

**فاز اول پروژه بازیابی اطلاعات**

**نام دانشجو: سپهر عسگریان ابیانه**

**شماره دانشجو: 9531901**

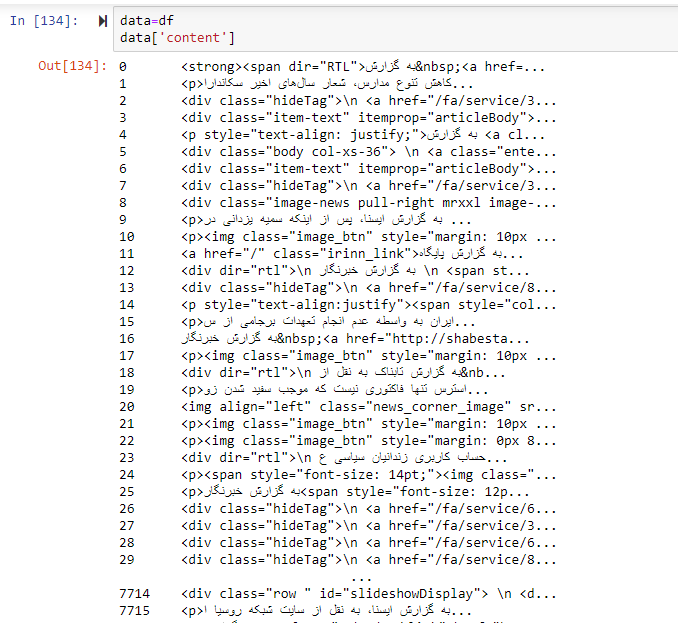
**اردیبهشت 1399**

**گزارش پروژه**

**ابتدا برای هر دو حالت تمامی دیتا ها با یک دیگر مرج شده اند و تشکیل دیتایی با 55000 داده را داده اند.**

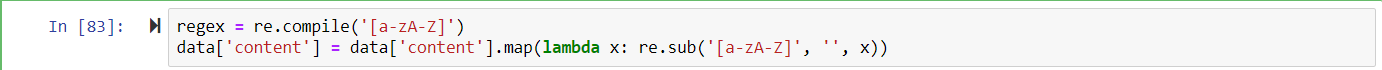
**ابتدا حالت اول به اسم فایل حالت اول که در jupyter پیاده سازی شده است و همچنین نتایج آن نیز در آن فایل قابل مشاهد است .**

**دیتای اولیه به صورت زیر قابل مشاهده است**

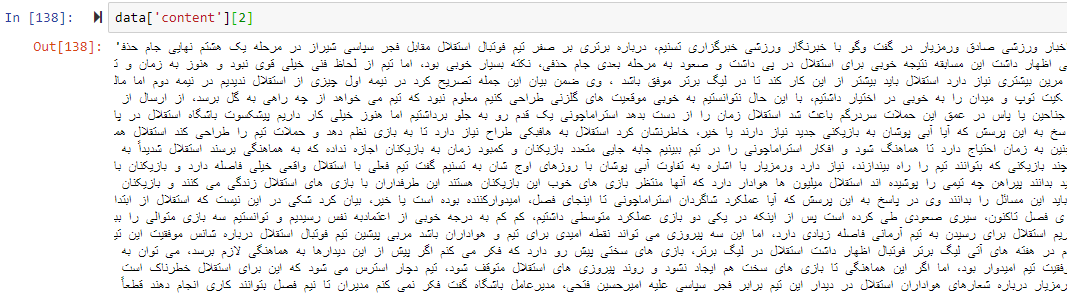
****

1. **حالت اول**

**در حالت اول نرمالایز خیلی ساده همراه با حذف stop words داشتیم به همره توکنایز کردن بسیار ساده که مثلا برای واژه کتاب ها توکنایز به صورت کتاب و ها تشکیل میشد که در اینجا توکنایز اضافی داریم . ابتدا در این بخش با استفاده از دستور**

