Data Intensive Computing Final Project

Niloofar Esmaeildoost

October 25, 2020

1 Project Description

The aim of the project is to perform a system for real-time news stream classification. This project consists of 3 parts: (i) gathering real-time news articles using stream tool, (ii) build a pipline to create the model for classification, and (iii) apply the model into the streaming news and categorize them.

2 Implementation

2.1 Produce Stream

To produce the stream, we used the stream tool provided by Guardian API. To use this API, we need to register as a developer for an API key through the link in Guardian platform [1]. My registered key that can be use to run the code is provided in section 2. In order to train the model with offline data, we need to save some data on our machine as a txt file and use it as an input to train the model. The script `save_stream.py` generates the Kafka streaming data from API and saves it into a txt file called `train_data.txt`. Following the API documentation and exploring the data, we can see how the responses from API look like. Basically,

inside the json response, there are several parameters that we can search for. In figure 1, a screenshot of the response has been shown.

Figure 1: A screenshot of Guardian API's response.

Note that the `show-fields:all` key-pair will return all available metadata, including the full text for the articles, therefore, the "fields" is truncated in the

screenshot. We can read the headlines and article texts under the "response", "results", key number, "fields", "headlines" and "bodytext". Each news article have on of the 32 categories, such as Australia news, US news, Football, Sport, World news, and so on. This script can capture a category for each news article and assign a label index for that category. After saving the data into txt files, we can use them for training model. To use this script for stream classification later on, we need to remove the save command section, otherwise, we might run out of memory on our virtual machine (however, this happened after installing Python packages, elasticsearch and Kibana but just to avoid and make sure that won't happen again, it's better to remove that part of the code in the next step).

2.2 Pipeline model

Once we have the offline data, we can create a pipeline model using Tokenizer, Stopword remover, Labelizer, TF-IDF vectorizer and a classifier. Spark has provided a comprehensive documentation about how to make a pipeline model and how it works, which can be found in reference [2]. In `pretrain.py`, we map the input file into pairs of (label, sentence) (first and second column that is used later in DataFrame in Figure 2.) and then we use Tokenizer to break the sentence into words (third column in Figure 2). Stop words are words which should be excluded from the input, typically because the words appear frequently and don't carry as much meaning, words like, I, a, to,... StopWordsRemover takes as input a sequence of strings (e.g. the output of a Tokenizer) and drops all the stop words from the input sequences (4th column in Figure 2). The list of stopwords is specified by the stopWords parameter. Default stop words are accessible by calling StopWordsRemover(). For each sentence (bag of words), we use HashingTF to hash the sentence into a feature vector (5th column in Figure 2). We use IDF to rescale the feature vectors [3]. In figure 2, we can see how the data frame look like after

doing the HashingTF. After that, we used NaiveBaye to do the prediction which is one of the most widely used algorithm in machine learning [4]. After running this section, we get what is shown in Figure 3.

```
+-----+--------------------+--------------------+--------------------+--------------------+
|label|            sentence|               words|            filtered|         rawFeatures|
+-----+--------------------+--------------------+--------------------+--------------------+
|    0|Northern Territor...|[northern, territ...|[northern, territ...|(300,[0,2,4,11,12...|
|    0|Tennant Creek: to...|[tennant, creek:,...|[tennant, creek:,...|(300,[0,1,2,3,4,5...|
|    1|Love, loss and a ...|[love,, loss, and...|[love,, loss, lot...|(300,[0,3,4,5,6,7...|
|    2|Observer killer s...|[observer, killer...|[observer, killer...|(300,[13,15,20,24...|
|    2|Observer sudokuCl...|[observer, sudoku...|[observer, sudoku...|(300,[5,15,23,24,...|
|    3|Azed crossword 2,...|[azed, crossword,...|[azed, crossword,...|(300,[5,16,18,30,...|
|    3|Speedy crossword ...|[speedy, crosswor...|[speedy, crosswor...|(300,[41,75,201],...|
|    3|Everyman crosswor...|[everyman, crossw...|[everyman, crossw...|(300,[18,75,113],...|
|    4|Synchronised bril...|[synchronised, br...|[synchronised, br...|(300,[0,1,2,3,4,6...|
|    4|Cristiano Ronaldo...|[cristiano, ronal...|[cristiano, ronal...|(300,[0,1,3,13,18...|
|    5|Protesters march ...|[protesters, marc...|[protesters, marc...|(300,[0,1,2,3,4,5...|
|    6|Rudy Giuliani cal...|[rudy, giuliani, ...|[rudy, giuliani, ...|(300,[0,1,3,4,6,1...|
|    7|Poll of Unite uni...|[poll, of, unite,...|[poll, unite, uni...|(300,[0,1,3,4,6,7...|
|    8|Kyle Edmund not r...|[kyle, edmund, no...|[kyle, edmund, re...|(300,[0,2,3,4,5,6...|
|    9|We must stop abus...|[we, must, stop, ...|[must, stop, abus...|(300,[0,1,3,4,6,8...|
|    9|Outlaw prostituti...|[outlaw, prostitu...|[outlaw, prostitu...|(300,[0,2,3,5,6,8...|
|    7|What's trickier t...|[what's, trickier...|[what's, trickier...|(300,[0,1,2,3,4,5...|
|    4|Refreshed or disr...|[refreshed, or, d...|[refreshed, disru...|(300,[0,1,2,3,4,5...|
|    4|Luka Modric: the ...|[luka, modric:, t...|[luka, modric:, u...|(300,[0,1,2,4,5,6...|
|    4|Dele Alli fit to ...|[dele, alli, fit,...|[dele, alli, fit,...|(300,[0,3,4,10,11...|
+-----+--------------------+--------------------+--------------------+--------------------+
only showing top 20 rows
```

