

Sistema de Recomendação em Fluxos Contínuos de Dados

Relatório de prática em Laboratório

Aluno: Carlos Alberto Rocha Cardoso

Professor: Pedro Kássio

Novembro 2019

1. Introdução

Este relatório descreve prática realizada em laboratório para desenvolvimento de um sistema de recomendação de filmes, aplicando técnicas de aprendizado de máquina e processamento de fluxos contínuos de dados, integrando os softwares Flume, Kafka, MongoDB e Spark Streaming. Para gerar as recomendações foi utilizada a técnica de filtro colaborativo, por meio de um algoritmo de vizinhos mais próximos (KNN) aplicado a um dataset contendo os filmes e avaliações dos usuários. A partir das recomendações geradas e armazenadas, foi desenvolvido um pipeline capaz de atender de forma contínua às solicitação de recomendação de filmes dado um identificador de usuário.

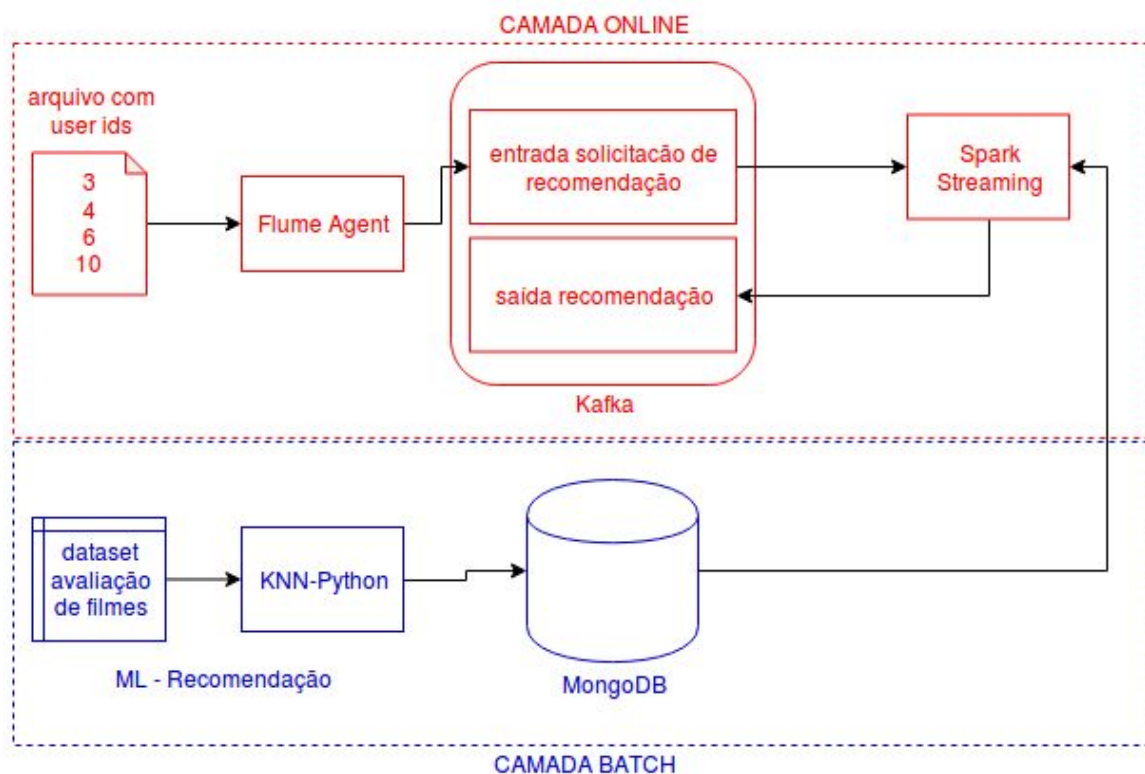
2. Objetivo

São objetivos da prática descrita neste relatório a aplicação e consolidação dos conhecimentos compartilhados nas disciplinas de Projeto Integrado e Processamento e Análise de Fluxos Contínuos de Dados, permeando as etapas de coleta, processamento, armazenamento e visualização de dados. Além da experimentação de softwares utilizados na construção de sistemas dessa especialidade, auxiliando no desenvolvimento e aperfeiçoamento de habilidades relacionadas.

