Indexing & Searching for Hot Topics

Done by:
Amogh Tewari(atewari)
Manish Dutta(mdutta3)
Utkarsh Sharma(usharma)

Overview

- In this project we are using Apache Spark and Kafka, along with ElasticSearch to build keyword-based querying and heavy hitter analysis over a real time Twitter text stream.
- To update heavy hitters from the incoming text we have implemented a CountMinSketch algorithms with stop word filtering.
- We have also included DRPC queries over Elastic Search.

Technology and Libraries

- Python 2.7
- Spark
- Kafka
- Zookeeper
- Elasticsearch 2.3.4
- Confluent
- Tweepy API
- Twitter API
- Ubuntu 16.04

Apache ZooKeeper

- It is a centralized service for maintaining configuration information, naming and providing distributed synchronization, and providing group services.
- Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable.
- ZooKeeper aims at distilling the essence of these different services into a very simple interface to a centralized coordination service.

Apache Kafka

- Kafka is a distributed streaming platform which provides fast, highly scalable and redundant messaging through a pub-sub model.
- It consists of four core APIs:
  - Producer API
  - Consumer API
  - Streams API
  - Connector API

Elasticsearch

- Elasticsearch is an open-source, RESTful, distributed search and analytics engine built on Apache Lucene.
- In this we send data in the form of JSON documents to Elasticsearch using the API or ingestion tools.
- It stores the original document and adds a searchable reference to the document in the cluster’s index.
- You can then search and retrieve the document using the Elasticsearch API.

Python Elasticsearch Client

- In this we are running a low level python client for Elasticsearch.
- It’s used to provide common ground for all Elasticsearch-related code in python and is very extendable.
- It is a thin wrapper around Elasticsearch REST API to allow for maximum flexibility.

Spark Streaming

- In this we are using an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.
- Data can be ingested from many sources like Kafka, Flume, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window.
- It provides a high-level abstraction called DStream, which represents a continuous stream of Data.

Confluent Python Client

- We have used confluent-kafka-python client for Apache Kafka.

- It is a high performance, lightweight wrapper around librdkafka.

Kafka Connect Elasticsearch

- Kafka Connect is a connector framework that provides the backbone functionality that lets you connect Kafka to various external systems.
- It either gets data into Kafka or get it out.
- One of such connectors is Kafka Connect Elasticsearch which allows sending data from Kafka to Elasticsearch.
- It uses Jest, which is a HTTP based Elasticsearch client library.

Queries

- We have used a high-level library, called Elasticsearch DSL, to help with writing and running queries against Elasticsearch.
- It stays close to the Elasticsearch JSON DSL, mirroring its terminology and structure.
- It exposes the whole range of the DSL from Python either directly using defined classes or a queryset-like expressions.

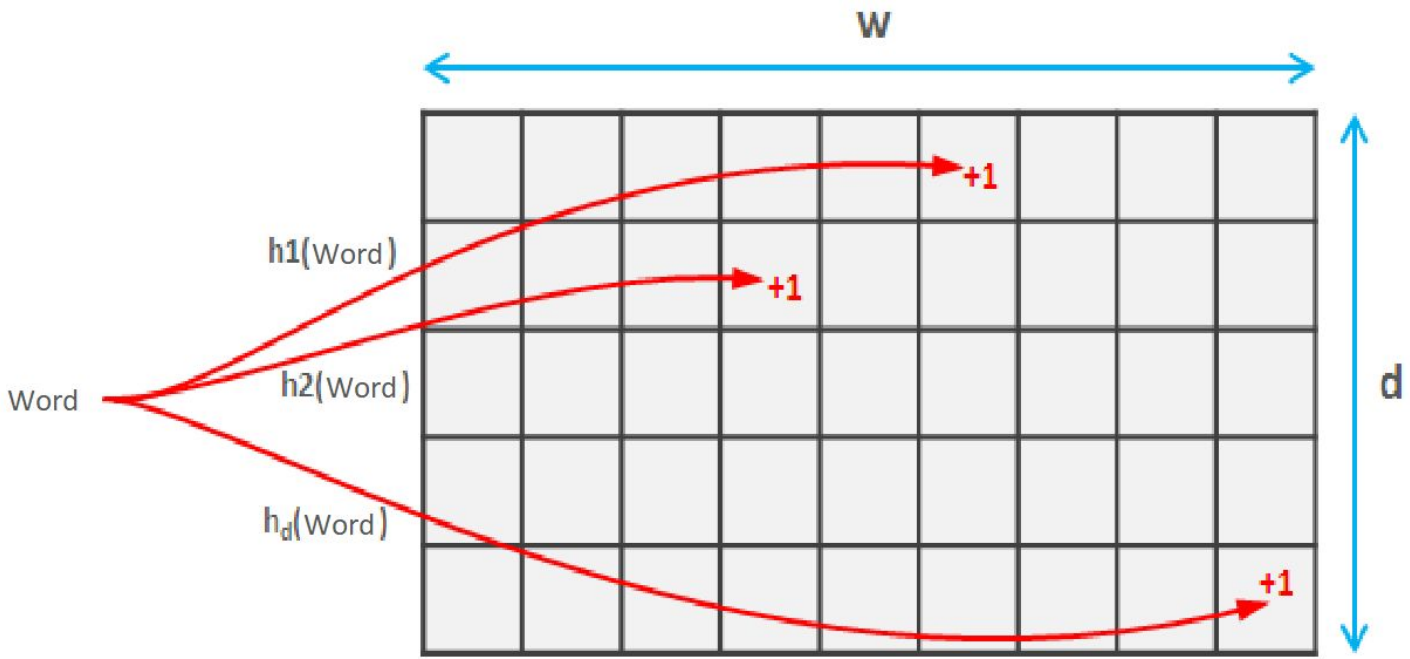
Queries implemented

- Range: This query matches the documents with fields that have terms within a certain range.
- Term: This query finds documents that contain the exact term specified in the inverted index.
- Match: This query is used whenever you need to query any field, the main use is for full-text search.

Count Min Sketch

- Count Min Sketch data structure is a probabilistic data structure that uses multiple hash functions and multiple buckets to increment counters when a word's hashed value resolves to specific buckets.
- In stream data, it would be difficult to keep absolutely accurate counts of words that appear and thus, such a data structure would be a more efficient approximation.
- If the objective is to track the most frequently mentioned words a.k.a. the Heavy Hitters, then Count Min Sketch is a good solution.

Count Min Sketch working for a word



References

Following references were used in the project:

https://docs.confluent.io/current/connect/managing/confluent-hub/client.html#confluent-hub-client

https://docs.confluent.io/current/connect/kafka-connect-elasticsearch/index.html

http://docs.tweepy.org/en/v3.5.0/api.html

https://github.com/barrust/count-min-sketch