به نام خدا

فاز سوم و چهارم پروژه بازیابی اطلاعات

محمدرضا صمدی ۹۵۳۱۰۵۲

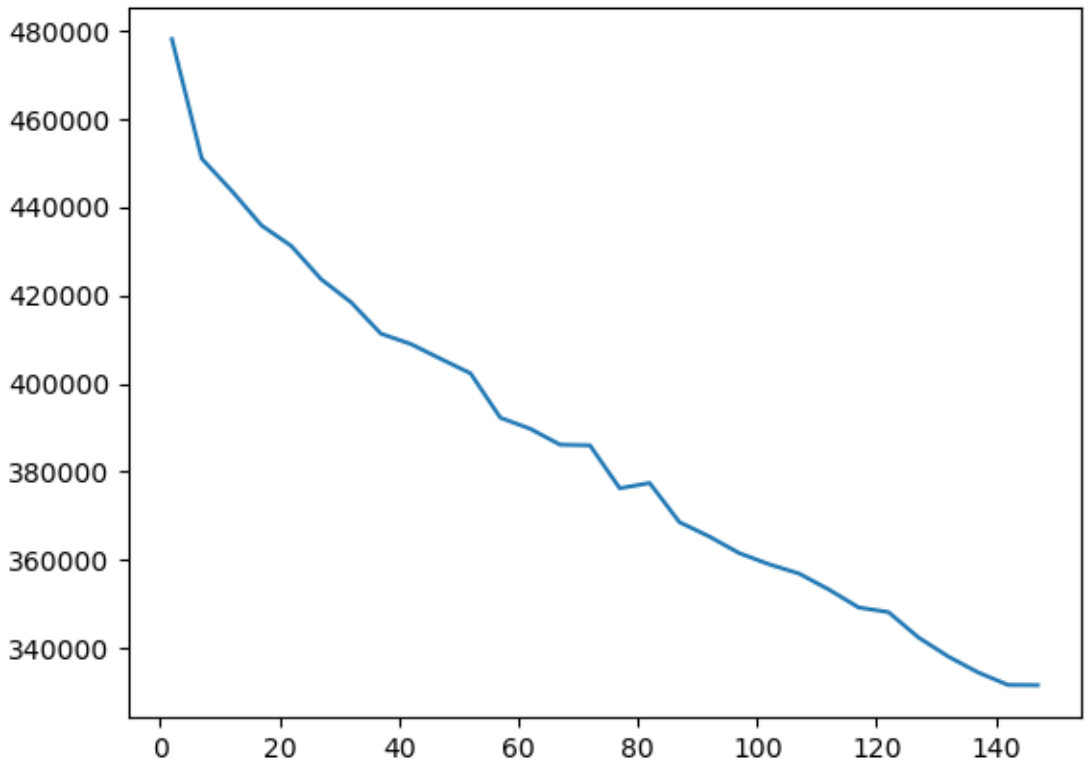
پارسا کاوه زاده ۹۵۳۱۰۷۱

محمد سامی ۹۵۳۱۰۹۵

- **K-means**

همانطور که در تعریف پروژه آمده است برای بهبود سرعت جستجو بهتر است اسناد آن را خوشه بندی کنیم و پس از آن درخواست کاربر را با مراکز هر خوشه شباهت کسینوسی بگیریم و خوشه مورد نظر را انتخاب کنیم. برای بهبود سرعت خوشه بندی k-means از ضرب های ماتریسی استفاده میکنیم که سرعتی بسیار بیشتر نسبت به حلقه های تو درتو دارد. علاوه بر آن برای بهبود استفاده از فضای مصرفی حافظه از بردار های sparse استفاده کردیم. تمامی بردار های اسناد نیز به صورت pickle شده در دیسک ذخیره می شود.

برای انتخاب مقدار k نیز از روش elbow استفاده کردیم به این صورت که با ۳۰ مقدار k توی بازه ۲ تا ۱۵۰ با فاصله ۵، مقدار خطای RSS را حساب میکنیم.در نهایت یک نمودار به شکل زیر با استفاده از مجموع خطا و مقدار k بدست خواهد آمد که بهترین K برابر با شکستگی نمودار است. همچنین برای هموار کردن نتایج درخواست کاربر از b بهترین مرکز استفاده میکنیم.



