Building Data Processing Pipeline for Recommender System Using Scala and Apache Kafka

Febi Agil Ifdillah (13514010)
Informatics/Computer Science
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Ganeca St. 10 Bandung 40132, Indonesia
febi_agil@students.itb.ac.id

***Abstract*—As the data growing in high-volume and high-velocity, businesses trying to compete each other and make themselves profitable by using it. They use it to know more about their customers so they could target the customer better. They learn about customers using Recommender System. With recommender system, businesses could automate the learning processes and provide affordable, personal, and high-quality recommendations. But, recommender system can not do its job without sufficient data. The RAW data itself, can not fed directly into the learning algorithm. So the data needs to processed in data processing pipeline to extract the data, transform it into much more readable format, and finally load the data so the algorithm could use it. By combining some tools and technologies like Scala, Java, MongoDB, Kafka, Spark, and Akka we could build this data processing pipeline with ease.**

Keywords—Big data; recommender system; data pipeline

I. Introduction

Currently, we produce as much data as we did from the dawn of civilization up until 2003 in just within two days[4]. The data mostly comes from user generated content like status in social media, chat history, and search history. All of the data together becomes so immense that is too difficult to process using traditional methods, such as databases and software. Such phenomena called big data.

There are hidden treasures on those data, ready to be found, for businesses that can successfully process it into information and analyze it to gather insight and act based upon what they found. Current technologies supports automation of such processes so that we could catch up with the ever-growing data with high-velocity property like that.

Now, businesses compete each other out by improving operational practices by identifying problems that may exist using Big Data. Another implementation of such technology is finding the pattern of customer behaviour and preferences, so they can better target them.

Every one of us is unique. Yet there are many people around the world similar to us. We have our own preferences upon items, we want to be different. But the fact is many of us liking the same thing, interacting with the same people, even exhibiting the same behaviour. It makes us very predictable.

Businesses around the world using that very fact to be more profitable. They learn about us using Recommender System. Basically, it uses to filter what statistically is most relevant for a particular user. Recommender systems automate some of these strategies with the goal of providing affordable, personal, and high-quality recommendations[1]. It is pervasive, it has impacted or even redefined our lives in many ways.

We could build good recommender system that will produce sensible recommendations by answer the following questions[2]:

1. What kind of learning algorithm should we run on this dataset to make good recommendations?
2. What data representations does this algorithm expect?
3. How will the data be fed into this algorithm overtime?

A typical Recommender system cannot do its job without

sufficient data. It is also very critical to supply the best data possible to make Recommender system effective. That being said, data is important. So, author's focus in this paper is to build a data processing pipeline. All of the code used in this paper is forked from Building a Recommendation Engine with Scala Book's example code.

II. Tools and Dataset

A. Java

Java is a programming language and computing platform first released by Sun Microsystems in 1995. Java provides a system for developing application software and deploying it in a cross-platform computing environment. We will need JDK already installed in order to run Scala.

B. Scala

In this paper, the author used Scala version 2.11.x that can be downloaded here: http://scala-lang.org/download. Scala is an acronym for "Scalable Language". Scala used in many critical systems, as many companies, including Twitter, LinkedIn, or Intel. In order to handle ever-growing data, we need to have a system that can handle the challenges of scalability, so, Scala will be the right language to choose. Moreover, it runs on JVM, so it offers a good balance between productivity and performance.

C. SBT

SBT is an open source build tool for Scala and Java projects, similar to Java's Maven or Ant. It is adopted by the majority of Scala's developer for building and managing a Scala project and its dependencies.

D. MongoDB

MongoDB uses JSON-like documents with schemas and classified as a NoSQL database program. It is free and open source, and document-oriented database program. We use MongoDB for data persistence and querying.

E. Apache Kafka

Apache Software Foundation developed an open-source stream processing platform written in Scala and Java, called Apache Kafka. We use Kafka for high performance persistence queuing.

F. Apache Spark

The uses of Spark in this project is to achieve our goals for high throughput stream processing.

G. Akka

Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM. We used it for message based concurrence to delegate as much as we could.

