**Individual Project 3: Machine Learning on Streaming Data**

**CS367**

**Cassandra Mason**

**11/03/24**

Table of Contents

[Project Overview 3](#_Toc181568280)

[The Dataset 3](#_Toc181568281)

[The Ingest Job 3](#_Toc181568282)

[Query Algorithm 4](#_Toc181568283)

[Results 6](#_Toc181568284)

[Discussion 6](#_Toc181568285)

[References 7](#_Toc181568286)

# Project Overview

The focus of this individual project is to create a query algorithm on streaming data. The query algorithm is to keep track of incremental changes over time as data streams into a hypothetical data lake from different sources. Changes and edits made to Wikipedia articles was the data stream for my project. My query algorithm kept track of the number of edits/changes that occurred, both from the start of program execution and through several rolling window timeframes, one minute, one hour and one day. A scrolling line graph of the changes to Wikipedia articles over time from the past 24 hours visualizes the results of the query algorithm. Future work includes adding more to the query algorithm and adding an incremental machine learning algorithm.

# The Dataset

The streaming data comes from a PubNub demo stream (*Demo*, n.d.). The stream is of changes made to Wikipedia articles. Each event in the stream contains the following information:

* Event: always a wiki-modification
* Item: the article name
* Link: the link to the Wikipedia article
* User: username of the editor
* Timestamp: the date and time the change was made
* Diff: Only on some events, I think it is the ID to track changes to existing pages

This data allows us to determine topics by the Item, track changes made by specific users, and track the volume of edits over time and for different time windows.

# The Ingest Job

On the Spyder IDE, I used the PubNub and Streamz libraries to ingest the streaming Wikipedia changes data onto my local machine (*Core Streams — Streamz 0.6.4 Documentation*, n.d.). I used the publicly available subscribe key to the Wikipedia changes channel to reference the correct data. I then created a callback class that pushed the PubNub data into a Streamz stream. Within that function, I converted the PubNub “timetoken” variable, which measures number of 100-nanosecond intervals the change was made since January 1, 1970, into a new variable “timestamp”, which is a datetime object. The code is below:



To verify the data was correctly being ingested from PubNub into the Streamz stream, I used the streamz “sink” function to print the stream data to the terminal:

A screenshot of a computer

Description automatically generated

The results were a stream of JSON formatted messages, including the transformed timestamp:

A white text on a black background

Description automatically generated

# Query Algorithm

To start a simple query algorithm on the streaming data, I began tracking the count of Wikipedia changes, or the number of events in the stream. To see the total changes since the program started running, I used a Streamz function, “accumulate” that keeps a running tally of the number of events. The algorithm also contains rolling windows: 1 minute, 1 hour, and 1 day long. The number of events added to the log in those time frames is recorded. This information is useful for finding peak edit times/days on Wikipedia. The algorithm is below:

A computer screen shot of text

Description automatically generated

There are 2 main program actions occurring simultaneously: the continuous incoming stream of data and the continuous query algorithm calculating total edits, and the number of edits in the last minute, hour and day. To allow both these processes to occur simultaneously, I created a thread for the streaming data so it could process as the same time as the query algorithm. I then called the query algorithm to begin tracking Wikipedia changes. The code is below:

A screen shot of a computer program

Description automatically generated

The terminal shows both these processes running, event logs and the query algorithm logging the total number of changes every event, the number of changes in the past minute every 60 seconds, and (I assume, haven’t run it for an hour or a day) the past hour’s events every hour and the past day’s events every day:

A screenshot of a computer

Description automatically generated

# Results

To visualize the count of Wikipedia changes over time, I tried creating a matplotlib line chart to graph a rolling window of 24 hours on the time axis, and a count of Wikipedia edits on the vertical axis. This code section creates a pandas dataframe that keeps track of the timestamp of the events in the stream and the count of events. A function called “process\_window” adds rows to the dataframe of the count of Wikipedia updates, drops records that are older than the last 24 hours, and creates a line graph with the dataframe “timestamp” column as the x axis and the “count” column as the y axis. I have not yet gotten the graph to appear in the Plots section of the Spyder IDE.

A computer screen shot of a program code

Description automatically generated

# Discussion

This project successfully ingested streaming data and applied a basic query algorithm to it to keep a running tally of the number of Wikipedia changes made since the program started, the past minute, the past hour, and the past day.

The next step is to visualize these results in a line chart, which I ran out of time to debug and test.

Future work includes adding more to the query algorithm to filter changes by user, topic, article and user location. This would render interesting insights into who is making edits, how frequently, and to which topics; the topics and articles of highest interest, etc. Deeper insights could be gained with an incremental machine learning algorithm. Incremental learning is a branch of machine learning where algorithms can be continuously refined with streaming data (Awan, 2023). Examples include online support vector machines that find the hyperplane with the greatest margin between categories, and can update with each new piece of data, and incremental decision trees, also a classification algorithm, that selects attributes to split the data on. An incremental machine learning algorithm could help categorize topics in Wikipedia, determine topics of interest, and flag suspicious user edit behavior, such as posting much more frequently or on topics they have never previously taken interest in.

Using the Spyder IDE, PubNub Python SDK, streamz, threading, datetime, and pandas, this project ingests streaming data and incrementally queries the number of Wikipedia changes that are being made in real time.

# References

Awan, A. A. (2023, June 29). *What is Incremental Learning?* https://www.datacamp.com/blog/what-is-incremental-learning

*Core Streams—Streamz 0.6.4 documentation*. (n.d.). Retrieved November 3, 2024, from https://streamz.readthedocs.io/en/latest/core.html#accumulating-state

*Demo: Easily Connect & Test Real-Time Streaming Data*. (n.d.). PubNub. Retrieved October 28, 2024, from https://www.pubnub.com/demos/real-time-data-streaming/