3. Experimentos

O diagrama a seguir apresenta o pipeline de fluxo e processamento contínuo de dados, implementado na prática em laboratório e descrito neste documento. Ele é formado por duas camadas: a batch, na parte inferior em azul, e online, na parte superior em vermelho. A camada batch é responsável por processar os datasets de filmes e avaliação de usuários, gerar e armazenar as recomendações de filmes para cada usuário. Já a camada online é responsável por coletar e responder às solicitações de recomendação para um determinado usuário, com base nas recomendações armazenadas pela camada batch.

Nas próximas seções é apresentada a construção e execução de cada uma das camadas desse diagrama.



3.1. Algoritmo de recomendação

A primeira etapa desta prática consistiu no entendimento e execução do programa python responsável por gerar recomendações de filmes para cada usuário, a partir do algoritmo de seleção de vizinhos mais próximos (KNN). O programa completo, em python, consta no Anexo I deste relatório.

Abaixo o algoritmo implementado pelo programa:

- Ler dos dados de avaliação dos filmes.
- Transformar os dados, incluindo eliminar usuários que tenham visto poucos filmes e filmes com poucas avaliações.
- Construir uma matriz de avaliação (ratings) contendo o id do usuário, o id do filme e a nota dada pelos usuários para cada filme assistido.

A partir desse passo, para cada usuário:

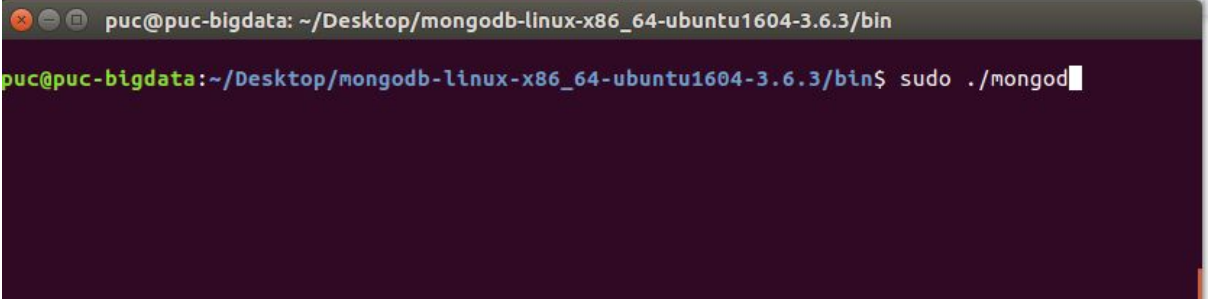
- Encontrar N usuários mais similares. Para esse passo foi utilizado algoritmo que implementa o cálculo da distância de Hamming .
- Retornar a matriz de avaliação para os N usuários mais similares.
- Excluir dessa matriz os filmes sem avaliação.
- Excluir dessa matriz os filmes já assistidos pelo usuário.
- Ordenar essa matriz pelas avaliações mais positivas.
- Retornar os N filmes mais recomendados a partir da matriz ordenada.

3.2. Armazenamento das recomendações

Nessa etapa foi desenvolvido e inserido no programa python original um trecho de código responsável por armazenar as recomendações de filmes conforme resultado do algoritmo de recomendação. A seguir as etapas executadas:

3.2.1. Inicialização do MongoDB

Com o MongoDB instalado, foi utilizado o seguinte comando para inicialização:



```
puc@puc-bigdata: ~/Desktop/mongodb-linux-x86_64-ubuntu1604-3.6.3/bin
puc@puc-bigdata:~/Desktop/mongodb-linux-x86_64-ubuntu1604-3.6.3/bin$ sudo ./mongod
```

