به نام خدا

گزارش فردی پروژه پایانی درس تحلیل‌ها و سیستم‌های داده‌های حجیم

لینک گیت‌هاب پروژه: https://github.com/zolfaShefreie/Online-Taxi-Service.git

وظایف انجام‌شده در این پروژه به‌ترتیب در بخش‌های بعدی توضیح داده می‌شوند.

گام اول

وظایف انجام‌شده در این گام شامل تعریف تولیدکننده[1] و مصرف‌کننده[2]، ارسال داده‌ها به کانال kafka و مرتب‌سازی مجموعه داده ورودی می‌باشد.

قطعه کد مربوط به تعریف مصرف‌کننده در تابع _normal_setup در فایل base_classes.py، نام topic که باید داده‌های آن را مصرف کند، به‌عنوان ورودی دریافت می‌کند. آرگمان value_serializer در قطعه کد مربوط به تعریف تولیدکننده در همین تابع، داده‌های پردازش‌شده در هر گام را به قالب json تبدیل می‌کند. کد ارسال داده به کانال kafka در تابع _send_data همین فایل، داده‌های تولید و پردازش‌شده هر گام را در topic مشخص‌شده ذخیره می‌کند.
در تابع _normal_run فایل file_stream.py، ابتدا مجموعه داده توسط تابع read_csv کتابخانه pandas و با در نظر گرفتن سطر نخست به‌عنوان نام ستون‌ها، خوانده می‌شود. سپس در صورت مرتب نبودن مجموعه داده، با استفاده از تابع sort_values مجموعه داده برحسب ستون Date/Time مرتب شده و نتیجه در قالب یک فایل csv و با نام معین‌شده، ذخیره می‌شود.

گام چهارم

وظایف انجام‌شده در این گام شامل اتصال elasticsearch به kafka، دریافت داده‌ها از کانال kafka، ساخت index در elasticsearch و ذخیره داده‌ها در index ایجادشده می‌باشد.

کدهای مربوط به این بخش در فایل elastic_analyse.py نوشته شده است. ابتدا متغیر کلاس es از طریق localhost elasticsearch را به سرور محلی متصل می‌کند. ساخت index در تابع _generate_index انجام می‌شود. به‌این‌صورت که در صورت عدم وجود indexای با نام taxi_service_ondex، index جدیدی با این نام و با ویژگی‌های Date/Time در قالب تاریخ، Lat و Lon در قالب عدد اعشاری و Base در قالب رشته که بیانگر ستون‌های مجموعه داده هستند، ویژگی Location برای نگهداری مختصات جغرافیایی در قالب geo_point (مورد استفاده در مراحل بعدی این گام) و ویژگی Cluster_number در قالب رشته که نشان‌دهنده شماره خوشه است، ایجاد می‌کند. ابتدا تابع _normal_run این کلاس فراخوانی می‌شود که در این تابع نیز نخست تابع _generate_index فراخوانی می‌شود. سپس تابع _normal_setup برای دریافت مقادیر topic مروطه از کانال kafka صدا زده می‌شود. برای هر یک از داده‌های دریافتی، تابع

[1] Producer

[2] Consumer

produce_answer_ فراخوانی شده و داده‌ها در index ساخته‌شده ذخیره می‌شوند. یک id یکتا نیز به‌ازای هر داده ایجاد و به‌همراه داده در ذخیره می‌شود. کلید داده‌ها، timestamp و داده دریافتی بازگردانده شده و توسط تابع send_data_ در topic مربوطه نوشته می‌شوند.

در فایل kaka_management.py شئ از این کلاس ایجاد می‌شود. topicهای مصرف‌کننده و تولیدکننده در این مرحله و با ایجاد شئ مشخص می‌شوند. topic مصرف‌کننده حاوی خروجی گام اول یعنی داده‌های مرتب‌شده است.

گام پنجم

وظایف انجام‌شده در این گام شامل اتصال cassandra به kafka، دریافت داده‌ها از کانال kafka و ایجاد جداول و ذخیره داده‌ها در آن‌ها است.