H. Entree Dataset

Entree dataset is a Restaurant data, that is well formed, and case-based recommender system dataset. It can processed by any text processing tool and has interesting cases and problem to solve. It can be downloaded here: https://archive.ics.uci.edu/ml/datasets/Entree+Chicago+Recommendation+Data

III. Building the Data Processing Pipeline

1. Extract-Transform-Load

We divide the data processing phase into three stages: extract, transform, and load. It is essential that the data we supply to the learning algorithm is clean. That being said, we need to focused the effort in cleaning and organizing data. This kind of pattern in processing data helps us in separating three big concerns of a data mining[2].

At the first stage, we gather data from all over the sources. The data obtained could be from different data-sources. It could be from web-server logs, database server, and any other sources. In our cases, which is restaurant data, it could be anything that is related to restaurant. It might be PDF document or Excel file. We need to extract the data out of these files.

The second stage is transforming data. It is important to note that we should map the data to a machine readable form. Finally, after the data is processed and cleaned we could load it into a data store.

2. Data processing pipeline for Entree

We want to design and build a pipeline that will allow the learning algorithm to keep learning as soon as new data arrives, and will allow us to look up the data on demand. With that said, we should choose tools that is suitable for such goals. We would use the tools we chose from the previous chapter, namely : Akka, MongoDB, Apache Kafka, Apache Sprak, Scala, and Java. The flow of the data would be like the following illustration:

Entree dataset text files -> Akka -> MongoDB -> Apache Kafka -> Apache Spark.

To get much clearer and bigger picture, lets see the following picture:

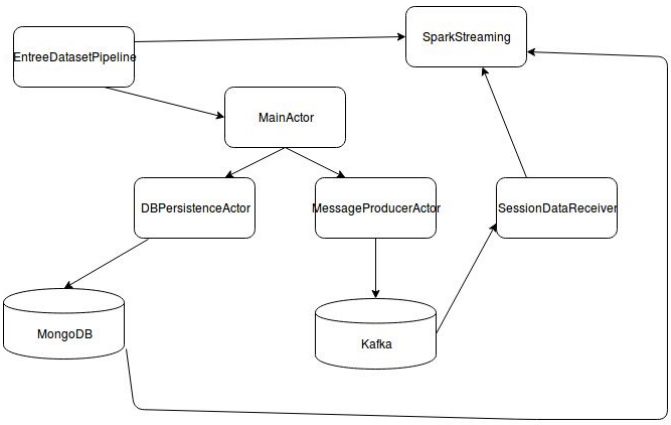

Illustration 1 - Entree Data Processing Pipeline. Adapted from Ansari, Saleem. Building a Recommender Engine with Scala, Packt Publishing, 2016.

First of all, ensure that both Kafka and MongoDB servers are running, then let's see some output now.

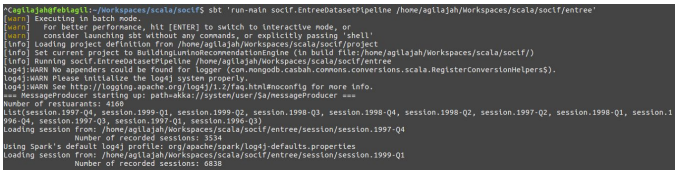

Illustration 2 - Starting Kafka and MongoDB server, and build the Scala Program using SBT

