Assignment 3

Streaming with Kafka and Spark Streaming

Ghaith Farfour
13261843

William Kent
13285337

Geoff Watkins
13284233

Project Overview

Background

Reddit is a website that hosts a collection of forums on specific topics. Each forum is known as a subreddit and allows users to share news and content, or comment on original submissions (also known as posts) on the subreddit topic. Reddit is currently the 19th most popular internet site globally, receiving approximately 1.7 billion unique visitors each month (Tankovska, 2021).

In addition to commenting on subreddit topics, registered users are able to provide feedback on submissions and comments by supplying upvotes for content they like and downvotes, for content they don't. Submissions and comments with a higher number of upvotes will receive more visibility within the subreddit than those with less.

There are a number of API's that make Reddit content readily available. To access content an OAuth client is required. The OAuth client is free.

Project Goal

The project aims to predict the number of comments a Reddit submission in the **r/worldnews** subreddit is likely to receive. The prediction needs to occur in real-time on submissions as they are created.

In addition, Reddit data needs to be visualised in real-time to show keywords in discussion topics.

Process Design

Reddit data is accessible available through API's, although an OAuth client is required. The process described below covers the end-to-end process from the streaming of Reddit submissions and comments through to the visualisation of data and prediction of comments per submission.

Technical Details

The versions of the technical components are:

- Python – 3.8.8
- Confluent – 6.1.1
- Apache Spark – 3.1.1

A number of python packages are used in the process. These packages, and versions, are:

- Configparser – 5.0.1
- Confluent-Kafka – 1.6.1
- Gensim – 4.0.1
- Matplotlib – 3.4.2
- Numpy – 1.20.3
- Pandas – 1.2.2
- Pillow – 8.2.0
- PRAW – 7.2.0
- Pyspark – 3.1.1
- Pyarrow – 2.0.0
- SSEClient – 0.0.27
- Wordcloud – 1.8.1

All of the above are installed through the Dockerfile.

There are 10 docker containers that comprise the services used to stream data from Reddit and process it through the various layers. The containers required are:

- broker
- connect
- control-center
- ksql-datagen
- ksqldb-cli
- ksqldb-server
- jupyter_docker_assignment_3
- rest-proxy
- schema-registry
- zookeeper

High-Level Design

The high-level design of the process can be seen in Figure 1. The process is divided into four sections:

1. Stream: Generates a real-time data stream of Reddit submissions and comments and makes them available as a topic through a Kafka broker.
2. KSQLDB: Filtering and aggregating real-time data in a KSQLDB.
3. Spark: Storing topic messages in parquet and visualising real-time data.
4. Model: Creation of a machine learning model to predict the number of comments per submission and using the model to make real-time predictions on new submissions.

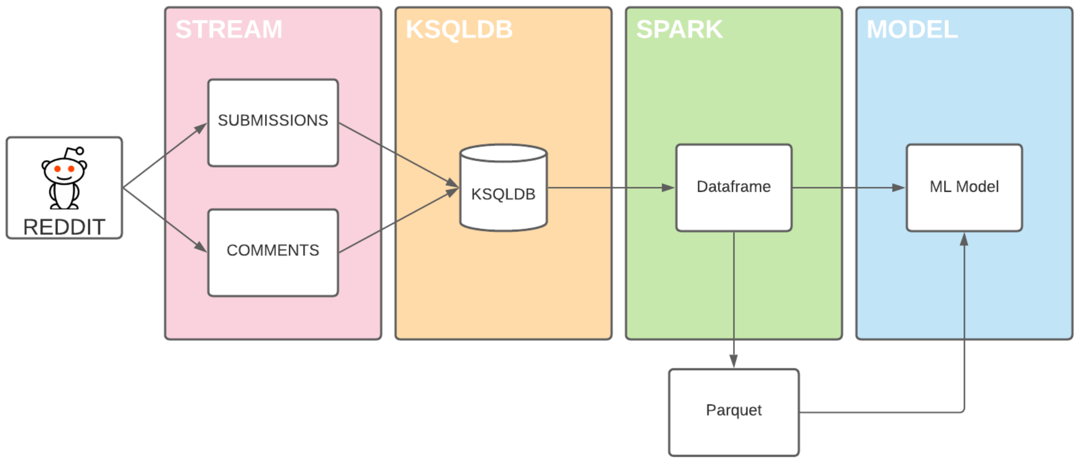

Figure 1: High-level design of process to stream data from Reddit to predict the number of comments per post.