3.2.2. Código responsável por armazenar recomendações

```
from pymongo import MongoClient

#Conexão com o MongoDB
client = MongoClient()
db = client['movies']
db_recomendacoes = db['knn_recomendacoes']

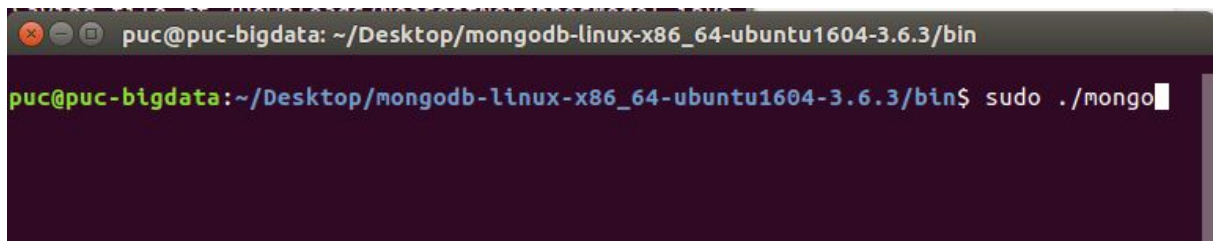
#Número de recomendações desejadas
nrec = 10
#Recupera lista de usuários a partir da matriz de avaliações
usuarios = pd.DataFrame(userItemRatingMatrix.index)

for uid in usuarios["userId"]:
    #topN retorna os N filmes mais recomendados com base no algoritmo de vizinhos mais próximos
    recs = topN(uid,nrec)

    #Cria o documento contendo as recomendações para o usuário
    rec_doc = {
        "userId": uid,
        "recs": recs["title"].values.tolist(),
        "prediction": recs["Prediction"].values.tolist()
    }

    #Insere as recomendações do usuário no MongoDB
    db_recomendacoes.insert_one(rec_doc)
```

3.2.3. Inicialização do client do MongoDB



```
puc@puc-bigdata: ~/Desktop/mongodb-linux-x86_64-ubuntu1604-3.6.3/bin
puc@puc-bigdata:~/Desktop/mongodb-linux-x86_64-ubuntu1604-3.6.3/bin$ sudo ./mongo
```

3.2.4. Visualizando recomendações armazenadas no MongoDB para userId 3



```
> use movies;
switched to db movies
> db.knn_recomendacoes.find({userId:3}).pretty();
{
  "_id" : ObjectId("5dcefc9b4d3c5f1068b7f930"),
  "userId" : 3,
  "recs" : [
    [
      "Adventures of Baron Munchausen, The (1988)",
      "Adventure|Comedy|Fantasy"
    ],
    [
      "Disturbing Behavior (1998)",
      "Horror|Thriller"
    ],
    [
      "Hustler White (1996)",
      "Romance"
    ],
    [
      "Whole Wide World, The (1996)",
      "Drama"
    ],
    [
      "Crimson Rivers, The (Rivières pourpres, Les) (2000)",
      "Crime|Drama|Mystery|Thriller"
    ]
  ]
}
```

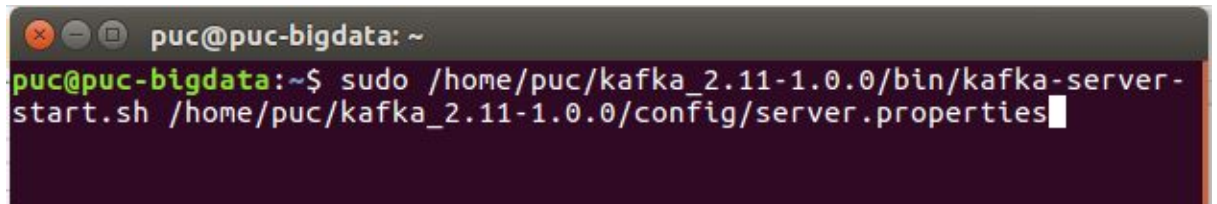
3.3. Criando as estruturas de fila para responder às solicitações de recomendação

Nessa etapa foram criados, utilizando o software Kafka, as estruturas responsáveis por armazenar temporariamente as solicitações e resposta relativas às recomendações, suportando o fluxo contínuo dos dados. No modelo conceitual do Kafka essas estruturas são chamadas de tópicos, funcionando como filas de armazenamento de dados para produtores e consumidores desses dados.