****

**کلیه واژگان انگلیسی را حذف کرده همچنین با استفاده از دستورات زیر علائم نگارشی ، تگ های html اعداد و اموجی ها همگی حذف گردیده اند . در اینجا از دیتور re.sub استفاد شده است و از کتابخانه regex بهره برده شده است . برای حذف علائم نگارشی همانطور که در کد نیز قابل مشاهده است با یک فاصله در متن عوض گردیده است**

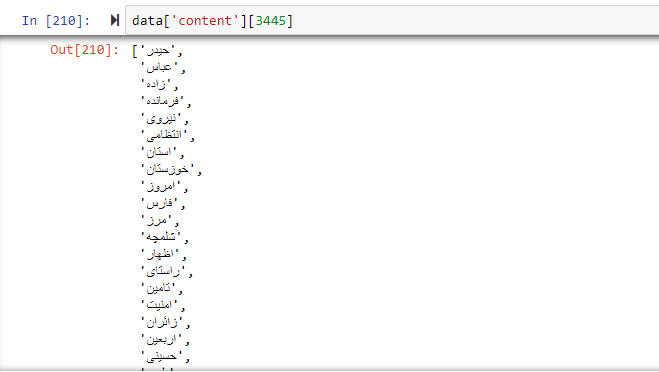
**حال داده را بعد از اینکه یک نرمالایز ساده شد در زیر نمایش داده شده است برای مثال خبر اول آمده است که فاقد هرگونه کلمه انگلیسی یا تگ میباشد. لازم به ذکر است نرمالایز کامل در مرحله دوم دقیقا طبق صورت پروژه انجام شده است**

**حال در این مرحله تمامی stop words ها از یک فایل word که در اینترنت نیز موجود بود برداشته شده است و خوانده شده**

**سپس همانطور که در صورت پروژه آمده است toekenize بر اساس فاصله ای که موجود است در متن انتخاب شده است و همچنین stopwords ها از متن بیرون کشیده شده اند(فایل stopwords استفاده شده در داکیومنت گزاشته شده است) هماند شکل زیر قابل مشاهده است.**



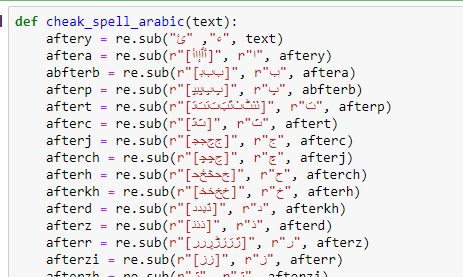
حال برای مثال بعد از مراحل بالا به شکل زیر میشود خروجی



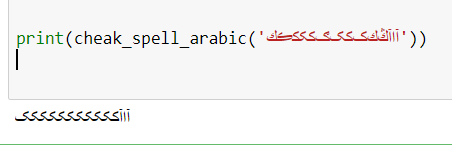
2-حالت دوم :

**در حالت دوم نرمالایز کردن با دقت بیشتری انجام میشود همانطور که در صورت پروژه نیز گفته شده است پس از مرج کردن دیتا ابتدا حالت های مختلف کلمه به یک حالت واحد تبدیل شده است**

