

Stream Processing: Using Kafka and KSQL for Twitter data

Emil Ramos, Allen Ansari, Yongjun Chu and Chad Madding

Abstract— Real time data processing has become so crucial in today’s analytical world due to the need to make operational adjustments in hours, even minutes now, not months or years in the past for business entities and government alike. Twitter, one of the most popular social media platforms in today’s world, provides large amount of data daily covering a variety of hot topics through tweets. To be able to harness the vast amount of information hidden within those live tweets would be of great interest to the data science community. To carry out the live data streaming processing, a system having the ability to continuously “listen” to new data input is needed. Herein, we demonstrated that by implementing a pipeline composed of Apache Kafka and KSQL, combined with machine learning techniques, we were able to automatically filter, aggregate, and analyze the streaming data. We have found that the general sentiment to the current US trade policy to be mildly positive. These results will be without doubt valuable to decision makers on trade policy.

Index Terms—real time, steaming, processing, Kfaka, KSQL, trade, sentiment analysis

I. INTRODUCTION

With today’s ever-increasing demand for real-time analytics, traditional batch-oriented data processing doesn’t suffice. To keep up with the flow of incoming data, many organizations are moving towards stream processing.

Twitter, being is one of the most often used social platforms in the modern world, has a fire hose of data. It disseminates more than 400 million messages per day, covering information about almost all industries from entertainment to sports, health to business etc. Twitter provides unprecedented access to our lawmakers and to our celebrities, as well as to news as it’s happening. Twitter represents an important data source for the business models of different sized companies. We can use stream processing to automatically filter, aggregate and analyze such data to harness the full value of the data.

Stream processing requires different tools from those used in traditional batch processing architecture. New tools have popped up for use with data streaming recently, such as Amazon’s Kinesis Streams [1] and Google DataFlow [2]. We chose Apache Kafka and KSQL as the pipeline for streaming analysis due to its broad flexibility and impressive capability [3] (**Fig. 1**).

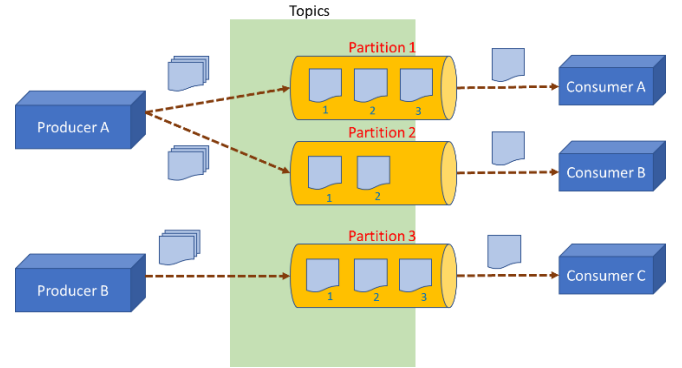


Figure 1. The schematic diagram of Kafka architecture.

Apache Kafka is an open-source stream-processing software platform originally developed by LinkedIn. It was designed to provide a unified, high-throughput, low-latency platform for handling real-time data feeds [4] with a massively scalable pub/sub message storage layer. Kafka can also connect to a variety of external systems (for data import/export).

KSQL, developed at Confluent, is the streaming SQL engine that enables live data processing against Apache Kafka through Kafka Connect [5]. It provides a powerful interactive SQL interface for stream processing on Kafka, effectively eliminating the need to write code in a programming language by end users. KSQL is scalable and fault-tolerant, and it enables users to do data filtering, transformations, aggregations, joins, and windowing and much more.

Sentiment Analysis, also known as Opinion Mining, is a field within Natural Language Processing (NLP) that builds systems that try to identify and extract opinions within text [6,7]. It has greatly helped at transforming unstructured information, such as the texts scattered in e-mails and blogs, into structured data of public opinions about products, services, brands and politics. This data can be very useful for commercial applications such as marketing analysis, public relations, product reviews, product feedback, and customer service.