Assim foram criados dois tópicos no Kafka:

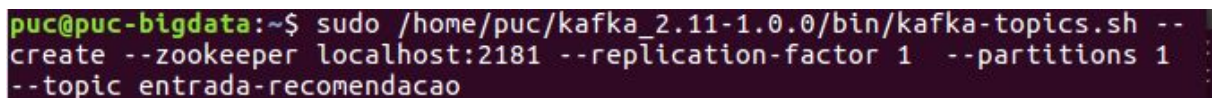
- tópico de entrada para armazenar as solicitações de recomendação
- tópico de saída para armazenar a resposta à essas solicitações.

3.3.1. Inicializando o Kafka

A terminal window with a dark background. The prompt is 'puc@puc-bigdata: ~'. The command entered is 'sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-server-start.sh /home/puc/kafka_2.11-1.0.0/config/server.properties'.

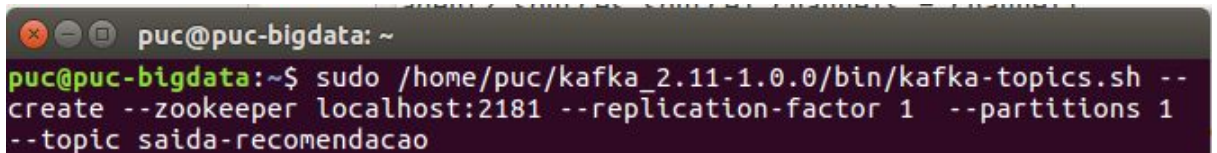
```
puc@puc-bigdata: ~  
puc@puc-bigdata:~$ sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-server-start.sh /home/puc/kafka_2.11-1.0.0/config/server.properties
```

3.3.2. Criando Tópico de Entrada

A terminal window with a dark background. The prompt is 'puc@puc-bigdata:~\$'. The command entered is 'sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic entrada-recomendacao'.

```
puc@puc-bigdata:~$ sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic entrada-recomendacao
```

3.3.3. Criando Tópico de Saída

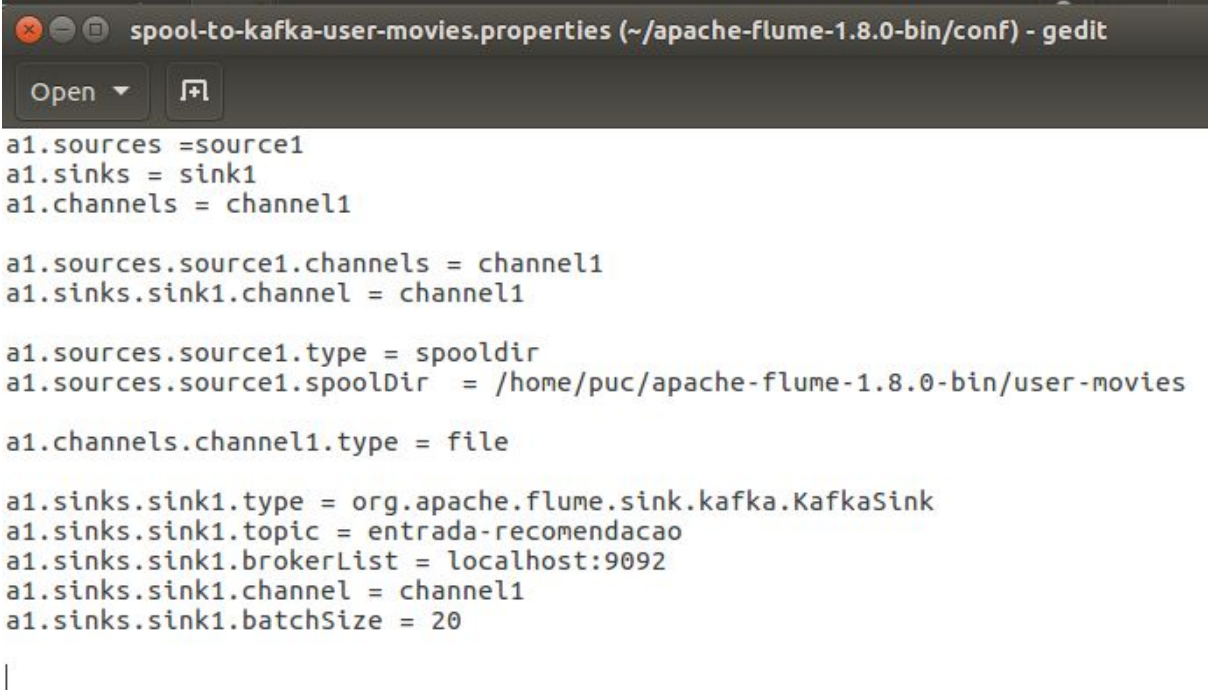
A terminal window with a dark background. The prompt is 'puc@puc-bigdata: ~'. The command entered is 'sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic saida-recomendacao'.

```
puc@puc-bigdata: ~  
puc@puc-bigdata:~$ sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic saida-recomendacao
```