Step Descriptions

Stream

Data is streamed from Reddit using the PRAW (Python Reddit API Wrapper) package. Access to Reddit's API requires an OAuth client. Credentials are obtained by registering an application with Reddit. This data stream has been registered with Reddit through the site https://www.reddit.com/prefs/apps. The OAuth credentials for this application can be found in the configuration file located at *src/configuration/config.cfg*.
The PRAW package is used to stream Reddit data into a JSON formatted message. Kafka is used to map the JSON message to the appropriate topic partition. The stream sends new

messages to either the *submissions* or *comments* topic. The bootstrap server on the Kafka cluster, to which the messages are sent, is registered as **broker:29092**.
The process that initiates the streaming of data is found in the *src/producer.py* file. To start the streams, in a terminal window, execute the command *python src/producer.py* in the working directory on the *jupyter_docker_assignment_3* container. The producer python script uses the **multiprocessing** python package to run both streams in parallel and captures a subset of the data available from Reddit.

Ksqldb

The ksqldb is used to join, filter and aggregate data being streamed by Kafka. In the ksqldb a basic stream is created over both the *submissions* and *comments* topics. In addition, two streams are created that filter data from the topics. These streams are:

- STREAM_BBC_SUBMISSIONS – using the *submissions* topic this stream filters submissions where the domain is equal to **bbc.co.uk**.
- STREAM_SUBMITTER_COMMENTS – using the *comments* topic this stream filters comments where the user making the comment made the original post.

All tables in the Ksqldb are prefixed with "T". A table is created over both the *submissions* and *comments* topics. These are named T_SUBMISSIONS and T_COMMENTS, respectively.
Three tables are created that aggregate data from the various streams. These tables are:

- T_AUTHOR_AVERAGE_SUBMISSION_SCORE – using the *submissions* stream the average submission score is calculated for each submission author. A tumbling window of 5 minutes has been set.
- T_COMMENT_COUNTS – using the *comments* stream the total number of comments per submission is counted. In this statement the LINK_ID is transformed to drop the field prefix and return the raw submission id. No window is applied on this aggregation.
- T_SUBMISSION_AUTHOR_COMMENTS – using the *comments* stream the total number of comments for each submission author is calculated. A tumbling window of 24 hours has been set.

The final table created in the ksqldb joins the T_SUBMISSIONS table and the T_COMMENT_COUNTS table to create the T_SUBMISSION_COMMENTS table. The resulting table has submission attributes and the total number of comments made under that submission.
The code for each ksqldb object can be found in the folder *ksqldb/base_queries*. These individual queries are merged into a single file by running the **create_ksqldb_objects.sh** shell script. The shell script ensures objects are added in the appropriate order to guarantee dependencies are in place when creating an object. The output file is named **create_ksqldb_objects.sql** and can be found in the *ksqldb* folder.
When starting the KSQLDB, the SQL statements in the **create_ksqldb_objects.sql** can be loaded into a KSQLDB editor window and executed. All required objects will be created.

PySpark

Spark is used to process data streams from Confluent/Apache Kafka. Whether it is transforming datasets and topics from Kafka, developing SQL queries on the streamed datasets, or applying machine learning algorithms to the data, Spark Streaming expands the operatives on streaming data to enable real-time analysis and visualisation of information.
As such, the two topics steaming via *running %run src/producer.py* and the messages generated by Kafka/Confluent are used to create the submissions and comments streams in Python.

The messages generated by the *submissions* and *comments* topics are used to generate the submissions and comments streams in Python.
Two dataframes are created at the early stage of streaming the **r/worldnews** data:

- *raw_comments_stream:* Containing Key-Value pairs of all user comments on Reddit posts with the information highlighted in the data description.
- *raw_submission_stream:* Containing Key-Value pairs of all Reddit posts on the **r/worldnews** subreddit with the information highlighted in the data description.

The *raw_comments_stream* is then further processed to create the following:

- *string_stream_df*: This dataframe isolates the values in the key-values pairs and generates its own key-value pairs.
- *json_stream_df*: the json stream highlights the comments (value) as well as their offset and timestamps.
- *comments_stream_df:* A modification of json_stream_df, in which the comments, link titles, and information on the author and event timestamp are included. The "*all_comments_view"* query produces a table with that data.

Based on *comments_stream_df*, a watermarked stream is generated, highlighting the counts of responses the comments received from other Reddit users.
An output option for the *comments_stream_df* dataframe can be through file sink. For this exercise, a file sink has been undertaken, which appends the dataframe into a partitioned parquet output in a separate directory. Although a variety of file types can be chosen for file sink, due to the size of data, was saved in Parquet format.
The *all_comments_view* query can also be used to create a wordcloud visualisation of the most repeated/occurrent words in the stream of comments by utilising *spark.table* to create the dataframe, and utilising the WordCloud and matplotlib packages. An example of the output is shown in Figure 2.

