دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش کارآموزی (هفته پنجم)

محل کارآموزی: شرکت سامانه گستر سحاب‌پرداز

نام استاد کارآموزی

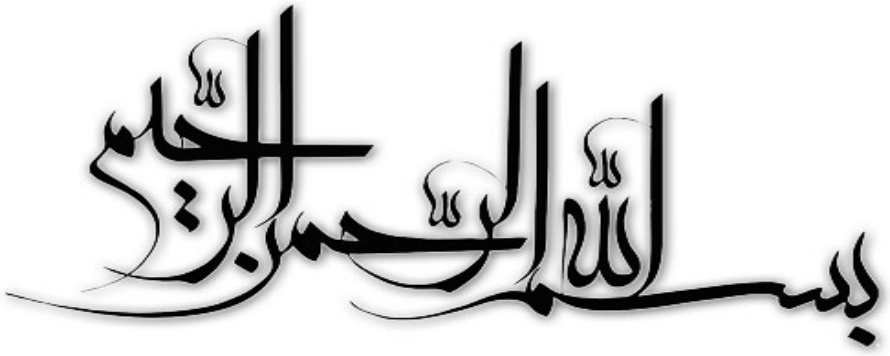
دکتر مسعود صبائی

نام دانشجو

امیرمحمد پیرحسینلو

۹۵۳۱۰۱۴

تابستان ۱۳۹۸


بسم الله الرحمن الرحيم

فهرست مطالب

فصل اول
پیاده سازی پروژه

۱-۱ مقدمه

در این هفته به پیاده سازی و برنامه نویسی موتور جستجو پرداختیم. اعضای تیم به برنامه نویسی بخش‌های مختلف پروژه پرداختند و در انتها با ترکیب کدها که در بستر **git** ذخیره شده بودند، برنامه‌ی واحدی که همان موتور جستجو است ساخته شد. این برنامه از دو بخش تشکیل شده است:

- خزنده: یک برنامه که با زبان جاوا نوشته شده است و وظیفه‌ی آن خزش **(crawl)** کردن صفحات وب و ذخیره اطلاعات آن است و
- **search.sh**: یک اسکریپت است که از کاربر **query** گرفته و آن‌را در بین اسناد جستجو می‌کند و نتیجه را برمی‌گرداند. نتیجه شامل اسنادی است که شامل **query** هستند یا عبارات **query** در آن‌ها حضور دارد. نتایج با توجه به امتیازشان به صورت نزولی مرتب شده و نمایش داده می‌شوند (بهترین سند یافت شده اول نمایش داده می‌شود.).

در ادامه به توضیح پکیج‌ها و کلاس‌های پروژه می‌پردازیم.

۱-۲ پکیج‌ها

پروژه از پکیج‌های زیر تشکیل شده است:

۱-۲-۱ crawler

این پکیج شامل تابع **main** است. در این تابع صف لینک‌ها با لینک **(URL**[1]**)** های اولیه پر می‌شود و نخ **(thread)** آمارگیری (کلاس **Statistics**) اجرا می‌شود. همچنین به تعداد نیاز از کلاس‌های **Fetcher** و **Parser** که خود نخ هستند نمونه ساخته شده و اجرا می‌شوند.
این پکیج شامل کلاس‌های زیر است: (کلاس‌ها در ادامه توضیح داده خواهند شد.)

- **Crawler**
- **Fetcher**
- **Parser**
- **LruCache**

۱-۲-۲ index

این پکیج شامل کلاس **Elastic** است که وظیفه برقراری ارتباط با **ElasticSearch** و تبادل داده با آن را برعهده دارد.

۱-۲-۳ storage

این پکیج شامل کلاس **HBase** است که وظیفه برقراری ارتباط با **HBase** و تبادل داده با آن را برعهده دارد.

[1] **Uniform Resource Locator**

test ۴-۲-۱

شامل کلاس‌هایی است که برای تست سایر کلاس‌های نرم‌افزار به کار می‌روند.

utils ۵-۲-۱

کلاس‌های زیر در این پکیج قرار می‌گیرند:

- **Constant**
- **Pair**
- **Prints**
- **Statistics**

این کلاس‌ها در ادامه توضیح داده می‌شوند.

language ۶-۲-۱

شامل کلاس‌های مورد نیاز برای تشخیص زبان یک متن است.