3.4. Coletando e armazenando as solicitações de recomendação

Nessa etapa, utilizando o Flume, foi desenvolvido um agente responsável por coletar as solicitações de recomendação para os usuários e inseri-las no tópico de entrada do Kafka.

3.4.1. Configurando o agente do Flume



```
spool-to-kafka-user-movies.properties (~/apache-flume-1.8.0-bin/conf) - gedit

a1.sources = source1
a1.sinks = sink1
a1.channels = channel1

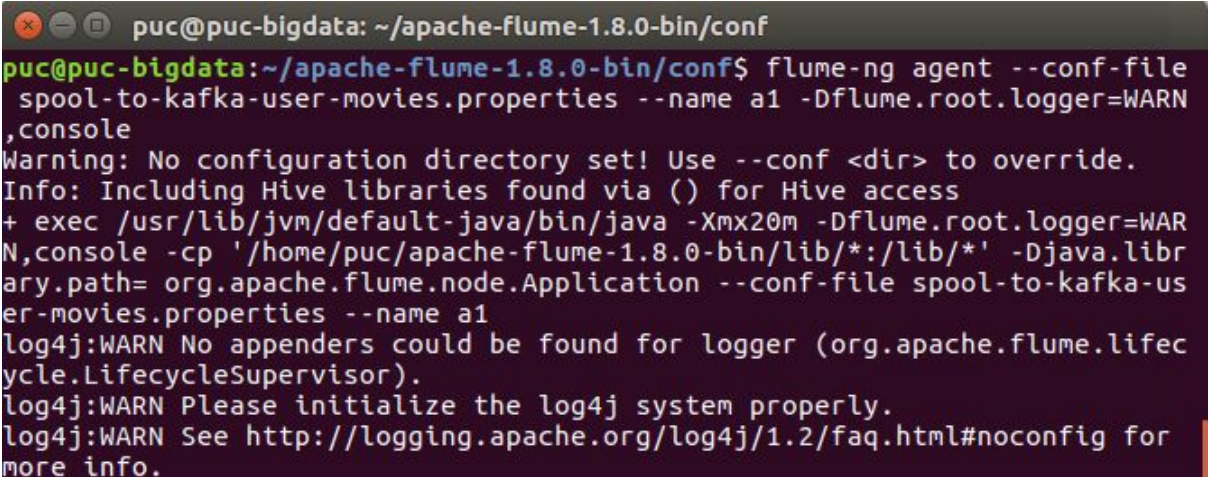
a1.sources.source1.channels = channel1
a1.sinks.sink1.channel = channel1

a1.sources.source1.type = spooldir
a1.sources.source1.spoolDir = /home/puc/apache-flume-1.8.0-bin/user-movies

a1.channels.channel1.type = file

a1.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.sink1.topic = entrada-recomendacao
a1.sinks.sink1.brokerList = localhost:9092
a1.sinks.sink1.channel = channel1
a1.sinks.sink1.batchSize = 20
```

