

به نام خداوند بخشنده و مهربان



دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گروه نرم افزار

بازیابی پیشرفته اطلاعات

پروژه پایانی

فاز اول

استاد: دکتر الهام اسماعیلی

ارائه دهندگان:

امیر سرتیپی ۹۹۳۶۱۴۰۱۹

مهدی مالوردی ۹۹۳۶۴۴۰۱۲

فهرست مطالب

۳ فاز اول
۳ drop_coulmns متد
۳ delete_existing_index متد
۳ delete_stop_words متد
۴ csv_reader_index متد
۴ فاز دوم
۴ skipgram روش
۵ delete_stop_words متد
۵ convert_to_vector متد
۵ فاز سوم
۶ convert_to_vector متد
۶ learn_test_model متد

فاز اول

در این پروژه قصد داریم بر روی مجموعه‌ای از کتاب انگلیسی که اطلاعات در یک فایل csv ذخیره شده است را به کمک الستیک ایندکس کنیم. زبان استفاده شده برای کد نویسی زبان پایتون می‌باشد و به کمک کتابخانه‌ای که مربوط به الستیک می‌باشد با API های آن ارتباط برقرار می‌شود. پورتهی که الستیک بر روی آن اجرا می‌شود همان مقدار دیفالت (localhost:9200) می‌باشد. در ادامه به ترتیب روند کاری و متدهای داخل پروژه توضیح داده خواهند شد. پروژه و مستندات آن از طریق این [لینک](#) در بر روی گیت در دسترس می‌باشد.

متد drop_coulmns

با توجه به این که نیازی به ایندکس کردن صفحات HTML کتاب‌ها نیست این متد یک پیش پردازشی از داده‌ها را انجام می‌دهد و ستون ۵ ام دیتا ست که مربوط به متن HTML می‌باشد را از csv حذف کرده و در فایل جدید دیگری با نام books.csv می‌نویسد. پس از این عمل حجم قابل توجهی از فایل ورودی که ۲.۸ گیگابایت بود به ۷۲۰ مگ کاهش پیدا کرد.

ورودی‌های این تابع نام فایل csv که می‌خواهیم ویرایش کنیم و ورودی دوم نام فایل خروجی می‌باشد.

متد delete_existing_index

این متد در صورتی که ایندکسی با نام پارامتری که در ورودی دریافت می‌کند بر روی الستیک وجود داشته باشد، آن را حذف می‌کند.

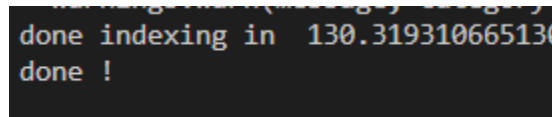
متد delete_stop_words

برای دقت بیشتر موتور جستجویی که می‌خواهیم بسازیم نیاز است تا کلمات توقفی را حذف کنیم. برای این کار از کتابخانه genism استفاده می‌کنیم که در داخل خود دارای لیستی از کلمات توقفی می‌باشد. ابتدا تکست را تماما به حروف کوچک تبدیل می‌کنیم. تابع remove_stopwords این کتابخانه یک متن را دریافت و کلمات توقفی را از آن حذف کرده و باز می‌گرداند. در نهایت اطلاعات پردازش شده در books_final.csv نوشته می‌شود. این عملیات نزدیک به ۳ دقیقه (۱۷۰) ثانیه طول کشید که ۱۶۴۵۱۸ رکورد را پردازش کرد. حجم فایل نهایی به نیز به ۴۰۰ مگ رسید.

متد csv_reader_index

این متد با دریافت فایل ورودی و نام ایندکسی که قرار است ساخته شود در ابتدا یک شی از کلاس Elasticsearch ساخته و به آدرس localhost:9200 متصل می‌شود. سپس فایل csv را خوانده و با کمک تابع bulk به صورت دسته‌ای شروع به ایندکس کردن اطلاعات فایل ورودی می‌کند.

بر اساس شکل ۱ زمان اندازه‌گیری شده که در مشاهده می‌کنید عملیات ایندکس کرد تقریباً ۱ دقیقه و ۲۰ ثانیه به طول می‌انجامد.