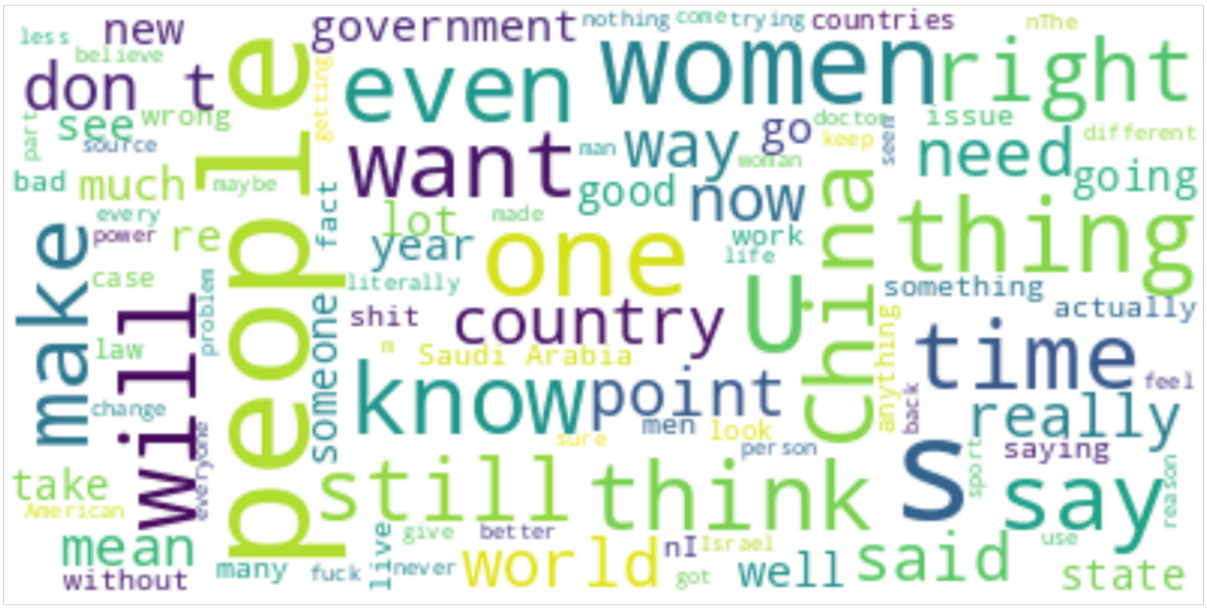

Figure 2: Example of word cloud produced from streamed data.

Data

Data Description

Submissions

- *id* – the unique identifier of the submission.
- *author_fullname* – the unique identifier of a Reddit user, prefixed with “t2” indicating the id type. This field can be absent in the messages from Reddit.
- *title* – the user provided title of the submission that can be up to 300 characters in length.
- *subreddit_name_prefixed* – the subreddit to which the submission was posted (e.g. r/worldnews).
- *name* – the unique identifier of the submission, prefixed with “t3” indicating the id type.
- *upvote_ratio* – the number of upvotes received divided by the number of votes received.
- *ups* – the number of upvotes shown for the submission on site. This may be different to the actual number of upvotes received as Reddit applies obfuscation on upvotes to prevent vote manipulation by bots.
- *score* – the number of submission upvotes minus the number of submission downvotes.
- *author_premium* – identifies if the author of the submission is subscribed, usually through a fee, to premium Reddit content. This field can be absent in the messages from Reddit.
- *created* – the number of seconds since 1st January 1970 to the time the submission was created, in user local time.
- *created_utc* – the number of seconds since 1st January 1970 to the time the submission was created, in co-ordinated universal time (UTC).
- *domain* – the site from which the posted URL comes.
- *url_overridden_by_dest* – a URL link to a website.
- *over_18* – indicates if the submission is only suitable for users over the age of 18.
- *subreddit_id* – the unique identifier of the subreddit in which the submission has been posted.
- *permalink* – the permanent reddit link to the submission, with all its comments.
- *parent_whitelist_status* – unknown.
- *url* – the URL in the submission linking to another webpage.