E. Ramos is with Southern Methodist University, Dallas, TX 75205 (e-mail: author@smu.edu).

A. Ansari is with Southern Methodist University, Dallas, TX 75205 (e-mail: aansari@smu.edu).

Y. Chu is with Southern Methodist University, Dallas, TX 75205 (e-mail: ychu@smu.edu)

C. Madding is with Southern Methodist University, Dallas, TX 75205 (email: cmadding@smu.edu).

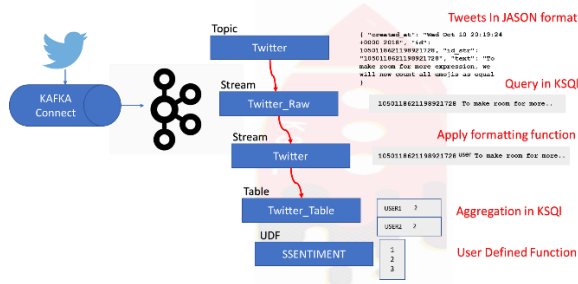


Figure 2. The workflow of Twitter data processing.

The goal for this project is to implement the Kafka/KSQL system on local computers and analyze the real time information from Twitter using this system (**Fig. 2**). We have found that the combination of Kafka and KSQL is robust at processing Twitter live data and implementing desirable machine learning methods to understand underlying information in tweets regarding current US trade policy.

II. MATERIAL AND METHODS

We used open-source software for this project. The Apache Kafka and KSQL package were downloaded through Confluent by choosing “self-managed software” (current version 5.3.0): <https://www.confluent.io/download/>

III. RESULTS

A. Software setup

Apache Kafka is designed to run in a Linux environment. The Kafka package may be downloaded from their main site:

<https://www.confluent.io/download/>

To get Confluent’s Kafka installed we followed the steps described in the following link:

<http://www.nielsberglund.com/2018/07/10/install-confluent-platform-kafka-on-windows/>

There were issues with the Windows 10 Linux installation at the beginning. We decided on a pure Linux environment. Ubuntu 18 was chosen because of its’ ease of installation and graphical environment. We tried installations on both virtual and native environments. The software seemed to run in both settings with no problems (**Fig. 3**).

```
This CLI is intended for development only, not for production
https://docs.confluent.io/current/cli/index.html

Using CONFLUENT_CURRENT: /tmp/confluent.3B9oPdE4
Starting zookeeper
zookeeper is [UP]
Starting kafka
kafka is [UP]
Starting schema-registry
schema-registry is [UP]
Starting kafka-rest
kafka-rest is [UP]
Starting connect
connect is [UP]
Starting ksql-server
ksql-server is [UP]
Starting control-center
control-center is [UP]
```

Figure 3. The successful installation of Kafka package.

KSQL is the querying side of Kafka. It can be run in both command line and a graphical environment and run from a web page on the local computer (**Fig. 4**).

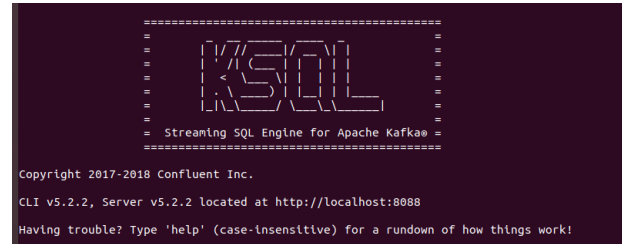


Figure 4. The successful installation of KSQL.

Once Kafka/KSQL package was installed, we followed the steps described in the following link to get the Twitter data coming in.

<https://www.confluent.io/blog/using-ksql-to-analyse-query-and-transform-data-in-kafka>

Specifically, we used Kafka Connect to pull the data from Twitter. The Twitter Connector can be found on GitHub. To install it, simply do the following:

```
# Clone the git repo
git clone https://github.com/jcustenborder/kafka-connect-twitter.git
```

```
# Compile the code
cd kafka-connect-twitter
mvn clean package
```

Once successfully built, unpack the resulting archive to allow Connect to use it:

```
cd target
tar -xvf kafka-connect-twitter-0.2-SNAPSHOT.tar.gz
```

To get Kafka Connect to pick up the connector that we’ve built, we had to modify the configuration file which is located in: etc/schema-registry/connect-avro-distributed.properties. To modify, we simply appended the following line to the end of the file:

```
plugin.path=share/java,/home/rmoff/kafka-connect-twitter/
```

Once the software was implemented we began data exploration. We started by using KSQL to process live data from streams or tables.

