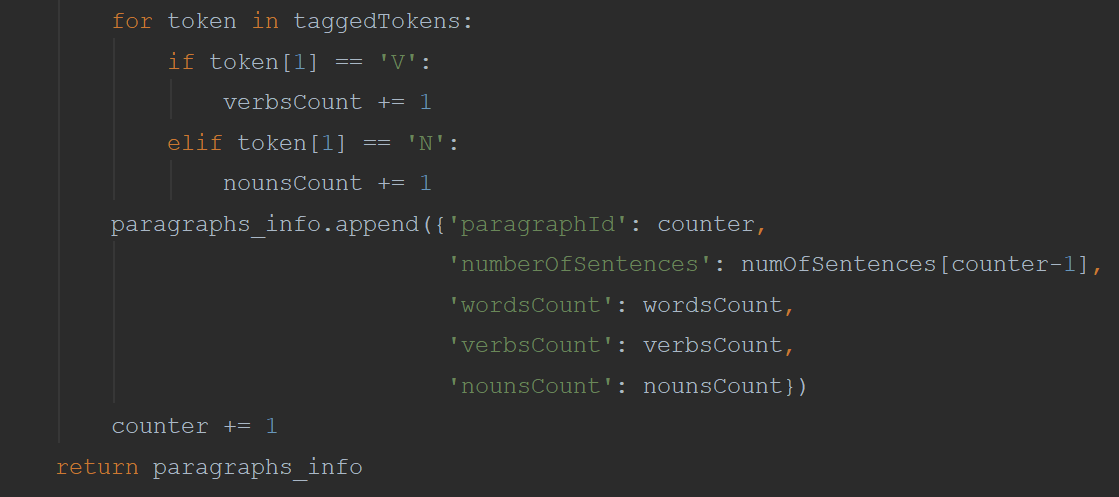
در اين گام، با استفاده از توابع numberOfSentences و extractPartOfSpeech اطلاعات موردنياز يعني تعداد جملات، تعداد كلمات، تعداد فعل‌ها و تعداد اسم‌ها را براي هر پاراگراف به دست‌ مي‌آوريم. پس در ابتدا، پاراگراف‌ها را از هم جدا مي‌كنيم.

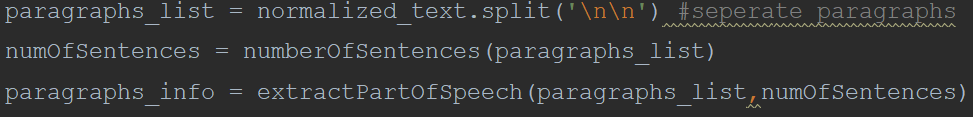
در تابع numberOfSentences، تمامي پاراگراف‌ها را به عنوان ورودي گرفته و سپس با استفاده از ماژول SentenceTokenizer كتابخانه‌ي hazm، يك نمونه از اين ماژول ساخته و با فراخواني متد tokenize آن بر روي هر پاراگراف، آرايه‌اي از توكن‌ها را به دست مي‌آوريم. هر يك از اين توكن‌ها، يك جمله از پاراگراف هستند. براي پردازش‌هاي پيش رو، تعداد جملات هر پاراگراف را نيز در يك ليست به نام numOfSentences ذخيره مي‌نماييم.