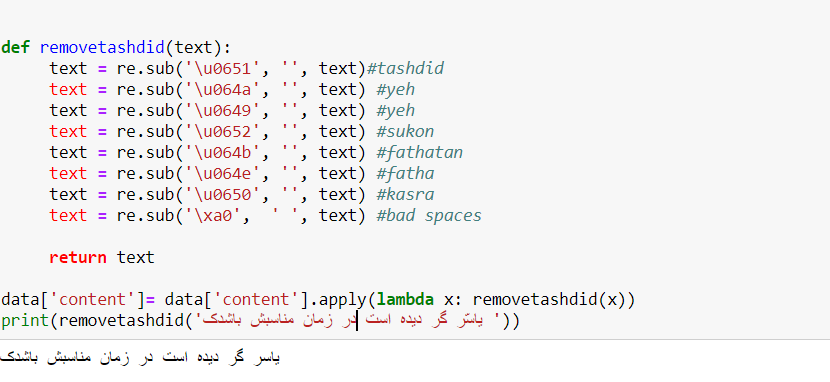
**کد آن مانند شکل زیر میباشد**



**برای مثال برای حلت زیر خروجی به صورت زیر میباشد**

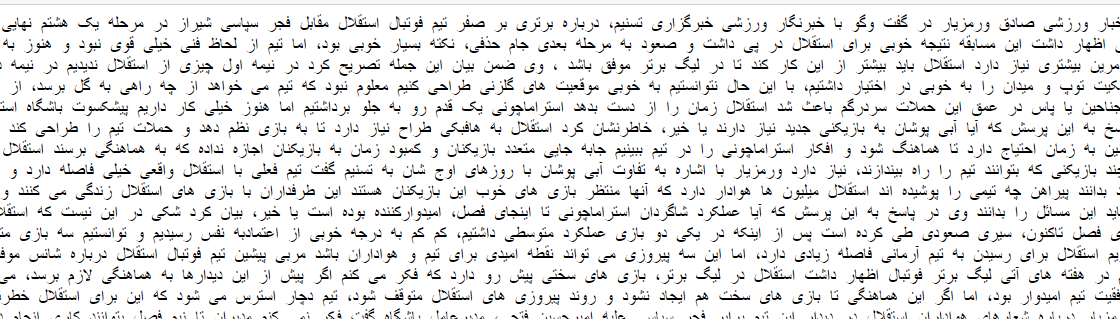


**حال علاوه بر آن انواع علائم تشدید تنوین کسره و غیره با کد زیر و مثال زیر حذف شده است**

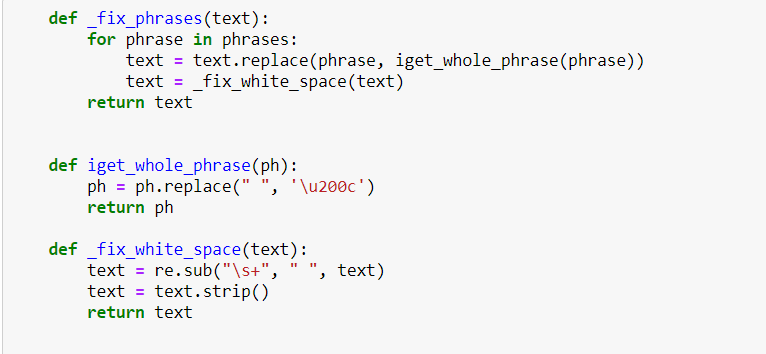
**همانطور که دیده میشود تشدید در زیر حذف گردیده است**

**بقیه ی نرمالایز همانند حالت اول انجام شده است مانند حذف اعداد ، حذف اموجی، حذف علائم نگارشی و غیره که در شکل حالت اول نیز موجود میباشد**

**بعد از نرمالایز خروجی ها به صورت زیر میباشند**



**حال در این قسمت علاوه بر خواندن فایل stopwords فایلی به نام عبارت ها نیز ذخیره شده است که از گیت‌هاب فایل ورودی آن را پیدا کرده ام. این فایل شامل عباراتی مثل فیه‌مافیه مع‌ذلک علی ای حال و غیره میباشد که همه در توکن کردن به یک شکل میباشند پس بنابراین برای اینکار این عبارات خوانده شده است و همانند شکل زیر مشکل آن هندل شده است**



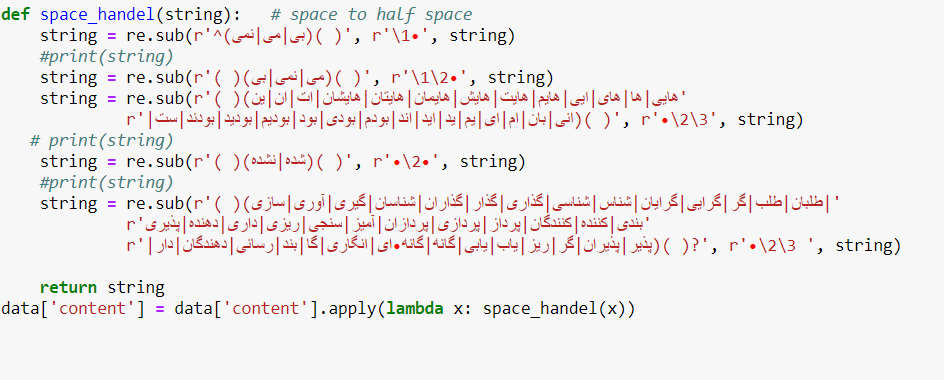
**که در اینجا برای هرکدام از آن ها بینشان اگر فاصله ا موجود میباشد پاک شده است و برای مثال علی ای حال به صورت جدا آمده که همانطور که می‌بینید به صورت سر هم نوشته شده است**