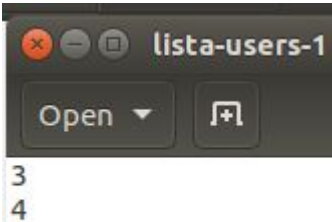
3.4.2. Inicializando o agente do Flume



```
puc@puc-bigdata: ~/apache-flume-1.8.0-bin/conf
puc@puc-bigdata:~/apache-flume-1.8.0-bin/conf$ flume-ng agent --conf-file
spool-to-kafka-user-movies.properties --name a1 -Dflume.root.logger=WARN
,console
Warning: No configuration directory set! Use --conf <dir> to override.
Info: Including Hive libraries found via () for Hive access
+ exec /usr/lib/jvm/default-java/bin/java -Xmx20m -Dflume.root.logger=WAR
N,console -cp '/home/puc/apache-flume-1.8.0-bin/lib/*:/lib/*' -Djava.lib
rary.path= org.apache.flume.node.Application --conf-file spool-to-kafka-us
er-movies.properties --name a1
log4j:WARN No appenders could be found for logger (org.apache.flume.lifec
ycle.LifecycleSupervisor).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for
more info.
```

3.4.3. Produzindo solicitações de recomendação

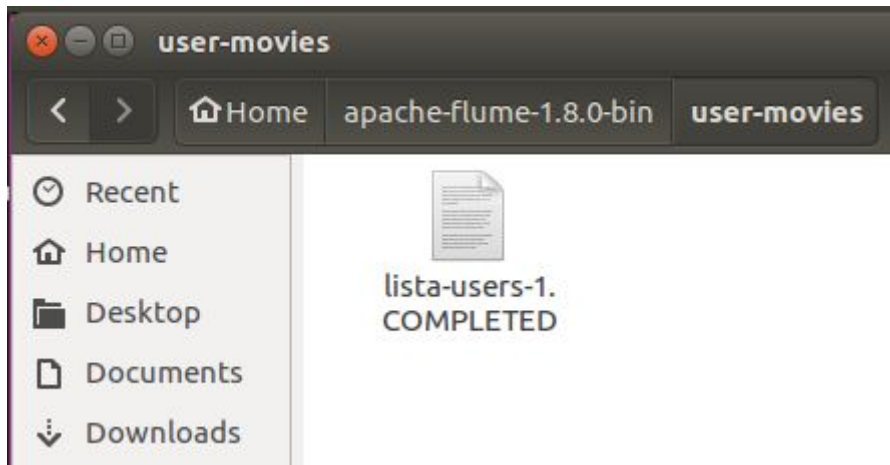
Arquivo criado com ids de usuário para os quais se deseja gerar recomendação



```
lista-users-1

3
4
```

Arquivo marcado pelo Flume como coletado, após ser copiado para a pasta monitorada pelo agente.



3.4.4. Visualizando IDs de usuários armazenados pelo Flume no tópico entrada-recomendacao do Kafka

```
puc@puc-bigdata: ~  
puc@puc-bigdata:~$ sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-console-consumer.sh  
--zookeeper localhost:2181 --topic entrada-recomendacao --from-beginning  
Using the ConsoleConsumer with old consumer is deprecated and will be removed in  
a future major release. Consider using the new consumer by passing [bootstrap-s  
erver] instead of [zookeeper].  
3  
4  
█
```

3.5. Processando Solicitação de Recomendações

Nessa etapa foi desenvolvido um programa em pyspark, utilizando o Spark Streaming, para processar as solicitações de recomendação conforme o seguinte algoritmo:

- a. Consumir o tópico Kafka entrada-recomendacao e coletar os ids de usuários inseridos no tópico

Para cada id lido do tópico Kafka

- b. Buscar a recomendação armazenada no MongoDB com base no id do usuário
- c. Postar a recomendação no tópico Kafka saida-recomendacao

Abaixo a função em python que implementa o algoritmo de processamento. O programa completo Spark Streaming que executa esse algoritmo se encontra no Anexo II deste relatório.

```
from pymongo import MongoClient
from kafka import KafkaProducer
import json

def processa_solicitacao(usuario):

    #Conexão com o banco de recomendações
    client = MongoClient('localhost', 27017)
    db = client.movies.knn_recomendacoes

    #Inicializa producer para o tópico kafka de saída das recomendações
    producer = KafkaProducer(bootstrap_servers='localhost:9092', \
                             value_serializer=lambda v: json.dumps(v).encode('utf-8'))

    for id in usuario:
        doc = db.find_one({'userId': int(id)})
        del doc['_id']
        producer.send('saida-recomendacao', doc)

    client.close()
```

