





Sentiment Analysis using PySpark

The slide features decorative elements consisting of two horizontal teal bars at the top and bottom, and two short horizontal olive-green bars positioned symmetrically below the main title.

Goals

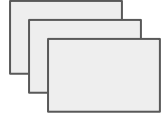
- Building a real time pipeline to analyze and classify tweets into neutral, positive or negative sentiment using PySpark 
- Data cleaning and tokenization 
- Tweet sentiment classification model development 
- Analyzing performance of streaming pipeline in terms of latency 

Division of Work

- Data Preprocessing: Amizhthan
- Model development: Yueyang
- Streaming pipeline: Lincy
- Tests and model evaluation: Anirudh

Design Description

2M + tweets, csv files:



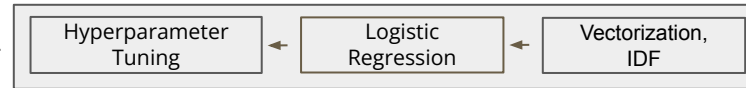
Data PreProcessing with Pyspark



Tweets taken with varying batch sizes

Saved Model

Model training with Pyspark ML



Classify each tweet into 1 of 3 categories

Positive

Neutral

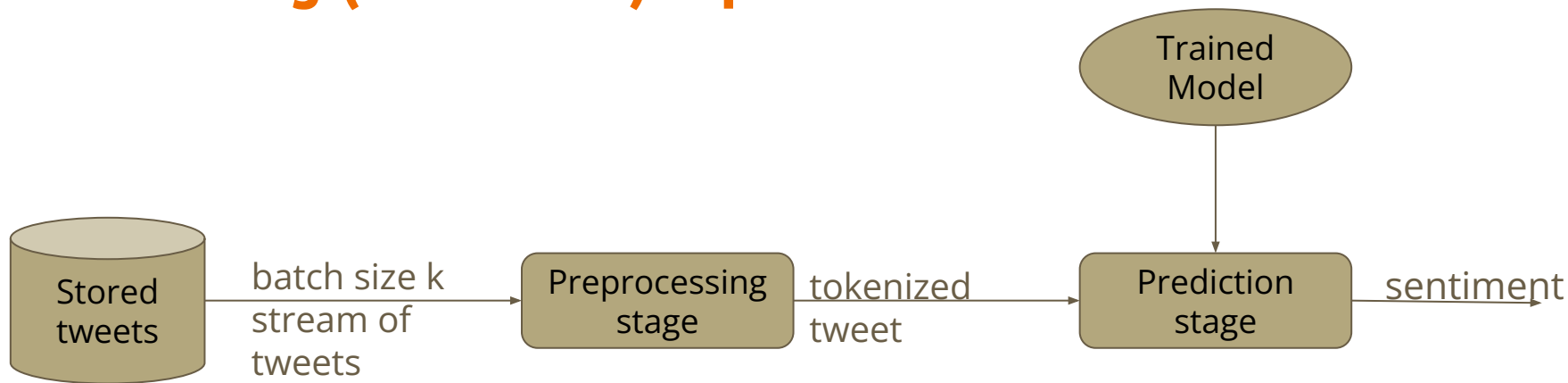
Negative

Performance metric: latency
Model evaluation: accuracy

Model Design

- Countvectorizer - words as vectors
 - depends on tweet length
- IDF - inverse document frequency
 - more common words, less weights
- Multiclass Logistic Regression - softmax, gradient, matrix multiplication
 - weights matrix size depends on total vocab size
- Hyperparameter tuning with cross validation
 - regularization parameter strength (0.01 - 0.1) and elastic net (0, 0.5, 1)
- Model evaluation with accuracy and precision
 - Both around 75 - 85%, but generally more accurate with less regularization

Streaming (Real Time) Pipeline



- Using Spark Structured Stream
- Stored data in different csv files passed to Spark stream, batch size specified using *maxFilesPerTrigger* parameter

Possible Improvements

- Test using Twitter API, for the purpose of project we created streaming data by sampling few tweets and dividing into 20 files and pass them into Spark's readStream()
- Varying model's dictionary size, selecting only top X words to tokenize, may improve speed at the cost of accuracy
- Evaluating tweet length (corpus size) in addition to batch size

Tests

We measure the latency of the program. The methodology adopted is as follows:

- The streaming data is first split into 20 csv files with 2000 data points in each file. The data is then streamed n number of files at a time where $n = 1, 2, 4, 5, 10, 20$. The general expectation is that the latency increases as we stream more files together. The result is then plotted as a graph.
- Another test case adopted is the streaming data is split into 20 csv files and is streamed 1 file at a time, but the number of data points changes. There are n number of data points in each file where $n = 1, 10, 50, 100, 200, 500$. Here the expectation is as the number of data points in a file increases, the latency should increase.