همچنین این عبارت



**به معنای این استکه هر text آن را میفرستد در تابع و خروجی می‌گیرد**

**سپس برای اینکه علائم جمع توکن اضافی همانند حالت اول به حساب نگیرد کد زیر نوشته شده است که علاوه بر علائم جمع پسوند ها و پیوند های مهم دیگر نیز دقت شده که در صورت مشاهده با نیم فاصله بیایند**

****

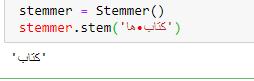
**برا مثال در شکل زیر این تابع برای رشته مشخص شده پیوشند ها و پسوند ها را درست کرده است**

**حال مانند قبل tokenize و stopwords ها هندل شده اند لازم به ذکر است که در TOKENIZE علائم پیشوندی و پسوندی به صورت کامل درست جدا شده اند و مزیت آن این است که در مرحله ریشه یابی با استفاده از کتابخانه hazm به خوبی ریشه یابی و lemmatize انجام میشود**

**سپس با استفاده از کتابخوانه hazm ریشه یابی و lemmatize انجام دشه است همانطور که در شکل زیر میبینید**

**موارد خواسته شده بدست آمده شده است**

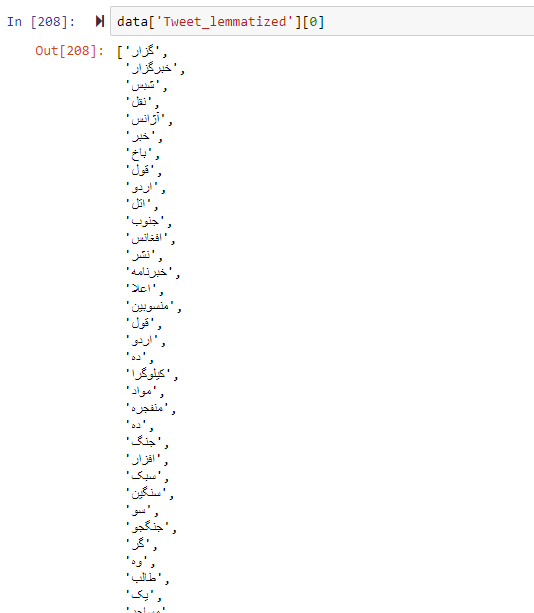
****

****

این عملیات مانند عملیات ریشه یابی برای انگلیسی است با این تفائت که library آن hazm میباشد

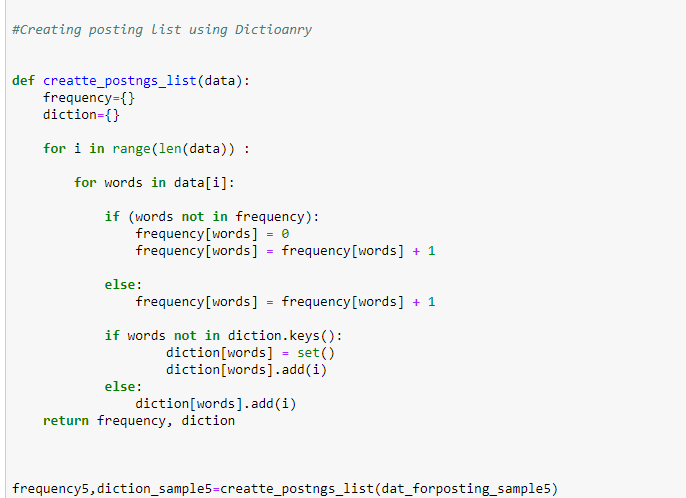


نمونه خروجی :

