دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش کارآموزی (هفته ششم)

محل کارآموزی: شرکت سامانه گستر سحاب‌پرداز

نام استاد کارآموزی

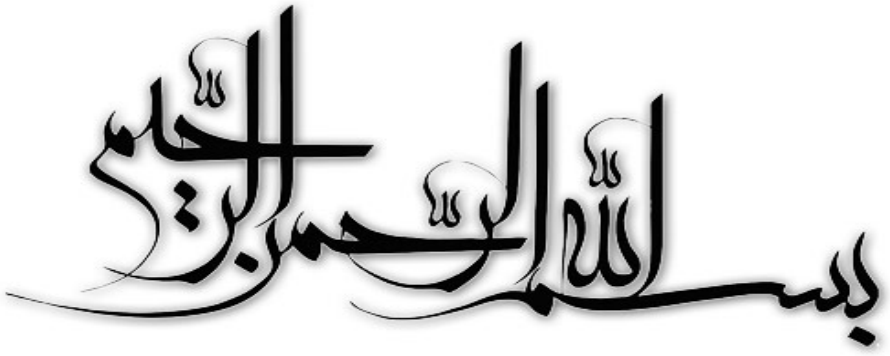
دکتر مسعود صبائی

نام دانشجو

امیرمحمد پیرحسینلو

۹۵۳۱۰۱۴

تابستان ۱۳۹۸


بسم الله الرحمن الرحيم

فهرست مطالب

فصل اول
بهبود عملکرد موتور جستجو

۱-۱ مقدمه

همان‌طور که در قسمت نتیجه‌گیری گزارش قبل ذکر شد، نتایج جستجو کیفیت خوبی ندارند. به همین دلیل مدیران تیم و اعضای شرکت پیشنهاداتی برای بهبود نتایج مطرح کردند که شامل موارد زیر می‌شود:

- بهبود جستجو با محاسبه تعداد لینک‌های ورودی
- بهبود جستجو با اعمال متن **anchor link** ها
- بهبود جستجو با **Page Rank**

به دلیل کمبود وقت و صرف وقت برای ارائه نهایی، تنها فرصت شد تا مورد اول را پیاده سازی کنیم. به همین دلیل در موارد دیگر تنها به توضیح بخشی از روند کاری که باید انجام شود می‌پردازیم.

۱-۲ بهبود جستجو با محاسبه تعداد لینک‌های ورودی

برای این‌که نتایج جستجو کیفیت بیشتری داشته باشند، باید صفحات مهم‌تر، شانس بیشتری برای ظاهر شدن در نتایج جستجو داشته باشند. ساده‌ترین راه، پیدا کردن تعداد لینک‌ها به یک صفحه است. اگر به یک صفحه، تعداد ارجاعات بیشتری باشد، طبعا مهم‌تر است. تعداد ارجاعات را با **Map/Reduce** پیدا می‌کنیم، آن‌گاه آن‌ها را به یک امتیاز تبدیل کرده و در کنار امتیازی که **ElasticSearch** برای **query** ها در نظر می‌گیرد به کار برده و در نتیجه‌ی نهایی تاثیر می‌دهیم.
برای پیاده‌سازی موارد بالا، دو برنامه نوشته شده است:

- برنامه اول برنامه‌ای است که تعداد لینک‌ها به یک صفحه را محاسبه کرده و در یک جدول در **HBase** قرار می‌دهد. این برنامه همراه گزارش ضمیمه شده است. برای اجرای آن کافی است مراحل زیر طی شوند:
  یک **instance** از **HBase** با دستور **start-hbase.sh** اجرا شود.
  یک جدول با نام **InnerlinksTable** با ستون فامیلی [۱] **NumOfLinks** ساخته شود.
  java -jar innerlinks-calculator.jar
- برنامه‌ی دوم برنامه‌ای است که از کاربر **query** می‌گیرد و با در نظر گرفتن تعداد لینک‌های ورودی به هر صفحه و تطابق محتوای هر صفحه با کلمات **query** ورودی، بهترین نتایج را به صورت نزولی نمایش می‌دهد.
  برای اجرای این برنامه باید مراحل زیر طی شود:
  اجرای **HBase** با دستور **start-hbase.sh**
  اجرای **ElasticSearch** با اجرای دستور **sudo systemctl start elasticsearch.service**
  java -jar query-processor-with-innerlinks.jar

۱-۳ بهبود جستجو با اعمال متن **anchor link**

اگر صفحات عنوان خوبی داشتند، عالی بود ولی خیلی از صفحات یا عنوان خوبی ندارند یا در عنوان‌شان فقط به یکی از جنبه‌های اطلاعاتی که می‌توان در آن صفحه یافت اشاره می‌کنند. حال چگونه می‌توان عناوین خوب تولید کرد؟ پاسخ، متن‌هایی است که صفحات دیگر

[۱] **Column Family**

برای آن صفحه انتخاب کرده‌اند یا همان متن **anchor link** ها.

حال باید با **Map/Reduce** ، **anchor text** های پر تکرار برای هر صفحه را بیابیم (البته نه عبارات بی اثری مانند «این‌جا» یا «لینک») و آن‌ها را در کوئری‌ها تاثیر دهیم تا جستجوهای بهتری داشته باشیم.

۱-۴ بهبود جستجو با Page Rank

برای محاسبه‌ی **Page Rank** ، به سراغ فریم‌ورک **Spark** می‌رویم. اما این دو چه هستند؟

- **Page Rank** الگوریتمی است که شرکت گوگل برای امتیازدهی به صفحات وب در موتور جستجویش از آن استفاده می‌کند. این الگوریتم با محاسبه‌ی تعداد لینک‌ها به یک صفحه و کیفیت آن‌ها میزان اهمیت آن صفحه را پیدا کرده و به آن یک عدد نسبت می‌دهد. فرض بر این است که سایت مهم‌تر تعداد ارجاعات بیشتری از طرف سایت‌های دیگر دارد. اطلاعات بیشتر در آدرس زیر:

https://en.wikipedia.org/wiki/PageRank

- **Spark**[۲] یک فریم‌ورک برای انجام محاسبات در یک خوشه از کامپیوترها **(cluster)** است. پردازش موازی داده‌ها در خوشه را امکان‌پذیر می‌سازد و به طور ضمنی **fault tolerant** است. کتابخانه‌های زیر از **Spark** استفاده می‌کنند:
  - **MLIB**
  - **GraphX**
  - **Spark SQL**
  - **Spark Streaming**

به دلیل این‌که **Spark** از عملیات‌های درون حافظه‌ای [۳] استفاده می‌کند، تا ۱۰ برابر سریع تر از **YARN** عمل می‌کند و این یکی از دلایلی است که به سراغ آن می‌رویم.

۱-۵ نتیجه‌گیری

در این دوره کارآموزی با مفاهیم و ابزارهای جدید در حوزه‌ی داده‌های حجیم آشنا شدم و همچنین کار با ابزارهای **maven** ، **git** و سیستم عامل لینوکس را فراگرفتم. همچنین در یک گروه ۶ نفره یک کار تیمی جدی را تجربه کردم. امیدوارم این گزارش مثمر ثمر واقع شود.

[۲] https://en.wikipedia.org/wiki/Apache_Spark
[۳] **in-memory operatios**