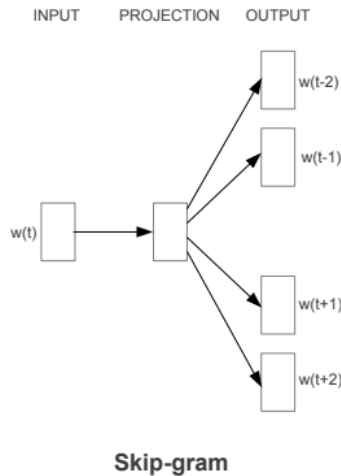
```
done indexing in 130.319310665136
done !
```

شکل ۱. مقدار زمان ایندکس کردن داده‌ها

فاز دوم

روش skipgram

برای یادگیری ماشین از داده‌های خام نیاز است تا این داده‌ها به ماشین داده بشوند. فرمتی که کامپیوتر متوجه به صورت اعداد هست. در این روش کلمات دارای یک فضای برداری هستند و اگر دو تا کلمه دارای بردار نزدیک به یکدیگر باشد یعنی که این دو کلمه به یکدیگر مربوط هستند. استفاده‌ی این الگوریتم یکی از تکنیک‌های یادگیری بدون نظارت است که برای یافتن مرتبط‌ترین کلمات برای یک کلمه مشخص استفاده می‌شود. این الگوریتم بر عکس CBOW عمل می‌کند که کلمه در یافت می‌کند و یک context رو در خروجی می‌دهد اما CBOW یک کانتکست به عنوان ورودی هست و در خروجی یک کلمه را تحویل می‌دهد. در ابتدا کلمات به باید به صورت on hot encode تبدیل شوند. بعد از اون باید کلمه ورودی به شبکه عصبی داده بشود که این میاد و ضرب نقطه‌ای بین کلمه و و هر کلمه از وکتور رو انجام میدهد و میفرسته به خروجی.



متد delete_stop_words

با توجه به این که ممکن است بعد جستجو بر روی این ابسترتکها قرار گیرد در این فاز ابسترتکها نیز پردازش شده‌اند. همچنین یک رجکس نوشته شده است تا فقط حروف انگلیسی در متن باشند و اعداد، علامت‌ها، اسلش‌ها و مواردی غیر از حروف حذف شوند تا متن بهتر پیش پردازش شود. با این کار عملیات پیش پردازش text_body و abstract به ۲۶۹ ثانیه (۴ دقیقه و ۴۶ ثانیه) رسید.

متد convert_to_vector

برای تبدیل متن‌ها به وکتور از کتابخانه fasttext استفاده شد. این کتابخانه براساس دو الگوریتم توضیح داده شده این عملیات را انجام می‌دهد.

در این تابع ابتدا با کوئری زدن بر روی ایندکس کتاب‌ها ابسترتک تمامی کتاب‌ها دریافت می‌شود. چون ورودی مدل train_unsupervised به صورت فایل txt می‌باشد ابتدا هر ابسترتک هر کتاب را داخل یک فایل نوشته و بعد فایل را به عنوان آرگومان به آن پاس می‌دهیم. سپس پس از ترین شدن در یک لیست یک دیشکنری که شامل مدل ترین شده و آیدی آن داکيومنت هست را برای مراحل بعدی برمی‌گردانیم.

فاز سوم

در این فاز تغییراتی در متد convert_to_vectore ایجاد شد تا داده‌هایی که قرار است برای آموزش مدل استفاده شود را بسازیم.

متد convert_to_vector

پس از آموزش مدل به روش unsupervised برای ابسترکت هر کتاب در خروجی می توان وکتور کلمات را دریافت کرد. کتابخانه fast text خود براساس اهمیت و frequency این وکتور را به ما می دهد. پس از دریافت این وکتور و با پیش پردازشی که انجام شده ۱۰ تای اول را بر می داریم (ممکن است بعضی از ابسترکت ها ۱۰ کلمه نداشته باشند). سپس به فرمت زیر یک خط تولید می کنیم و در نهایت رشته ابسترکت را به آن می چسبانیم.

```
__label__x __label__y abstract \n
```

که در آن x و y برچسب های ما هستند. و همه را در آرایه ریخته و برمی گردانیم.

متد learn_test_model

در این قسمت ابتدا داده های تست و ترین را جدا می کنیم و هر ابسترکت به همراه لیبل هایش را در یک خط می نویسم. دو فایل train و test آماده شده را به txt تبدیل کرده و به ورودی الگوریتم train_supervised می دهیم.

پس از آموزش این الگوریتم متد test را از مدل آموزش دیده فراخوانی می کنیم. داده های test را به آن پاس می دهیم. خروجی این الگوریتم در یک tuple به ما تعداد داده های تست، دقت و در نهایت فراخوانی را می دهد.

برای پیش بینی نیز می توان یک متن را predict مدل پاس داده که لیبل های اون را پیش بینی و یک کانفیدنسی برای آن ها تعیین می کند.