****

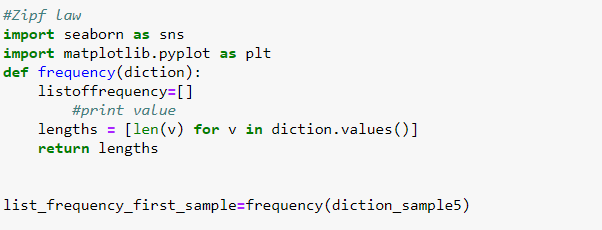
بخش سوم:

در این بخش به پیاده سازی heap و zeap میپردازیم اما در ابتدا تابعی نوشته شده تحت عنوان create\_posting\_list



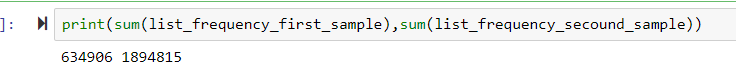
که طبق آنچه در صورت پروژه آمده بود به صورت dic ذخیره شده است تا سرعت بالا باشد در این جا پستینگ لیست مورد نظر ساخته میشود به این صورت که اگر ورد در آن نباشد آن ترم اضافه گشته و اگر موجود باشد شماره ترم آن ذخیره گشته است .

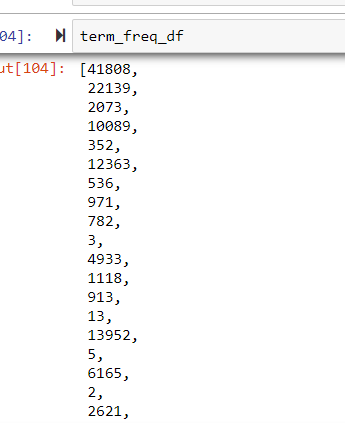
برای محاسبه freq کد زیر مورد استفاده قزاز گرفت



که در آن برای هر کلمه تعداد تکرار آن محاسبه شده است

برای مثال برای لیست 15000 تایی و 5000 تایی در کل به تعداد زیر توکن موجود میباشد





برای کل دیتا ست

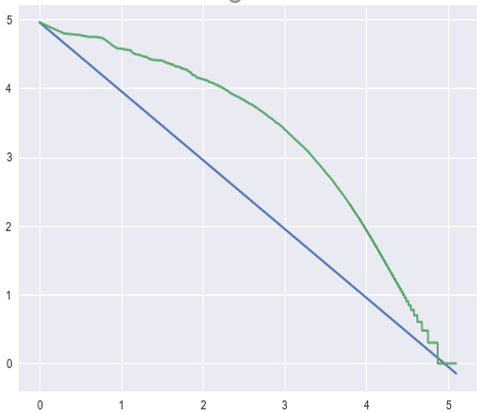
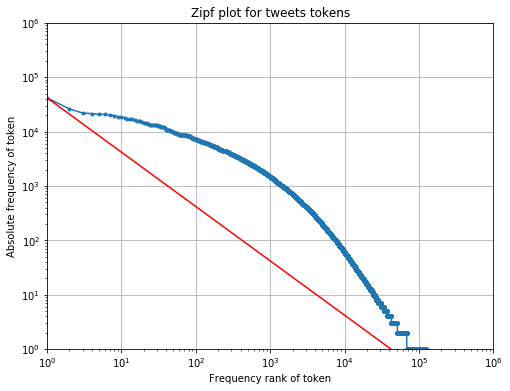
در کل برای هر دو قانون حالت دوم بهتر از حالت اول است بخاطر اینکه بهتر نرملایز شده و بهتر tokenize انجام شده و همچنین ریشه یابی شده است این باعث میشود که تعداد توکن های منحصر به فرد کمتر شود که بسیار برای ما بهتر است و میتوان دید که هر دو قانون درست میباشند