Figure 2: Spark DataFrame aftering applying Tokenizer, StopWordRemover, HashingTF and IDF.

```
+----------+-----+
|prediction|label|
+----------+-----+
|       0.0|    0|
|       1.0|    1|
|       0.0|    2|
|       3.0|    3|
|       1.0|    1|
|       4.0|    4|
|       5.0|    5|
|       6.0|    6|
|       2.0|    7|
|       8.0|    8|
|       4.0|    9|
|      10.0|   10|
|      11.0|   11|
|      17.0|    4|
|       4.0|   11|
|       4.0|   11|
|       4.0|    4|
|       0.0|    0|
|       5.0|    5|
```

Figure 3: Output of the pretrain.py script. A comparison between the prediction and the actual label can been seen for the offline data.

2.3 Stream Consumer and Visualization

`classifier.py` fetches the news from Kafka server in intervals of 5 seconds and uses the trained model that does the classification process and displays the performance results. It also connects to the Elasticsearch and Kibana and inside Kibana, we see the logs that are being published as the messages are runnung inside the consumer window. Figure 4 shows the result in Kibana.

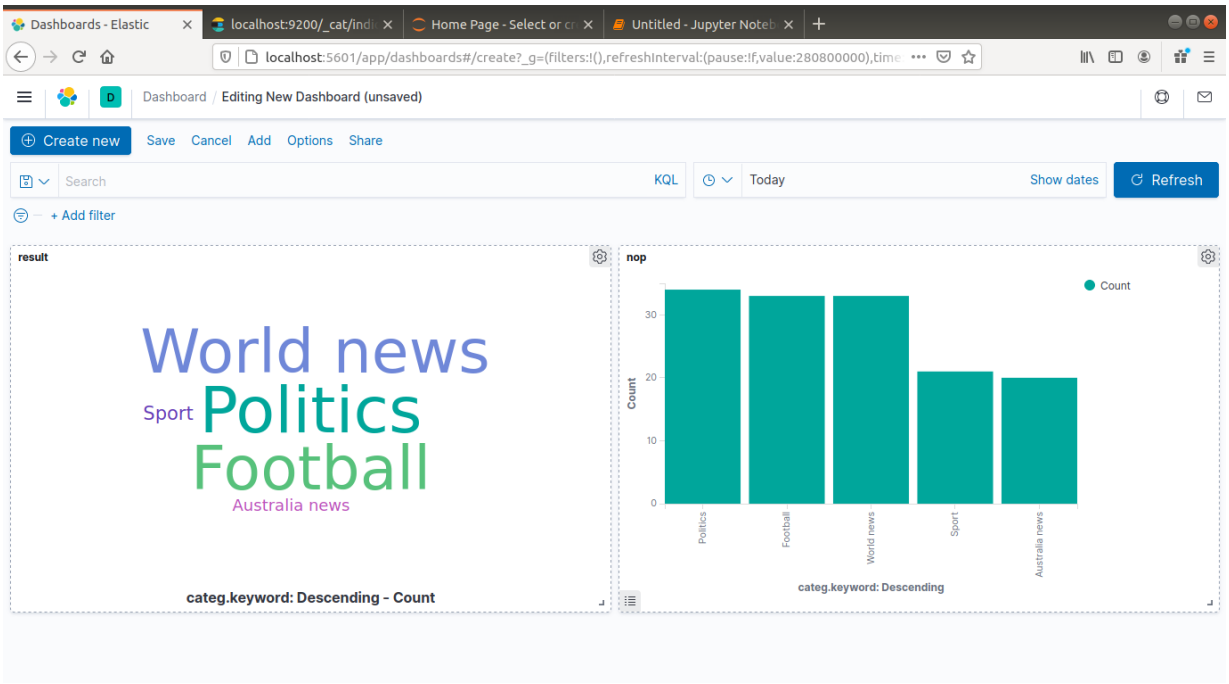

Figure 4: Showing the results in Kibana window.

3 How to Run

Requirements for the project are:

- Kafka
- Zookeeper

- python3, PySpark
- Elasticsearch and Kibana

To do the pretraining for the model:

1. Open Zookeeper Using the command below in one line:

`$KAFKA_HOME/bin/zookeeper-server-start.sh $KAFKA_HOME/config/zookeeper.properties`

2. Open Kafka in a new terminal using the following command:

`$KAFKA_HOME/bin/kafka-server-start.sh $KAFKA_HOME/config/server.properties`

3. Then create a topic called guardian2 using the command below:

`$KAFKA_HOME/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic guardian2`

4. Open a consumer window to check the messages:

`$KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic guardian2`

5. Run producer using the command:

python3 `save_stream.py` [API key] start-date end-date

My registered API key that can be used for running the code is:

457809d1-c088-46b4-9eb8-d5d4aa1d7aab

6. After saving data, we can train the model by typing python3 pretrain.py.

7. Then we run the producer again.

8. Then we start the classification, using the command below:

`spark-submit --packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.0.1 classifier.py localhost:9092 guardian2`

9. We can now check Kibana and plot the coming result.

4 References

[1] [Online]. Available: https://open-platform.theguardian.com/access/.

[2] [Online]. Available: https://spark.apache.org/docs/latest/ml-pipeline.html#pipeline.

[3] [Online]. Available: https://spark.apache.org/docs/latest/ml-features.

[4] [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html.