3.5.1. Consumo do tópico Kafka de solicitações de recomendação no Spark Streaming

```
spark = SparkSession \
    .builder \
    .appName("GetRecomendacaoService") \
    .config("spark.jars.packages", "org.apache.spark:spark-streaming-kafka-0-8_2.11:2.2.0") \
    .config("spark.streaming.kafka.maxRatePerPartition", 1000) \
    .config("spark.default.parallelism", "2") \
    .config("spark.driver.memory", "4g") \
    .getOrCreate()

sc = spark.sparkContext
ssc = StreamingContext(sc, 1)

kvs = KafkaUtils.createStream(ssc, '127.0.0.1:2181', "spark-streaming-consumer", {"entrada-recomendacao": 1})
lines = kvs.map(lambda x: x[1])
lines.foreachRDD(lambda rdd: rdd.foreach(processa_solicitacao))
lines.pprint()

ssc.start()
ssc.awaitTermination()
```

era-recomendacao

```
-----
Time: 2019-11-16 15:56:17
3
4
-----
Time: 2019-11-16 15:56:18
-----
```

Na figura acima é possível ver o console de execução da aplicação. Nele é exibido os dois ids lidos do tópico Kafka: entrada-recomendacao.

3.5.2. Visualizando as recomendações inseridas no tópico Kafka saida-recomendacao, após processamento da solicitação

```
puc@puc-bigdata:~$ sudo /home/puc/kafka_2.11-1.0.0/bin/kafka-console-consumer.sh --zooke
eper localhost:2181 --topic saida-recomendacao --from-beginning[sudo] password for puc:
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a futur
e major release. Consider using the new consumer by passing [bootstrap-server] instead o
f [zookeeper].
{"userId": 3, "recs": [{"Adventures of Baron Munchausen, The (1988)", "Adventure|Comedy|
Fantasy"}, {"Disturbing Behavior (1998)", "Horror|Thriller"}, {"Hustler White (1996)", "
Romance"}, {"Whole Wide World, The (1996)", "Drama"}, {"Crimson Rivers, The (Rivi\u00e8r
es pourpres, Les) (2000)", "Crime|Drama|Mystery|Thriller"}, {"River's Edge (1986)", "Cri
me|Drama"}, {"Lumumba (2000)", "Drama"}, {"Ready to Wear (Pret-A-Porter) (1994)", "Comed
y"}, {"Blood and Wine (Blood & Wine) (1996)", "Crime|Drama|Thriller"}, {"Truth About Cat
s & Dogs, The (1996)", "Comedy|Romance"}], "prediction": [5.0, 5.0, 5.0, 5.0, 5.0, 5.0,
5.0, 5.0, 5.0, 5.0]}
{"userId": 4, "recs": [{"Rocketship X-M (1950)", "Sci-Fi"}, {"Treasure Island (1950)", "
Adventure|Children"}, {"Hotel de Love (1996)", "Comedy|Romance"}, {"Beautician and the B
east, The (1997)", "Comedy|Romance"}, {"Little Women (1994)", "Drama"}, {"Kid in King Ar
thur's Court, A (1995)", "Adventure|Children|Comedy|Fantasy|Romance"}, {"Just Cause (199
5)", "Mystery|Thriller"}, {"Comfort and Joy (1984)", "Comedy"}, {"River, The (1984)", "D
rama"}, {"Abbott and Costello Meet the Mummy (1955)", "Comedy|Horror"}], "prediction": [
5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0]}
```

4. Conclusão

A prática em laboratório descrita neste documento possibilitou a aplicação de conhecimentos compartilhados nas disciplinas de Projeto Integrado e Processamento e Análise de Fluxos Contínuos de Dados, auxiliando no desenvolvimento de habilidades necessárias para projetar, desenvolver, testar e operar sistemas escaláveis para coleta, processamento, armazenamento e visualização de dados, seja em fluxo contínuo ou sob demanda, em lotes ou batch. Além disso foi possível experimentar os softwares Flume, Kafka, Spark Streaming, MongoDB, comprovando sua aplicabilidade aos processos e tarefas relacionadas à engenharia e ciência de dados.