Comments

- *id* - the unique identifier of the comment.
- *subreddit_id* - the unique identifier of the subreddit in which the submission has been posted.
- *body* – the full text of the comment provided by the user in response to a submission or another users comment.
- *link_id* – the unique identifier of the parent submission, prefixed with “t3” indicating the id type.
- *link_title* – the user provided title of the parent submission.
- *link_author* – the user alias of the Reddit user that created the submission.
- *link_url* - the URL in the submission that links to another webpage.
- *name* – the unique identifier of the comment, prefixed with “t1” indicating the id type.
- *permalink* – the permanent Reddit link to the comment.
- *is_submitter* – indicates if the comment was posted by the user that posted the original submission.
- *created* – the number of seconds since 1st January 1970 to the time the submission was created, in user local time.
- *ups* – the number of upvotes the comment has received.

Modelling

Model overview and use cases

A model has been developed to predict the number of comments every new submission is expected to generate. Rather than predicting a precise number of comments, it predicts one of four categories that bucket the expected number of comments into: none, very few, moderate or many. The model can help identify which type of news submissions attract more attention, thereby assisting at least two potential use cases. It
could allow publishers to publish more attention-grabbing articles or indicate which articles a user can re-post or comment on if they want to gain more attention themselves.
A random forest multi-class classification is used, as this model is regularly shown to provide good fit and is efficient to train in Pyspark as training can be parallelised. However before reaching the machine learning model stage, several processing steps are required on the submissions data. The hypothesis is that the number of comments can be predicted by four key feature types.

Topic modelling

Firstly, and most significantly for processing, the topic of the submission is required. To automatically determine the topic of a given submission it is necessary to apply a natural language processing (NLP) approach. Latent Dirichlet Allocation (LDA), an unsupervised clustering algorithm, is used to generate the terms which can be best combined to indicate a broad topic from the title of the submission. LDA has been widely used in topic modelling over recent years. The number of topics which the algorithm constructs is a user-chosen variable. For the pilot version of the model a limit of 20 topics was used. This can be adjusted in future work to investigate an optimal number of topics for best prediction.
Before performing the actual LDA, submission titles are processed to remove punctuation and numbers, and remove short repetitive words via **stopwords**. The resulting title is split up into individual words (or tokenised). Many text-processing approaches then perform a further step to lemmatise each word, stripping them back to their root word. However, recent research has found that this step does not appear to improve topic modelling and may, in fact, worsen the clustering algorithm (Schofield, et al, 2017)**.** The tokenised words are then quantified into vectors based on the number and choice of words in a title compared to the whole vocabulary made up of all words in the training dataset.
The trained LDA model is then applied to each title in the training dataset to generate the probability of that title belonging to each of the potential 20 topics, resulting in a probability distribution vector of length 20. This is the first feature variable that goes into the random forest model used to predict the future number of comments.

Submission-specific features

Three feature variables are extracted directly from each submission for use in the predictive model. These are:

- hour – the time of day at which the submission is posted,
- day – the day of the week, and
- domain – the domain of the submitter. Each submission in the **worldnews** subreddit connects to an article from reputable news source such as the BBC, Reuters and Associated Press (AP).

These variables were chosen as they are likely to influence the attention a post receives. The majority of commenters are US-based, so the hour of the day is expected to be important for gaining attention. User attention is also likely to vary based on whether the submission is published on a weekday or over the weekend, reflecting active online time. The domain is also likely to impact popular sources such as the BBC having a wider visibility and possibly some sources that are inherently more controversial attracting greater comments per user outcomes.

Training the random forest model

An attempted was made to train the model using the submissions and comments topic data generated by the Kafka producers. However, it became evident that the volume of submissions generated in the **worldnews** subreddit was not enough to provide an adequate dataset size, either for topic modelling or machine learning. Accordingly, the saved parquet files from the Pyspark queries were not used. Instead, a separate historical data extract script was written to access historical Reddit data stored in the Pushshift storage and analytics project. The code used to access this historical data can be found in the *additional_files/notebooks* folder and is named **Extract Reddit History**. This process generated approximately 127,000 submissions, from January to May 2021, along with an aggregated count of comments linked to each submission. These csv files were used to train and test the machine learning model.
Training on the 127,000 submissions became memory intensive when considering that the topic modelling required (sparse) term vectors covering 68,000 words for the 127,000 submissions. The memory issue was solved by saving the tokenised word data to parquet before running the LDA model. Similarly, the features dataset for the random forest model is saved to parquet before applying the random forest machine learning model.
The resultant word count vectorising model, LDA model and random forest pipeline model can be run using the **Part_4_ML_Model_With_Spark** jupyter notebook located in the home directory. This code can be used to refresh the model as new data becomes available.