۳-۱ کلاس‌ها

برای پیاده سازی پروژه از کلاس‌های مختلفی استفاده شده است که در ادامه به تشریح آن‌ها می‌پردازیم:

Crawler ۱-۳-۱

تابع **main** در این کلاس قرار دارد. عملیات انجام شده در این تابع در قسمت پکیج **crawler** توضیح داده شد.

Fetcher ۲-۳-۱

این کلاس خود یک نخ است و وظیفه دارد که به طور متناوب از صف لینک‌ها **(URLs queue)** ، لینک **(URL)** دریافت کرده و محتوای آن را دانلود کند سپس لینک و محتوایش **(document)** را در صف لینک-سند **(url-document)** قرار دهد. برای انجام این عملیات از توابع زیر استفاده شده است:

fetchURL

یک لینک از سر صف لینک‌ها برداشته و برمی‌گرداند.

getDomainIfLruAllowed

بررسی می‌کند که آیا به **Host** لینک مورد نظر در ۳۰ ثانیه اخیر درخواستی داده شده است یا خیر. در صورت مثبت بودن، این لینک مجددا در انتهای صف لینک‌ها قرار می‌گیرد و لینک دیگری اخذ می‌شود.

fetch

محتوای یک لینک را دانلود کرده و برمی‌گرداند.

putFetchedData

لینک و محتوای آن را در صف لینک-سند قرار می‌دهد.

LruCache ۳-۳-۱

در این کلاس پیاده سازی یک **LruCache** انجام شده است.

Parser ۴-۳-۱

این کلاس خود یک نخ است که به طور متناوب از صف لینک-سند داده (لینک و سند مربوط به آن) ر ا استخراج می‌کند. محتوای لینک را **parse** کرده و موارد زیر را از آن استخراج می‌کند:

- عنوان لینک **(title)**
- **anchor link** ها و **anchor text** ها
- تگ‌های **<p>** (پاراگراف‌ها)

پس از استخراج اطلاعات بالا، عملیات‌های زیر انجام می‌شود:

- قرار دادن لینک و عنوان آن و پاراگراف‌ها در **ElasticSearch**
- قرار دادن لینک، **anchor link** ها و **anchor text** ها در **HBase**
- تحویل **anchor link** ها به **Kafka**

برای انجام عملیات‌های بالا از توابع زیر استفاده شده است:

takeFetchedData

از سر صف لینک-سند داده برداشته و برمی‌گرداند.

extractLinkAnchors

anchor link های یک سند را استخراج می‌کند و برمی‌گرداند.

putToElastic

لینک و عنوان آن و پاراگراف‌های سند مربوط به لینک را در **ElasticSearch** قرار می‌دهد.

putAnchorsToHBase

لینک، **anchor link** های درون سند مربوط به لینک و **anchor text** ها را در **HBase** قرار می‌دهد.

putToKafka

anchor link های مربوط به یک سند را به **Kafka** تحویل می‌دهد.

Elastic ۵-۳-۱

رابط برنامه و موتور جستجوی **ElasticSearch** است. مهم‌ترین تابع آن **indexData** است که لینک و عنوان آن و پاراگراف‌های سند مربوط به آن را دریافت کرده و داخل **ElasticSearch** قرار می‌دهد.

HBase ۶-۳-۱

رابط برنامه و پایگاه داده‌ی **HBase** است. مهم‌ترین تابع آن **insertLinks** است که لینک، **anchor link** های درون سند مربوط به لینک و **anchor text** ها را گرفته و داخل **HBase** قرار می‌دهد.

Constants ۷-۳-۱

مقادیر ثابت و پارامترهای برنامه در این کلاس قرار دارند.

Statistics ۸-۳-۱

کلاسی است که خود یک نخ است و وظیفه دارد به صورت تناوبی وضعیت سایر نخ‌ها (**Parser** ها و **Fetcher** ها) را گزارش دهد.

Pair ۹-۳-۱

پیاده سازی یک ساختار داده برای نگه‌داری یک کلید و مقدار متناظر آن است.