B. Start streaming

A typical configuration file to initialize the receiving of live tweets would look like this:

```
filter.keywords=trade, policy
kafka.status.topic=twitter_avro
```

```
twitter.oauth.accessToken=<TWITTER ACCESS TOKEN>
twitter.oauth.accessTokenSecret=<TWITTER ACCESS
TOKEN SECRET>
twitter.oauth.consumerKey=<TWITTER ACCESS
CUSTOMER KEY>
twitter.oauth.consumerSecret=<TWITTER CUSTOMER
SECRET>
```

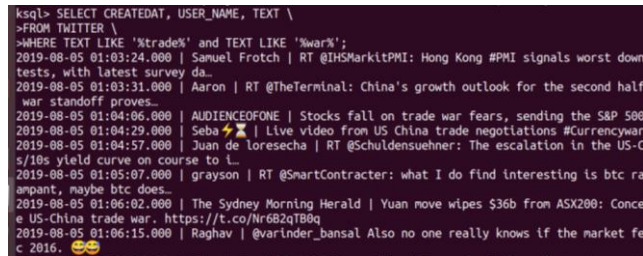
Obviously, each user needs to apply for the Twitter developmental APIs at the first place to be permitted for Twitter connection.

An example of live tweet is shown below:

```
"Crude Oil Price Drop May Speed Up as US-China Trade War
Sizzles" https://t.co/b7vvgJTibm #trading #forex"
"RT @IHSMarkeitPMI: Hong Kong #PMI signals worst
downturn for over a decade in July amid trade wars and local
protests, with latest survey da..."
```

C. SQL-like commands for data abstraction and aggregation

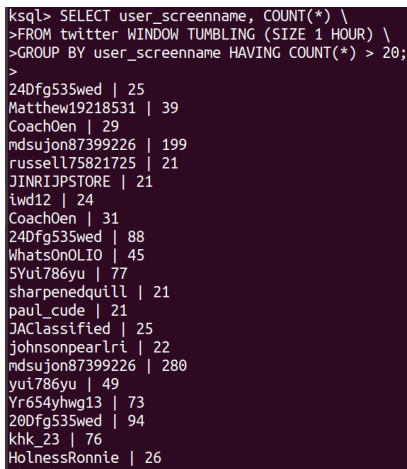
We can use SQL commands to run queries in KSQL. In the first example, we used the “select” command to filter tweets that meet the selection condition –matching the key words (Fig. 5).



```
ksql> SELECT CREATEDAT, USER_NAME, TEXT \
>FROM TWITTER \
>WHERE TEXT LIKE '%trade%' and TEXT LIKE '%war%';
2019-08-05 01:03:24.000 | Samuel Frotch | RT @IHSMarkeitPMI: Hong Kong #PMI signals worst down
tests, with latest survey da...
2019-08-05 01:03:31.000 | Aaron | RT @TheTerminal: China's growth outlook for the second half
war standoff proves.
2019-08-05 01:04:06.000 | AUDIENCEOFONE | Stocks fall on trade war fears, sending the S&P 500
2019-08-05 01:04:29.000 | Seba | Live video from US China trade negotiations #Currencywar
2019-08-05 01:04:57.000 | Juan de loresecha | RT @Schuldensuehner: The escalation in the US-C
s/10s yield curve on course to L...
2019-08-05 01:05:07.000 | grayson | RT @SmartContracter: what I do find interesting is btc ra
ampant, maybe btc does...
2019-08-05 01:06:02.000 | The Sydney Morning Herald | Yuan move wipes $36b from ASX200: Conce
e US-China trade war. https://t.co/Nr68ZqTB0q
2019-08-05 01:06:15.000 | Raghav | @varinder_bansal Also no one really knows if the market fe
c 2016.
```

Figure 5. The output of KSQL with “select” command.