Or we could see it in text version:

```
agilajah@febiagil:~/Workspaces/scala/socif$ sbt 'run-main socif.EntreeDatasetPipeline /home/agilajah/Workspaces/scala/socif/entree'
[warn] Executing in batch mode.
[warn]   For better performance, hit [ENTER] to switch to interactive mode, or
[warn]   consider launching sbt without any commands, or explicitly passing 'shell'
[info] Loading project definition from /home/agilajah/Workspaces/scala/socif/project
[info] Set current project to BuildingLuminoRecommendationEngine (in build file:/home/agilajah/Workspaces/scala/socif/)
[info] Running socif.EntreeDatasetPipeline /home/agilajah/Workspaces/scala/socif/entree
log4j:WARN No appenders could be found for logger (com.mongodb.casbah.commons.conversions.scala.RegisterConversionHelpers$).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
=== MessageProducer starting up: path=akka://system/user/$a/messageProducer ===
Number of restuarants: 4160
List(session.1997-Q4, session.1999-Q1, session.1999-Q2, session.1998-Q3, session.1998-Q4, session.1998-Q2, session.1997-Q2, session.1998-Q1, session.1996-Q4, session.1997-Q3, session.1997-Q1, session.1996-Q3)
Loading session from: /home/agilajah/Workspaces/scala/socif/entree/session/session.1997-Q4
                Number of recorded sessions: 3534
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Loading session from: /home/agilajah/Workspaces/scala/socif/entree/session/session.1999-Q1
                Number of recorded sessions: 6838
17/05/05 21:41:24 INFO Remoting: Starting remoting
17/05/05 21:41:25 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriver@192.168.43.135:44065]
Loading session from: /home/agilajah/Workspaces/scala/socif/entree/session/session.1999-Q2
                Number of recorded sessions: 1299
Loading session from: /home/agilajah/Workspaces/scala/socif/entree/session/session.1998-Q3
                Number of recorded sessions: 4848
Loading session from: /home/agilajah/Workspaces/scala/socif/entree/session/session.1998-Q4
                Number of recorded sessions: 7956
Loading session from: /home/agilajah/Workspaces/scala/socif/entree/session/session.1998-Q2
                Number of recorded sessions: 5502
```

```
Loading session from:
/home/agilajah/Workspaces/scala/socif/entree/session/session.1997-Q2
        Number of recorded sessions: 4002

Next batch...

(SKIPPED OUTPUT)
```

While the pipeline is running, we could also inspect what happen with the MongoDB instance we created. The data should be populated there, too.

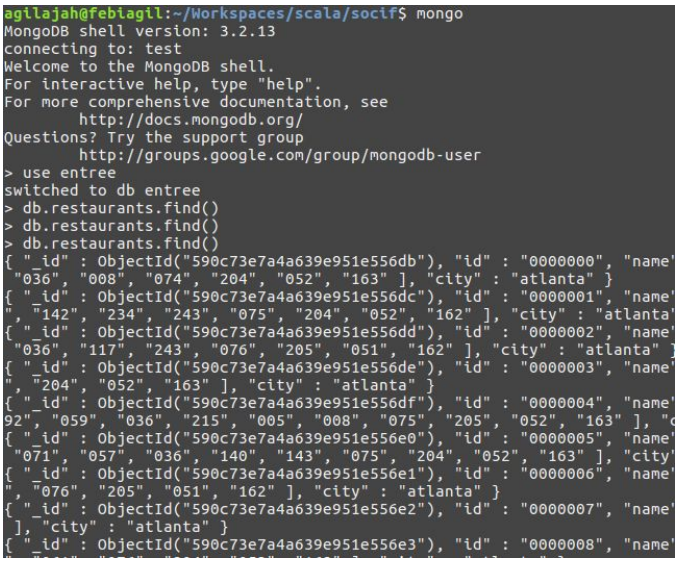

Illustration 3 - Data processed by kafka also populated in MongoDB database instance

IV. Conclusion

Data is very important. It is very critical to supply the best data possible to make Recommender system effective. The phase of data processing itself can be divided into three stages, namely extraction, transformation, and load the data to data storage. This kind of pattern in processing data helps us in separating three big concerns of a data mining. By combining Apache Spark, Apache Kafka, Scala, Akka, and MongoDB, we could compose a complete data processing pipeline with ease.

Acknowledgement

I would like to say Alhamdulillah to Allah for His guidance and blessing so that I was able to finish writing this paper. My million thanks to my parents for everything they have done for me. I would also say thanks to Dr. Ir. Rinaldi Munir, MT., Dr. Eng. Ayu Purwarianti, St., MT., and Dessi Puji Lestari ST,M.Eng.,Ph.D. for their patience in teaching me.

References

[1] Jannach, Dietmar, et al. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[2] Ansari, Saleem. *Building a Recommender Engine with Scala.* Packt Publishing, 2016

[3] Aggarwal, Charu C. *Recommender System: The textbook.* Springer, 2016.

[4] https://techcrunch.com/2010/08/04/schmidt-data/. Every 2 Days We Create As Much Information As We Did Up To 2003. Accessed at Friday, May 5th 2017. 07.00 P.M.

Declaration

I hereby declare that the paper is my own writing, not an adaptation, nor translation from another person's paper, and not a form of plagiarism.

Bandung, May 5th 2017

Febi Agil Ifdillah (13514010)