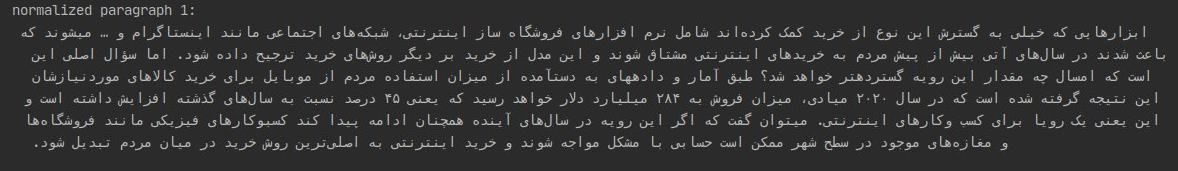


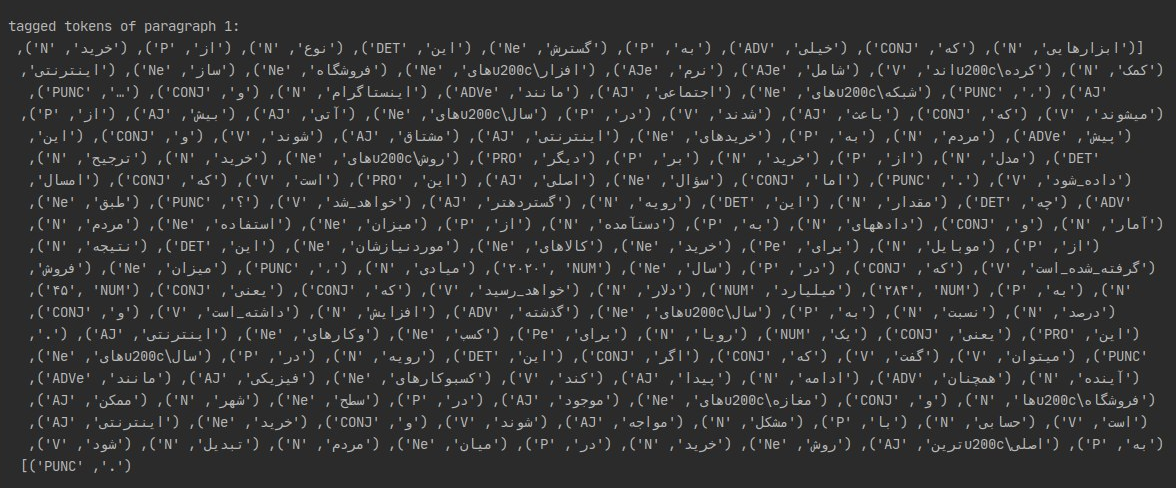
در تابع extractPartOfSpeech، از ماژول‌هاي POSTagger و WordTokenizer كتابخانه hazm استفاده مي‌كنيم. پس از نمونه‌سازي از اين ماژول‌ها و با پيمايش هر پاراگراف، با استفاده از متد tokenize نمونه‌ي ساخته شده از WordTokenizer، تعداد كلمات را در آن پاراگراف به دست مي‌آوريم. حال با استفاده از متد tag نمونه‌ي ساخته شده از POSTagger، توكن‌هاي به دست آمده از مرحله‌ي قبل را تگ مي‌زنيم. حال با پيمايش بر روي اين توكن‌هاي تگ زده شده، اگر توكن فعل بود(تگ V)، تعداد فعل‌ها را يك واحد افزايش مي‌دهيم و اگر توكن اسم بود(تگ N)، به تعداد اسم‌ها يك واحد مي‌افزاييم. در نهايت اطلاعات به دست آمده را در يك ديكشنري ذخيره كرده و به ليست paragraphs\_info كه دربردارنده‌ي اطلاعات موردنظر هر پاراگراف است، اضافه مي‌نماييم.

