Memsource Big Data Engineer Homework Assignment

Goals

The assignment asks to create a simple streaming data application that will aggregate one metric. This document should come with:

1. a Maven skeleton project written in Java and Kafka Streams
2. “src/main/resources/kafka-messages.jsonline” file containing a sample data set. Its schema is described in a separate section later in this document

As you might remember from the demo during the interview, every document uploaded to Memsource is split into **segments** (segmentation). The data of these **segments**, and all the modifications done to them during translation, live in MongoDB. Whenever the data changes, a MongoDB CDC event is captured by Debezium (https://debezium.io/docs/connectors/mongodb/) and sent to a Kafka topic. A short record/log of such events is provided “kafka-messages.jsonline” sample file, one event per line. Your goal is to process these events and count the number of **confirmed segments** of each **task** (document), storing the results in a Kafka topic with the key being ***taskId*** of the **task** and the value being the number of **confirmed segments** in this **task**. The sample dataset contains short record of modifications done to **segments** of two **tasks**. After aggregating all the sample data, you should end up with one **task** having zero **confirmed segments** and one **task** having two **confirmed segments**. Note that:

- during the translation process, an already **confirmed segment** can be unconfirmed
- there can be multiple modifications recorded for a **segment**, regardless of it being confirmed or not

Terminology

Task – a document uploaded to Memsource for translation. A **task** is made of 0 to N tgroups

Tgroup – the smallest unit of work within **task**. For the sake of this assignment, you can assume that each **tgroup** contains exactly one **tunit**

Tunit – another name for **segment**. This is usually one sentence that needs to be translated.

Confirmed segment – a **segment** that was deemed translated by the linguist/translator. A **Segment** is considered confirmed when the first element of ***levels*** is equal to ***tUnits[0].confirmedLevel***

Notes

- the skeleton project is provided for your convenience to get you up and running quicker. It is not required to use it as a starting point, you can use any language or streaming data technology to solve the homework (e.g. Scala, Python, Spark Structured Streaming, Apache Flink, …)
- treat the code you create as production-level source code
- please, deliver the assignment in a publicly available repository, e.g. GitHub

- our aim is for the assignment to not take longer than 4-8 hours, depending on your experience. Please, let us know how long it took you to finish it.
- With proper implementation, re-sending the sample data into the source topic should be an idempotent operation with regards to the expected result

Sample data JSON Schema

```
{
  "op": string, // can be "c", "u", "d"
  "after": string, // contains JSON as a string, if "op" is equal to "c". See its schema below
  "patch": string // contains JSON as a string, if "op" is equal to "u". Same schema as "after"
}
{
   "taskId": string,
   "tGroupId": number,
   "levels": [number],
   "tUnits": [
      {
        "tUnitId": string, // made from "<taskId>:<tGroupId>"
        "confirmedLevel": number
      }
   ]
 }
```

- We suggest working only documents where ***op*** is equal to “c” and where ***levels*** is not null and contains at least one element
- You can assume the ***tUnits*** array will always contain exactly one element

Bootstrapping the skeleton project

- We recommend starting Kafka through Docker. A nice tutorial is located here - https://docs.confluent.io/current/quickstart/ce-docker-quickstart.html
- It is sufficient to use “examples/kafka-single-node” for the homework o We suggest exposing Zookeeper port by adding the following to “docker-compose.yml”

```
ports:
  - 2181:2181
```

- The skeleton project assumes the Kafka topic that will contain the segment modification records is called “homework”. You can send the content of the sample dataset into such topic with

  ```
  cat src/main/resources/kafka-messages.jsonline | <kafka/confluent-home>/bin/kafka-
  consoleproducer --broker-list localhost:9092 --topic homework
  ```