We can also easily carry out the data aggregations. Here we collected the user_screen names that had over 20 tweets within one-hour window counting backward that matched the key words (Fig. 6).



```
ksql> SELECT user_screenname, COUNT(*) \
>FROM twitter WINDOW TUMBLING (SIZE 1 HOUR) \
>GROUP BY user_screenname HAVING COUNT(*) > 20;
>
24Dfg535wed | 25
Matthew19218531 | 39
Coach0en | 29
mdsujon87399226 | 199
russell75821725 | 21
JINRIJPSTORE | 21
iwd12 | 24
Coach0en | 31
24Dfg535wed | 88
WhatsOnOLIO | 45
5Yui786yu | 77
sharpenedquill | 21
paul_cude | 21
JAClassified | 25
johnsonpearlri | 22
mdsujon87399226 | 280
yui786yu | 49
Yr654yhqw13 | 73
20Dfg535wed | 94
khk_23 | 76
HolnessRonnie | 26
```

Figure 6. The output of KSQL with a data aggregation command.

D. Sentiment analysis set up

To carry out the sentiment analysis, we had to first set up a connection between Stanford’s Natural Language API and KSQL. To do that, we first modified the Pom.xml file content in the package by updating the dependencies:

```
<dependencies>
  <!-- KSQL dependency is needed to write your own
UDF -->
  <dependency>
    <groupId>io.confluent.ksql</groupId>
    <artifactId>ksql-udf</artifactId>
    <version>5.2.2</version>
  </dependency>
  <dependency>
    <groupId>edu.stanford.nlp</groupId>
    <artifactId>stanford-corenlp</artifactId>
    <version>3.6.0</version>
  </dependency>
  <dependency>
    <groupId>edu.stanford.nlp</groupId>
    <artifactId>stanford-corenlp</artifactId>
    <version>3.6.0</version>
    <classifier>models</classifier>
  </dependency>
  <dependency>
    <groupId>org.apache.opennlp</groupId>
    <artifactId>opennlp-tools</artifactId>
    <version>1.6.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.opennlp</groupId>
    <artifactId>opennlp-uima</artifactId>
    <version>1.6.0</version>
  </dependency>
  <dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>18.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-collections4</artifactId>
    <version>4.0</version>
  </dependency>
  <dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.4</version>
  </dependency>
</dependencies>
```

Next, we had to build a user-defined function within KSQL by using the following java code:

```
package org.activiti.cloud.connectors.processing.analyzer;
```

```
import edu.stanford.nlp.ling.CoreAnnotations;
import edu.stanford.nlp.neural.rnn.RNNCoreAnnotations;
import edu.stanford.nlp.pipeline.Annotation;
import edu.stanford.nlp.pipeline.StanfordCoreNLP;
import edu.stanford.nlp.sentiment.SentimentCoreAnnotations;
import
edu.stanford.nlp.sentiment.SentimentCoreAnnotations.Sentim
entAnnotatedTree;
import edu.stanford.nlp.trees.Tree;
import edu.stanford.nlp.util.CoreMap;
import io.confluent.ksql.function.udf.Udf;
import io.confluent.ksql.function.udf.UdfDescription;
```

```
import java.util.List;
import java.util.Properties;
import java.util.Arrays;
```

```
@UdfDescription(name = "Ssentiment", description =
"sentiment scoring using Google NL API")
```

```
public class Ssentiment {
    @Udf(description = "sentiment scoring using Stanford NL
API")
    public int Ssentiment(String text) {

        Properties props = new Properties();
        props.setProperty("annotators", "tokenize, ssplit, parse,
sentiment");
        StanfordCoreNLP pipeline = new
StanfordCoreNLP(props);
        int score = 2; // Default as Neutral. 1 = Negative, 2 =
Neutral, 3 = Positive
        String scoreStr;
        Annotation annotation = pipeline.process(text);
        for(CoreMap sentence
:
annotation.get(CoreAnnotations.SentencesAnnotation.class))
        {
            scoreStr
=
sentence.get(SentimentCoreAnnotations.SentimentClass.class)
;
            Tree tree
=
sentence.get(SentimentAnnotatedTree.class);
            score = RNNCoreAnnotations.getPredictedClass(tree);
            System.out.println(scoreStr + "\t" + score + "\t" +
sentence);
        }
        return(score);
    }
}
```

Now, the sentiment test is set up for running within KSQL.