استثناها ۴-۱

موتور جستجوی ساخته شده تا حد بسیار زیادی مطابق با اطلاعات داده شده در این گزارش و گزارش‌های قبلی است و فقط در چند مورد زیر تفاوت دارد که علت آن نیز ذکر شده است:

- عدم استفاده از پکیج تشخیص زبان: به دلیل خطای بالای تشخیص زبان و زمان زیادی هم که می‌گرفت، از کلاس‌های این پکیج استفاده نشد. در اسرع وقت راه‌حلی برای مشکل پیش آمده پیدا می‌کنیم.
- عدم استفاده از **Kafka**: به جای **Kafka** ، **anchor link** ها را مستقیما در صف لینک‌ها قرار دادیم و عدم مشاهده آن‌ها تاکنون را به یک صف دیگر که مدیریت آن بر عهده **HBase** است، سپردیم. دلیل عدم استفاده از **Kafka** ، عدم پیکربندی مناسب و از کار افتادن بی مورد آن بود. در اسرع وقت سعی می‌کنیم این مشکل را حل کنیم.

اجرای برنامه ۵-۱

برنامه توسط ابزار مدیریت پکیج **maven**[۲] ساخته **(build)** شده است. نسخه‌ای که در کنار گزارش قرار دارد نسخه‌ای است که بر روی یک سرور اجرا می‌شود. برای مشاهده نسخه اصلی کد که در دو سرور اجرا می‌شود به آدرس زیر مراجعه کنید:

https://github.com/Ahmad535353/NimboSearchEngine

برای اجرای بخش خزنده برنامه کافی است روند زیر طی شود:

- اجرای **Hadoop** ، **HBase** و **ElasticSearch**

[۲] https://maven.apache.org

- ایجاد یک جدول به‌نام **CrawlerTable** در **HBase** با فامیلی ستون [۳] **data**
- **java -jar Crawler.jar**

برای ارسال **query** و دریافت پاسخ، کافی است فایل اجرایی **search.sh** را اجرا کنید. با اجرای فایل از شما درخواست می‌شود تا **query** را وارد کنید. پس از قرار دادن **query** و زدن دکمه **Enter** ، پاسخ دریافت و نمایش داده می‌شود.

۱-۶ موارد باقی مانده

به دلیل ضیق وقت، فرصت نشد تا موارد زیر پیاده سازی شوند. سعی می‌شود در اسرع وقت این موارد پیاده سازی شوند:

- پشتیبانی گرفتن از صف‌های لینک، لینک-سند و صف کلاس **HBase** و جلوگیری از دست رفتن داده‌ها (**data loss**) هنگام قطع شدن برق.
- فیلتر کردن برخی پروتکل‌ها مانند **FTP** و ...
- برطرف کردن مشکل **Kafka**
- پیاده سازی و یا پیدا کردن یک ماژول تشخیص دهنده زبان یک متن
- جایگزین کردن کلاس‌های منسوخ شده **(deprecated)** برنامه مانند کلاس‌های استفاده شده در کلاس **Elastic** و ...

۱-۷ نتیجه گیری

با اتمام پیاده سازی، یک موتور جستجو در اختیار ماست که فقط «کار» می‌کند. نتیجه **query** ها اصلا خوب نیستند. تیم شرکت به ما توصیه کردند که از الگوریتم **Page Rank**[۴] و مدل برنامه نویسی **MapReduce** برای امتیازدهی سایت‌ها و رتبه بندی آن‌ها استفاده کنیم و همچنین نواقص قبلی را برطرف کنیم. برای پیاده سازی **Page Rank** از فریم‌ورک **Spark**[۵] استفاده می‌کنیم. پیاده سازی باقی مانده و رفع نواقص قبلی در هفته آینده انجام خواهد شد.

[۳] **Column Familiy**
[۴] https://en.wikipedia.org/wiki/PageRank
[۵] https://spark.apache.org/

واژه‌نامه‌ی انگلیسی به فارسی

B

build ساختن. در این‌جا منظور ساخت فایل **jar** است.

C

crawl . . . . . خزیدن

D

deprecated . . . . . منسوخ شده

document . . . . . سند

T

thread . . . . . نخ

title . . . . . عنوان