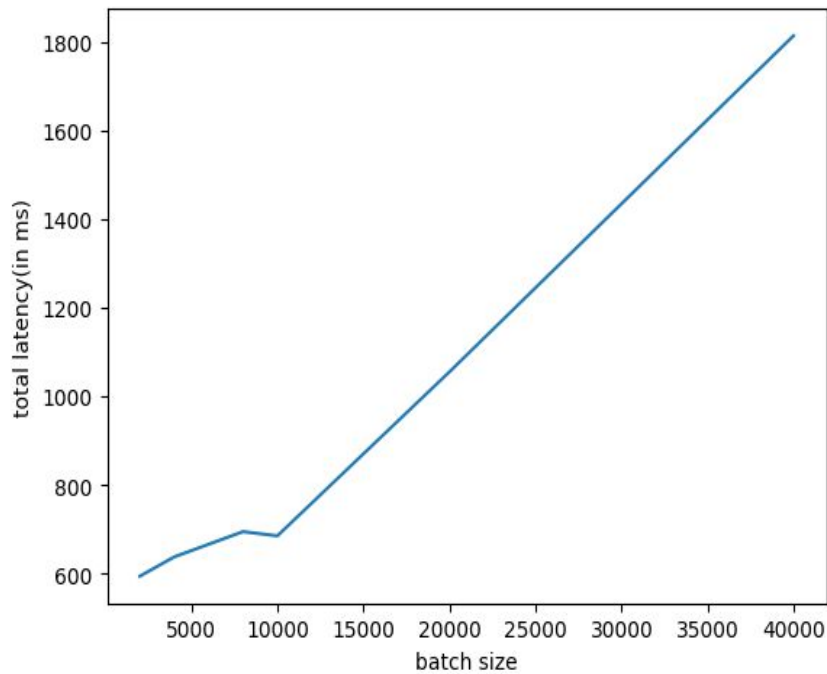
Experimental Results

23/11/30 10:58:29 WARN DAGScheduler: Broadcasting large task binary with size 1565.1 KiB

words	features	prediction	preprocessing_latency_ms	prediction_latency_ms	total_latency_ms
[belarus, stated,...]	(30000,[20,94,367...]	0.0	415.3	144.305	559.605
[onion, headline,...]	(30000,[92,1572,1...]	0.0	415.3	144.305	559.605
[standwithukraine]	(30000,[13],[2.81...]	0.0	415.3	144.305	559.605
[jesus, hippie]	(30000,[2025,1126...]	0.0	415.3	144.305	559.605
[eccl, nndoj, dem...]	(30000,[277,698,9...]	0.0	415.3	144.305	559.605
[eccl, nndoj, dem...]	(30000,[277,698,9...]	0.0	415.3	144.305	559.605
[mr, johnson, said]	(30000,[63,490,70...]	0.0	415.3	144.305	559.605
[pm, say, statement]	(30000,[16,590,59...]	0.0	415.3	144.305	559.605
[magazine, style,...]	(30000,[62,256,38...]	0.0	415.3	144.305	559.605
[magazine, style,...]	(30000,[62,256,38...]	0.0	415.3	144.305	559.605
[possible, narrat...]	(30000,[1,222,858...]	0.0	415.3	144.305	559.605
[travel, guide, j...]	(30000,[34,1326,1...]	0.0	415.3	144.305	559.605
[example, arm, co...]	(30000,[273,300,1...]	0.0	415.3	144.305	559.605
[as]	(30000,[1177],[6...]	0.0	415.3	144.305	559.605
[per, stmt]	(30000,[975],[6.2...]	0.0	415.3	144.305	559.605
[said, m, suprun]	(30000,[63,2772],...]	0.0	415.3	144.305	559.605
[fsb, said]	(30000,[63,4184],...]	0.0	415.3	144.305	559.605
[official, said]	(30000,[63,181],[...]	0.0	415.3	144.305	559.605
[woow, nato, time...]	(30000,[1,3,43,87...]	0.0	415.3	144.305	559.605
[mexico]	(30000,[510],[5.5...]	0.0	415.3	144.305	559.605

