به‌نام 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 993614019**

**مهدی مالوردی 993644012**

فهرست مطالب

[فاز اول 3](#_Toc76490320)

[متد drop\_coulmns 3](#_Toc76490321)

[متد delete\_existing\_index 3](#_Toc76490322)

[متد delete\_stop\_words 3](#_Toc76490323)

[متد csv\_reader\_index 4](#_Toc76490324)

[فاز دوم 4](#_Toc76490325)

[روش skipgram 4](#_Toc76490326)

[متد delete\_stop\_words 5](#_Toc76490327)

[متد convert\_to\_vector 5](#_Toc76490328)

[فاز سوم 5](#_Toc76490329)

[متد convert\_to\_vector 6](#_Toc76490330)

[متد learn\_test\_model 6](#_Toc76490331)

# 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 کتاب خانه‌ای که مربوط به الستیک می‌باشد با API های آن ارتباط برقرار می‌شود. پورتی 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](https://github.com/amirsartipi13/advanced-IR) 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 پیش‌پردازشی از داده‌ها را انجام می‌دهد و ستون 5 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 2.8 گیگابایت بود به 720 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 موتورجستجویی 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 پرداز شده در books\_final.csv نوشته می‌شود. این عملیات نزدیک به 3 دقیقه (170) ثانیه طول کشید که 164518 رکورد را پردازش کرد. حجم فایل نهایی به نیز به 400 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 1 زمان اندازه گیری شده که در مشاهده می‌کنید عملیات ایندکس کرد تقریبا 1دقیقه و 20 ثانیه به طول می‌انجامد.

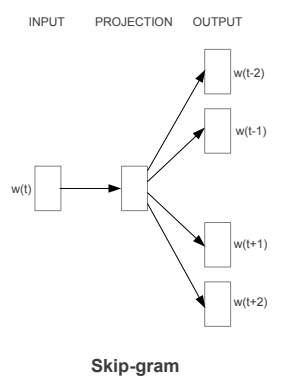


شکل 1. مقدار زمان ایندکس کردن داده‌ها

# 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 دوتا کلمه دارای بردار نزدیک به یک‌دیگر باشد یعنی که این دوکلمه به یک‌دیگر مربوط هستند. استفاده‌ی این الگوریتم یکی از تکنیک های یادگیری بدون نظارت است که برای یافتن مرتبط ترین کلمات برای یک کلمه مشخص استفاده می شود. 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 میده و میفرسته به خروجی.



# متد delete\_stop\_words

با توجه به این که ممکن است بعد جستجو بر روی این ابسترکت‌‌ها قرار گیرد در این فاز ابسترکت‌ها 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 269 ثانیه (4 دقیقه و 46 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 ابسترکت تمامی کتاب‌ها دریافت می‌شود. چون ورودی مدل train\_unsupervised به صورت فایل .txt می‌باشد ابتدا هر ابسترکت 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 داکیومنت 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 می‌توان وکتور کلمات را دریافت کرد. کتابخانه fast text خود براساس اهمیت و frequency این وکتور را به ما می‌دهد. پس از دریافت این وکتور و با پیش پردازشی که انجام شده 10 تای اول را بر می‌داریم (ممکن است بعضی از ابسترکت ها 10 کلمه نداشته باشند). سپس به فرمت زیر یک خط تولید می‌کنیم و در نهایت رشته ابسترکت را به آن می‌چسبانیم.

**\_\_label\_\_x \_\_label\_\_y abstract \n**

که در آن x و y برچسب‌های ما هستند. و همه را در آرایه ریخته و برمی‌گردانیم.

# متد learn\_test\_model

در این قسمت ابتدا داده‌های تست و ترین را جدا می‌کنیم و هر ابسترکت به همراه لیبل‌هایش را در یک خط می‌نویسم. دو فایل test و train اماده شده را به txt تبدیل کرده و به ورودی الگوریتم train\_supervised می‌دهیم.

پس از آموزش این الگوریتم متد test را از مدل آموزش دیده فراخوانی می‌کنیم. داده‌های test را به آن پاس می‌دهیم. خروجی این الگوریتم در یک tuple به ما تعداد داده‌های تست، دقت و در نهایت فراخوانی را می‌دهد.

برای پیش‌بینی نیز می‌توان یک متن را predict مدل پاس داده که لییل‌های اون را پیشبینی و یک کانفیدنسی برای آن‌ها تعیین می‌کند.