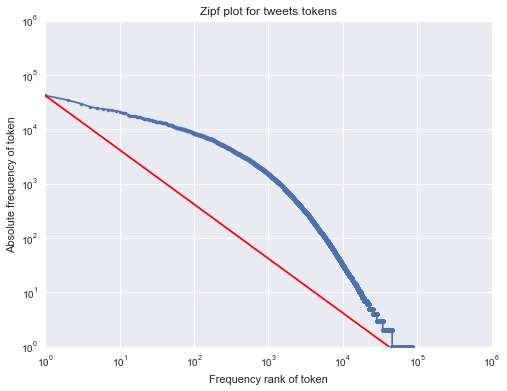
قانون Zipf یک قانون در مورد توزیع فراوانی کلمات در زبان است (یا در مجموعه‌ای که به اندازه کافی بزرگ است به طوری که نماینده زبان باشد). برای نشان دادن قانون Zipf اجازه دهید فرض کنیم که ما یک مجموعه داریم و اجازه می‌دهیم در مجموعه کلمات منحصر به فرد (دایره لغات)باشد.

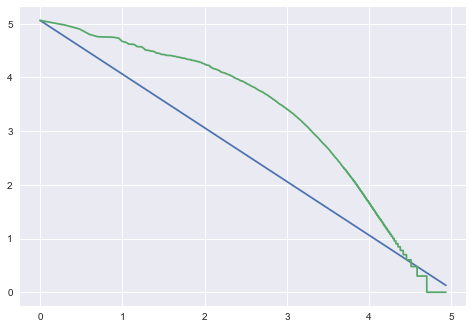
برای پیاده سازی آن همانطور که در کد نیز مشاهده میشود شکل زیر بدست آمد

نمودار ها با seasborn کشیده شده اند

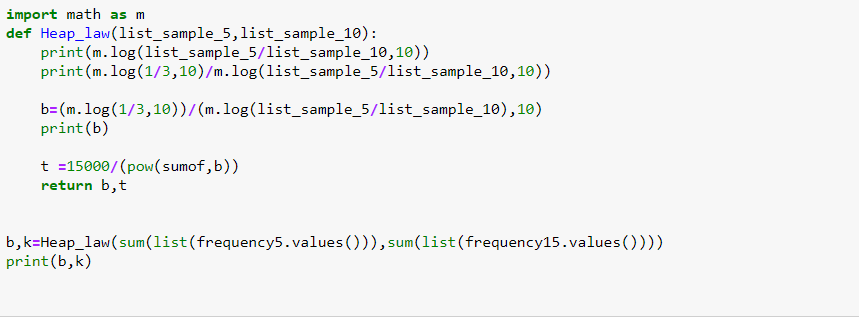
نمودا ر حالت 1



نمودار در حالت دوم



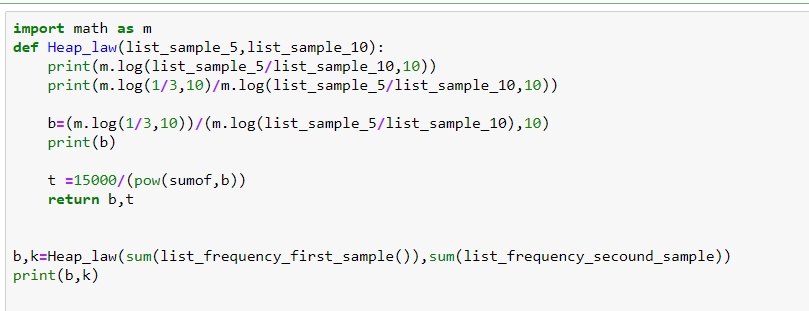
در این بخش نیز heap پیاده سازی شده است



که در اینجا دو زیرمجموعه 5000 تایی و 15000 تایی به صورت رندوم با تابع



ساخته شده که با توجه به عدد رندوم انتخاب میشود که چه داکیومنت هایی برای posting list انتخاب شوند

در اینجا نیز مقدار بهینه b و k با توجه به دو معادله دو مجهول بدست می‌آید