- **KNN**

برای این قسمت از دو الگوریتم knn و naive bayes استفاده کردیم. برای پیاده سازی این الگوریتم ها از کتابخانه sklearn استفاده کرده ایم. در ابتدا ۱۰۰۰ خبر اول فایل اول اخبار را به صورت دستی لیبل زدیم. سپس این ۱۰۰۰ داده لیبل خورده را به الگوریتم knn میدهیم. باید این ۱۰۰۰ خبر را از حالت فشرده شده ای که در قسمت قبل توضیح داده شد خارج کنیم و به صورت numpy array به الگوریتم knn بدهیم. سپس برای محاسبه داده های لیبل نخورده به صورت batch شده به الگوریتم میدهیم تا دسته آن ها مشخص شود. این کار به این دلیل انجام میگیرد تا مقدار numpy array ورودی به الگوریتم classification در ram قابل جادادن باشد. اما

مشکل اساسی در این قسمت sparse بودن بردار های هر سند است که برا حل این مشکل از کتابخانه scipy استفاده کردیم که فقط ایندکس های غیر یک یک بردار بسیار بزرگ را ذخیره میکند و از overflow شدن در ram جلوگیری میشود. برای بخش محاسبات و دسته بندی و خوشه بندی باید بردار ذخیره شده را به حالت numpy array برگردانیم و محاسبات را انجام دهیم.

در الگوریتم knn یک پارامتر ورودی k داریم. برای مشخص کردن کلاس یک داده لیبل نخورده باید k تا نزدیکترین داده لیبل خورده به آن داده را پیدا کنیم و بیشترین لیبل متعلق به آن k داده را به عنوان لیبل داده جدید انتخاب کنیم. برای اینکه حالت تساوی به وجود نیاید معمولا مقادیر k اعداد فرد هستند. رایجترین اعداد k عدد ۳ و ۵ هستند. هرچه k را بالاتر ببریم بایاس الگوریتم بیشتر میشود چون امکان انعطاف مدل برای نسبت دادن داده جدید به لیبل های ویژه کمتر میشود.

- **Crawler**

برای پیاده سازی خزشگر از همان معماری mrk استفاده کردیم. این بخش دارای دو بخش اصلی Front Queue و Back Queue است که در ادامه به توضیح آن میپردازیم.

1. Front Queue

خزشگر ابتدا باید مجموعه ای از RSS خبرگزاری ها را به صورت یک فایل .csv بخواند. این فایل .csv دارای یک ستون اولویت خزش نیز می‌باشد که مقداری بین 0 تا هرعدد دلخواه کمتر از تعداد لینک های RSS دارد. همانطور که میدانیم این بخش باید میزان تازه بودن اخبار هر خبرگزاری را

در روند خزش تاثیر دهد. به همین منظور براساس نرخ تولید خبر هر خبرگزاری مقدار نرخ تازگی آن را به روز رسانی میکنیم تا اگر یک خبرگزاری خیلی سریع اخبار را تولید میکند، اخبارش را زودتر به روز رسانی کنیم. اما همانطور که در تعریف پروژه آمده است رابطه به روز رسانی این نرخ خیلی ساده است چراکه معیار زمان در آن دخالتی ندارد. به صورت عمومی خبرگزاری ها در صبح اخبار بیشتری نسبت به شب و یا ظهر منتشر میکنند. به همین دلیل یک رابطه به روزرسان مناسب باید شامل معیار زمان باشد.

پس از اولویت بندی این RSS ها، آن هارا داخل صف ها با توجه به اولویت خزش قرار میدهیم. هنگام انتخاب یک RSS ابتدا صف های با اولویت بالاتر بررسی میشوند و اگر زمانی که برای آن ها اختصاص داده شده است کمتر از زمان فعلی باشد برای خزش به Back Queue داده میشود. اما قبل از اختصاص به صف پشت باید لینک های تکراری را از داخل آن حذف کنیم.

2. Back Queue

این بخش مسئولیت بررسی هر لینک و انجام عمل خزش را به عهده دارد. نکته قابل توجه در این بخش رعایت نرخ درخواست به سرور های میزبان هاست چرا که اگر این نرخ درخواست زیاد باشد سرور کند میشود و ممکن است دسترسی ما را مسدود کند. به همین منظور ما یک نرخ درخواست تعریف میکنیم که به هر میزبان با توجه به این مدت زمان درخواست بدهیم. از طرفی لینک ها در این بخش براساس میزبان هایشان تقسیم میشوند. همچنین یک جدول به منظور تبدیل ادرس میزبان به شماره صفی که لینک های مرتبط با آن در آن قرار گرفته اند ایجاد میکنیم که بوسیله آن لینک هایی که از صف جلو می آیند را بتوانیم به صف های مناسبی اختصاص دهیم. از طرفی برای آنکه بتوانیم لینک هایی که آماده ارسال درخواست هستند را مشخص کنیم از یک هیپ که براساس زمان درخواست به میزبان مرتب شده است استفاده میکنیم.

همچنین به منظور افزایش سرعت از چند Thread برای برداشت داده و شاخص گذاری استفاده میکنیم. یک Thread مسئول اضافه کردن کلمات و ترکیب های اسناد جدید به شاخص معکوس است. همچنین اطلاعات اسناد بازیابی شده هم داخل یک DataFrame ذخیره میشود.