Prediction Outcome

The generated random forest model predicts the expected volume of comments broken into four categories: none, very few, moderate, or many.

Prediction on streamed submissions

The prediction of the number of comments a submission will receive is executed using the **Part_4_Stream_Predictions_with_Spark** jupyter notebook. The notebook creates a Spark session named *stream-predictions* and subscribes to the submissions topic running on the Kafka broker. A Spark dataframe is generated from the topic and cleaned and transformed using the methods described in the creation of the original model, namely a word count vectorising model and an LDA model, along with other minor transformations used to facilitate prediction.
The transformed, streamed Spark dataframe is then passed to the pre-calculated random forest machine learning model to predict comment volume for newly arrived submissions. The output from the model can be viewed using Spark SQL or through a topic created on the Kafka broker specifically for the prediction results. The topic with the predictions is called *comment_predictions* and is located on the same server as the *submissions* and *comments* topics. The *comment_predictions* topic contains the original submission unique identifier and the comment volume prediction. The *comment_predictions* topic will run until Spark session is closed.

Getting Started

To start the process in production:

1. Open a terminal window and navigate to the base folder for the assignment.
2. Execute *docker-compose up -d*, ten containers should build and start.
3. To connect to the jupyter container:
    a. Run *docker ps*,

   b. Identify the **jupyter_docker_assignment_3** container and copy the container id,
   c. Run *docker logs <container id> --tail 10* and find the URL to the container. It should look something like http://127.0.0.1:8888/lab?token=<hexidecimal token id>.
4. In the jupyter container open a terminal window and, in the working directory, execute *python src/producer.py*. This will start both the *submissions* and *comments* producers.
5. To create KSQLDB objects:
   a. Copy SQL statements in the *create_ksqldb_objects.sql* file,
   b. Open the confluent control center using the URL http://localhost:9021/clusters,
   c. Navigate to the KSQLDB editor, paste SQL statements and execute.
6. In the jupyter container open the **Spark-Comments** notebook and execute code. This code notebook will create:
   a. Required Spark streaming dataframes,
   b. A window stream with a watermark,
   c. Execute a Spark query,
   d. Export streamed comments to the **path/to/destination/dir** parquet file,
   e. Generate a word cloud with most common words used in comments to highlight what Reddit users are talking about in the **worldnews** subreddit.
7. To generate random forest machine learning model on historical data, execute code in the **Part_4_ML_Model_With_Spark** jupyter notebook. Model is saved to **pipeline_model** folder.
8. To apply the random forest model to predict the number of comments for new submissions, execute the code in the **Part_4_Stream_Predictions_with_Spark** jupyter notebook.
9. To stop the producers from streaming *submissions* and *comments* from Reddit press Ctrl-C in the terminal window where the producers were initiated.

Source Control

All source code saved to https://github.com/will-kent/StreamingWithKafka. The repository can be cloned to review change history, or to make amendments.

Reference

Arora, L. 2019 *How to use a Machine Learning Model to Make Predictions on Streaming Data using Pyspark*, viewed 6 June 2021, <https://www.analyticsvidhya.com/blog/2019/12/streaming-data-pyspark-machine-learning-model/>

Kelechava, M. 2019 *Using LDA Topic Models as a Classification Model Input*, viewed 6 June 2021, <https://towardsdatascience.com/unsupervised-nlp-topic-models-as-a-supervised-learning-input-cf8ee9e5cf28>

Schofield, A., Magnusson, M., Thompson, L. and Mimno, D., 2017. Understanding text pre-processing for latent Dirichlet allocation. In *Proceedings of the 15th conference of the European chapter of the Association for Computational Linguistics* (Vol. 2, pp. 432-436).

Silveri, S. 2020 *Distributed Topic Modelling using Spark NLP and Spark MLLib (LDA)*, viewed 3 June 2021, <https://medium.com/analytics-vidhya/distributed-topic-modelling-using-spark-nlp-and-spark-mllib-lda-6db3f06a4da3>

Tankovska, H. 2021 *Total global visitor traffic to Reddit.com 2021*, Statista, viewed 6 June 2021, <https://www.statista.com/statistics/443332/reddit-monthly-visitors/>

Teichmann, J. 2020 *How to embed a Spark ML Model as a Kafka Real-Time Streaming Application for Production Deployment*, viewed 3 June 2021, <https://towardsdatascience.com/how-to-embed-a-spark-ml-model-as-a-kafka-real-time-streaming-application-for-production-deployment-933aecb79f3f>