کدهای مربوط به این بخش در فایل cassandra_analyse.py نوشته شده است. متغیر کلاس KEYSPACE_NAME نام keyspace ایجادشده برای این پروژه را نگهداری می‌کند. متغیر شئ first برای تشخیص اولین داده تعریف شده است. متغیرهای week_keys، midday_keys و month_keys لیست‌هایی برای نگهداری کلیدهای جداول و متغیرهای week_values، midday_values و month_values دیکشنری‌هایی برای نگهداری داده‌های جداول هستند. با توجه به این که برای کلیدهای هفتگی، ۱۲ساعته و ماهیانه بیش از یک داده وجود دارد، مقادیر این دیکشنری‌ها به‌صورت لیست تعریف شده‌اند. جدول ماهیانه، بازه زمانی یک ماهه را ذخیره می‌کند و در گام هفتم استفاده می‌شود. برای ساخت cluster، سازنده کتابخانه Cluster فراخوانی می‌شود. ساخت session نیز با استفاده از تابع connect انجام شده است. به‌منظور cast کردن خودکار دستورات insert به شکلی قابل فهم برای cassandra، از دستور encoder.cql_encode_tuple استفاده شده است. دستورات ساخت جداول در تابع create_tables_ آمده است. در این تابع ابتدا وجود keyspace بررسی می‌شود. پس از آن به‌ترتیب جداول مربوط به کلید زمانی هفته‌ای، مختصات نقطه شروع، کلید uuid، کلید زمانی ۱۲ساعته و کلید زمانی ماهانه ایجاد می‌شوند. تابع separation_ برای جداسازی مقادیر داده‌ها می‌باشد. توابع insert_week_table_، insert_midday_table_، insert_month_table_، insert_start_coordinates_ و insert_uuid_ به‌ترتیب برای ذخیره داده در جدول‌های کلیدهای زمانی هفته‌ای، ۱۲ساعته و ماهانه، مختصات نقطه شروع و کلید uuid تعریف شده‌اند. در تابع insert_start_coordinates_ ابتدا وجود مختصات داده دریافتی بررسی می‌شود. در صورتی که داده‌ای با مختصات نقطه شروع مشابه در جدول وجود داشته باشد، سطر مربوطه به‌روز شده و داده دریافتی نیز به آن افزوده می‌شود. در غیر این صورت، داده دریافتی به‌عنوان سطر جدیدی در جدول درج می‌شود. تابع check_intervals_ برای بررسی کلیدهای زمانی هفته‌ای، ۱۲ساعته و ماهانه می‌باشد. در صورتی که داده دریافتی نخستین داده باشد، تاریخ این داده به‌همراه فاصله زمانی هفت روزه بعد از آن به‌عنوان عناصر لیست week_keys مقداردهی می‌شوند. در غیر این صورت تا زمانی که بازه انتهایی تاریخ‌های درون این لیست عقب‌تر از تاریخ دریافتی باشند، داده‌های نگهداری‌شده درون جدول مربوطه درج می‌شوند. سپس بازه زمانی جدید در لیست مقداردهی شده و داده‌های قبلی که در جدول درج شده‌اند از مقادیر دیکشنری week_values حذف می‌شود. درنهایت بخش‌های مختلف داده دریافتی در لیست‌های متناظر با مقادیر دیکشنری week_values مقداردهی می‌شوند. اگر تاریخ دریافتی در بازه قبلی قرار داشته

باشد و متعلق به هفته جدیدی نباشد، داده جدید دریافتی به مقادیر قبلی (که همگی به یک کلید تعلق دارند) اضافه می‌شود و اگر داده دریافتی به هفته جدیدی تعلق داشته باشد، به لیست‌هایی که در مرحله قبل خالی شده‌اند افزوده می‌شوند. همین روند برای کلید زمانی ۱۲ساعته و ماهانه نیز تکرار می‌شود.
در این گام نیز مشابه گام چهارم، تابع normal_run_ کلاس فراخوانی می‌شود. درون این تابع پس از فراخوانی تابع normal_setup_ ، تابع create_tables_ صدا زده می‌شود. سپس به‌ازای هر داده دریافتی تابع produce_answer_ و درون آن توابع separation_ و check_intervals_ فراخوانی می‌شوند. در انتها session و connection بسته می‌شوند. مشابه گام قبل، در فایل kaka_management.py شئ از این کلاس ایجاد می‌شود. topicهای مصرف‌کننده و تولیدکننده در این مرحله و با ایجاد شئ مشخص می‌شوند. topic مصرف‌کننده حاوی خروجی گام چهارم یعنی داده‌های ذخیره‌شده در elasticsearch است.

گام ششم

وظایف انجام‌شده در این گام شامل اتصال redis به kafka، دریافت داده‌ها از کانال kafka و ذخیره داده‌ها است.

ابتدا داده‌های دریافتی مشابه مرحله قبل جداسازی می‌شوند. این کار در تابع separation_ انجام شده است. کلیدهای redis تاریخ و زمان داده‌ها به‌همراه پیشوند day یا hour در نظر گرفته شده است. با دریافت هر داده، تابع add_ فراخوانی شده و به‌کمک تابع rpush داده جدید اضافه می‌شود. در صورتی که کلید داده جدید وجود داشته باشد (مثلا داده‌های یک روز دریافت شود) داده جدیدی به لیست مقادیر همان کلید افزوده می‌شود. در غیر این صورت، کلید جدیدی ایجاد می‌شود. پس از اضافه کردن داده، تابع delete_ صدا زده می‌شود. این تابع یک هفته و یک روز قبل از تاریخ و زمان داده دریافتی را محاسبه می‌کند و در صورتی که کلیدی با این زمان وجود داشته باشد، آن را حذف می‌کند. با این کار، با شروع هفته یا روز جدید، یک روز یا یک ساعت از هفته یا روز قبل حذف می‌شود. تابع get_previous_six_hour_ ابتدا تاریخ و زمان شش ساعت پیش از داده دریافتی کنونی را محاسبه می‌کند. سپس در بین تمامی کلیدهای مربوطه به ساعت، کلیدهایی که در بازه زمانی شش ساعت گذشته قرار می‌گیرند به‌دست آورده شده و مقدار آن‌ها توسط تابع lindex دریافت می‌شود. برای یافتن نقطه خاص در شش ساعت گذشته، مختصات نقاط مقادیر به‌دست آمده از مرحله قبل، با نقطه ورودی مقایسه می‌شود. تابع previous_hour_ نیز برای دریافت داده‌های مربوط به یک ساعت گذشته است و روند آن مشابه تابع قبلی است.