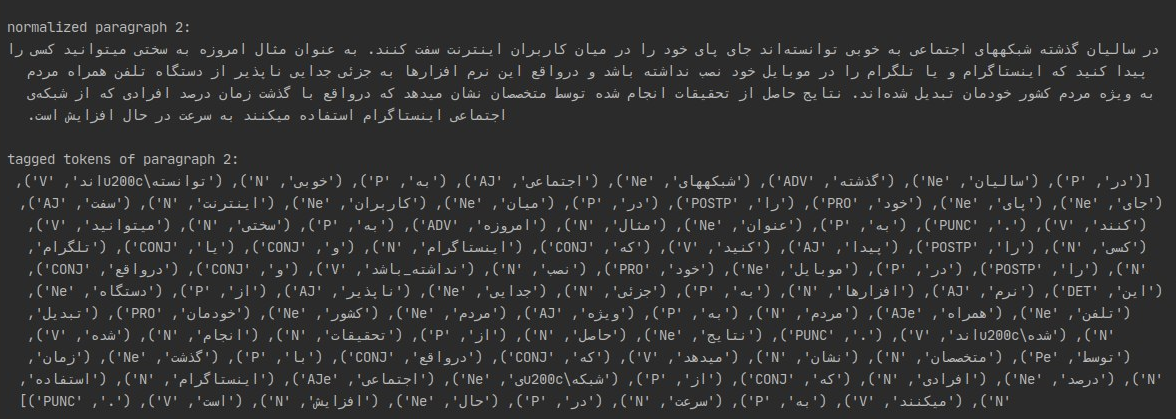


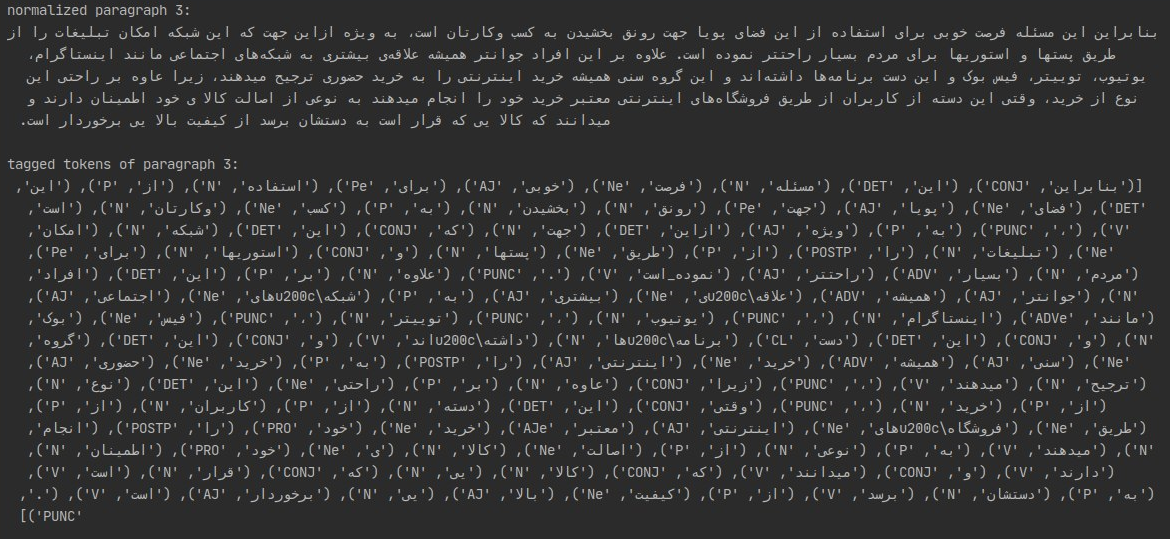
**

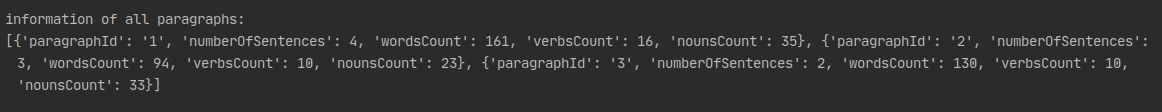




**

**

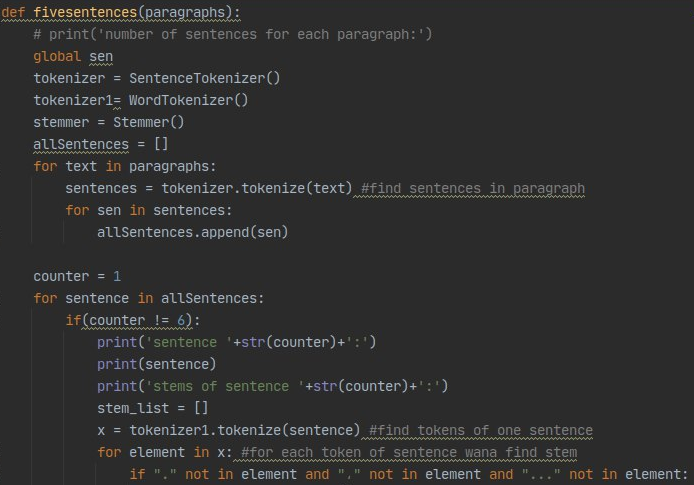
**

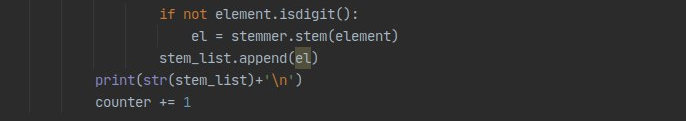
**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | تعداد جملات | تعداد كل كلمات | تعداد فعل‌ها | تعداد اسم‌ها |
| پاراگراف 1 | 4 | 161 | 16 | 35 |
| پاراگراف 2 | 3 | 94 | 10 | 23 |
| پاراگراف 3 | 2 | 130 | 10 | 33 |

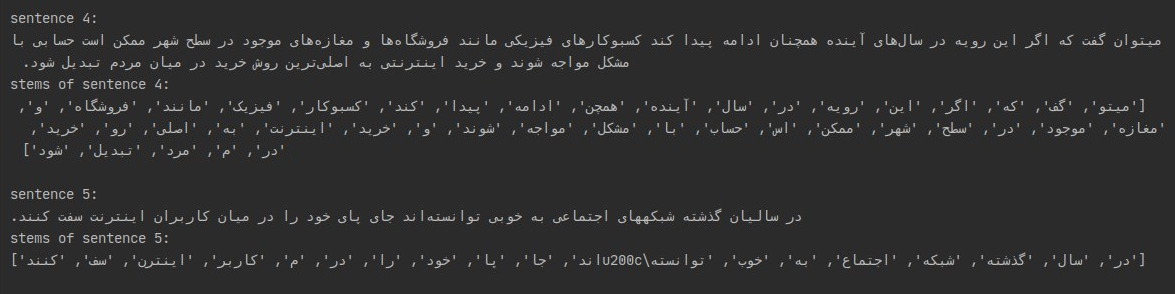
## ريشه‌يابي

در اين گام، به كمك به متغيري به نام counter، 5 جمله اول متن را انتخاب مي‌كنيم. در ليست allSentences نيز تمامي جملات متن را به صورت تفكيك شده ذخيره مي‌نماييم. سپس بعد از چاپ شماره و محتواي جمله، به كمك نمونه ساخته شده از ماژول WordTokenizer، توكن‌هاي هر جمله را به دست آورده و براي هر توكن، در صورتي كه كلمه يا فعل باشد، به كمك نمونه ساخته شده از ماژول Stemmer كتابخانه‌ي hazm، ريشه‌ي آن توكن را به دست مي‌آوريم و حاصل را در stem\_list ذخيره مي‌كنيم.









از مواردي كه ريشه‌يابي آنها به درستي در مرحله قبل ظاهر شده‌اند، مي‌توان به موارد زير اشاره كرد:

1. توانسته اند ---> توانست#توان
2. نداشته باشند ---> داشت#دار
3. برنامه ها ---> برنامه
4. متخصصان ---> متخصص