only showing top 20 rows

Experimental Results

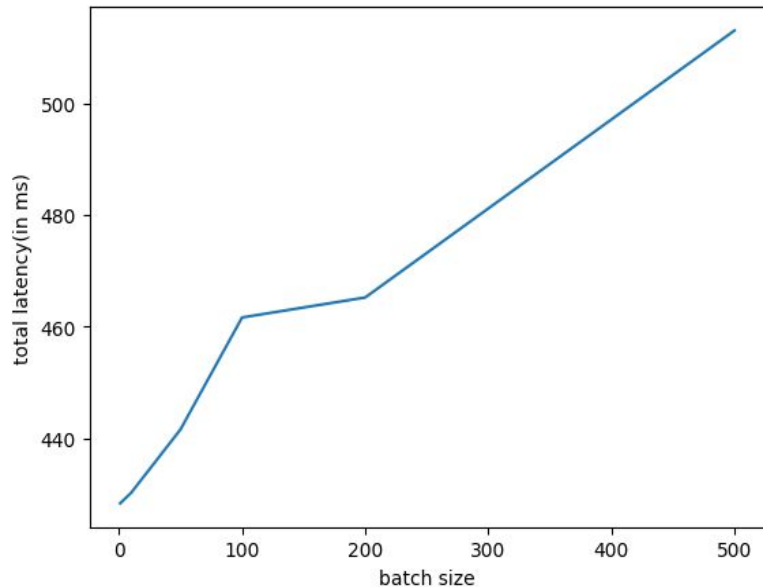


Test 1: number of files streamed changes while the number of data points in a file remains constant(2000)

- As can be seen from the latency plot, the program performs as expected with a general increase in latency as the batch size increases, i.e as we stream more files together($n=1,2,4,5,10,20$) the latency increases. Here, the batch size is calculated as follows:
- Here, the batch size is calculated as follows:

Batch Size = number of files * number of data points per file.

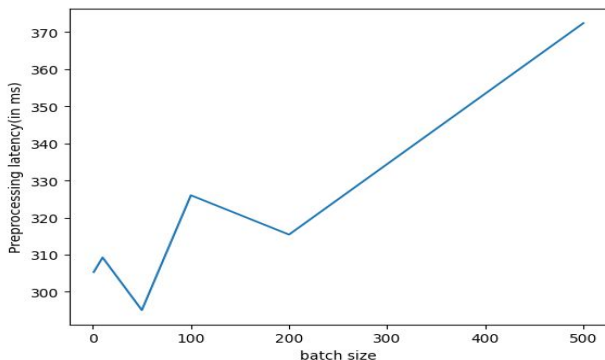
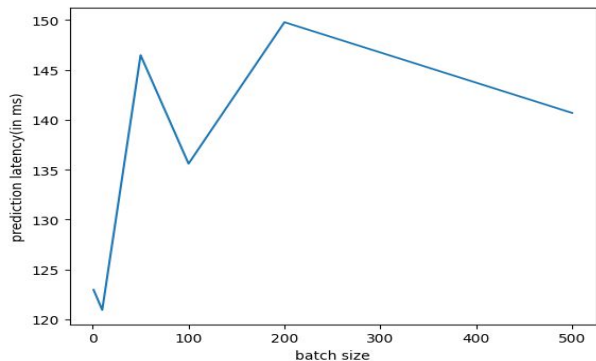
Experimental Results



Test 2: number of data points in a file changes while the number of files streamed remains constant

- The range of total latency in Test 2 is lower than in Test 1 as the batch size is smaller in Test 2. This result is as per expectations as we expect higher latencies when we stream more files together.
- We see that the graph plotted of total latency vs batch size is as expected with an increase in total latency as the batch size increases.
- Overall we can conclude that since the total latency increases with batch size in both Test 1 and Test 2, the system functions as per expectations.

Experimental Results



Preprocessing latency vs prediction latency:

- The values of pre processing latency are observed to be much higher than that of prediction latency. This could be due to the fact that pre processing requires some of its tasks to be sequential, while prediction can be parallelized. Also since pre-processing is performed in batches, the latency might be influenced by the size and characteristics of the batches.
- The graph of pre processing latency oscillates. This could be due to the fact that variations in the nature of the data in different batches. Some batches might contain data that requires more complex pre-processing, leading to higher pre-processing latency.
- The fluctuation in prediction latency could be because for small batch sizes, the system might not fully utilize the parallel processing capabilities of the hardware (e.g., GPU). As the batch size increases, the model can take advantage of parallelism, leading to a decrease in latency.

THANK YOU!