E. Sentiment Analysis results

There are endless ways to analyze enormous amount of information hidden in tweets. We are particularly interested in general attitude within Twitter users toward current US trade policy. Here we used the sentiment analysis, a machine learning technique, to assess how people react to the current trade policy with positive, neutral or negative sentiment scores. The result is

shown in **Figure 7**. A little surprising, the general feeling was found to be slightly positive about US current trade policy since the total percentage of tweets with positive, weakly positive and strongly positive opinions is higher than that with negative, slightly negative and strongly negative opinions. Considering the fact that US is having a tough time negotiating with China, EU and other countries for better deals and tariffs have been used extensively executed by different parties, not having a negative attitude toward US trade policy by majority of people indicates that the impact from having a tough trade stance is still absorbable. On the other hand, it also suggests that the US internal economy may be still in stable condition on a solid ground. Nevertheless, more analysis is clearly needed in the future to solidify or disapprove these claims.

Detailed Report:
13.40% people thought it was positive
22.00% people thought it was weakly positive
2.40% people thought it was strongly positive
1.80% people thought it was negative
10.80% people thought it was weakly negative
1.60% people thought it was strongly negative
47.00% people thought it was neutral

How people are reacting on trade by analyzing 500 Tweets.

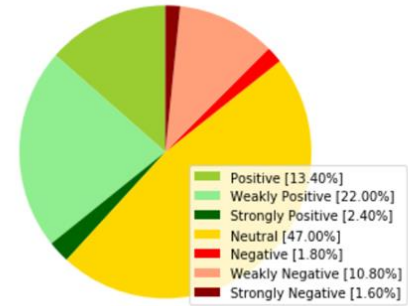


Figure 7. The sentiment analysis result on trade

IV. DISCUSSION

The implementation of Apache Kafka and KSQL turned out to be a not-so-easy process. It requires a Linux operating system. For those users who have no prior experience with UNIX operation system, it may appear to be a challenge at the beginning. Another technically challenging part is the creation of use-defined functions within KSQL and the establishment of a connection with external packages for data analysis and visualization. Users may need to seek support from the developer (Confluent). Nevertheless, once everything is set up properly, the power of this system is not deniable. Using live data from Twitter as an example, we have demonstrated that we can use it to do data filtering, transformations, aggregations, joins, and windowing with streaming data.

The sentiment analysis examining the general opinion about US trade policy yielded interesting results. Although tariffs have caused the price increase for imports from a variety of countries, the impact from that appears to be manageable for majority of people at this point. However, we admit that this sentiment analysis result is still preliminary. More extensive analysis is clearly needed. For example, we may look at the sentiment score distribution geographically to profile which

parts of country are more prone to have positive opinion toward the current trade policy and which parts are more negative.

V. CONCLUSION

Overall, we have demonstrated that the Apache Kafka /KSQL system can be successfully established locally on our personal computers. By analyzing the streaming data from Twitter, we have learned a great deal about people's opinions toward US trade policy. To fully utilize its power, we may connect this system to a cloud storage and multi-cluster parallel-computing facilities to carry out more complex streaming data analysis in the future.

REFERENCES

Basic format for books:

- [1] "Amazon Kinesis", <https://aws.amazon.com/kinesis/>, 2019.
- [2] "Cloud Dataflow", <https://cloud.google.com/dataflow/>, 2019.
- [3] "Streaming SQL for Apache Kafka", <https://www.confluent.io/product/ksql/>, 2019.
- [4] "Apache Kafka, a distributed streaming platform" <https://kafka.apache.org/>, 2019.
- [5] "Introducing KSQL: Streaming SQL for Apache Kafka", <https://www.confluent.io/blog/ksql-streaming-sql-for-apache-kafka/>.
- [6] Stone, Philip J., Dexter C. Dunphy, and Marshall S. Smith. "The general inquirer: A computer approach to content analysis." MIT Press, Cambridge, MA (1966).
- [7] Goldberg, Yoav (2016). A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research, **57**, 345–420.