Wikipedia Clickstream Anomaly Detection System

by Dirgha Jivani, Aryaman Jalali, and Harshith Keshavamurthy

1.1 ABSTRACT

Designing and implementing a scalable anomaly detection system for Wikipedia clickstream data is the primary goal of this term project. Through monthly aggregated datasets that need distributed processing, Wikipedia clickstream monitors how users move between pages. In order to identify odd patterns like viral traffic spikes, strange navigation routes, and bot-like referrer patterns, our system will use Apache Spark on a computing cluster to process more than 50GB of uncompressed clickstream data. We will construct an end-to-end pipeline that takes in raw clickstream data, uses Spark MLlib statistical techniques and clustering to perform distributed anomaly detection, and uses an interactive dashboard to display the anomalies that have been found. According to the guidelines, we have made sure to make use of public, large-scale data and a Spark cluster.

1.2 I/O EXAMPLES

Input: Wikipedia Clickstream Data (Monthly Aggregates)

prev	curr	type	n
other-search	Cat	external	15000
other-wikipedia	Dog	link	500
other-internal	Main_Page	internal	200

Output: Detected Anomaly

```
{
  "anomaly_id": "anom_202401_001",
  "month": "2024-01",
  "prev": "other-search",
  "curr": "Cat",
  "n": 15000,
  "baseline_median": 500,
  "deviation_ratio": 30.0,
  "z_score": 4.2,
  "anomaly_type": "traffic_spike",
  "confidence": 0.91,
  "description": "30x normal traffic from search engines to Cat page (Z-score: 4.2)"
}
```

1.3 REQUIREMENTS

1.3.1 Definite Requirements

- At least 50GB of uncompressed Wikipedia clickstream data must be processed by the system over several monthly files.
- Traffic spikes exceeding 10 times the historical baseline or indicated by a robust Z-score > 3.5 will be detected by the anomaly detection engine for any prev->curr pair..
- At least three different kinds of anomalies must be produced by the system: navigation-edge anomalies, prev->curr-mix shifts, and spikes in traffic volume.
- The pipeline will output any anomalies that are found to a web dashboard and storage.

1.3.2 Not-decided-yet Requirements

- Cross-month pattern analysis will be used by the system to identify new navigation trends.
- Filtering options by anomaly type and confidence level will be available on the dashboard.
- The system will use the absolute traffic volume to classify anomalies by impact level.

1.3.3 Nice-to-do Requirements

- For extra temporal resolution, Wikipedia pageview data will be incorporated into the system.
- Navigation anomaly network graph visualizations will be part of the dashboard.
- The system will have the ability to export data for additional analysis.

1.4 HOW SUCCESS WILL BE ASSESSED

- **Processing Scale**: The success criterion will be to use Spark distributed processing to process over 50GB of uncompressed Wikipedia clickstream data and quantitatively demonstrate that memory constraints render single-machine processing unfeasible.
- **Anomaly Detection Quality**: Over a 6-month holdout, obtain ≥75% Precision@50 for detected traffic anomalies using Wikimedia trending/viral lists as weak ground truth.
- **Completeness of the Pipeline**: From raw data ingestion to anomaly detection to interactive visualization, the end-to-end pipeline must demonstrate a complete workflow with measurable performance benchmarks.

1.5 METHODOLOGY EXPLANATION

- **Primary Implementation**: Using Spark MLlib for Statistical Anomaly Detection
  Because Wikipedia clickstream data represents monthly aggregates that need distributed processing, we will use PySpark with statistical methods and clustering. We'll put into practice:
    - **Statistical Techniques** with MAD-based robust Z-scores and baseline = median (last four to six months)
    - **K-means** using prev->curr feature vector clustering to identify odd navigation patterns
    - **Window functions** for establishing a baseline and analyzing cross-month trends
    - **approxQuantile** for calculating the percentile threshold
    - **Confidence scores** (0-1) derived from sigmoid of standardized anomaly metrics

An alternate topic of discussion is third-party machine learning libraries.

- We will compare the installation complexity and detection accuracy of libraries like pyspark-iforest, which could offer additional anomaly detection capabilities, even though our primary implementation makes use of core Spark MLlib.

1.6 DATA SOURCES

Primary Data Source: Wikipedia Clickstream Data

- **URL**: https://dumps.wikimedia.org/other/clickstream/
- **Size**: 4-6 months of recent data (e.g., 2023-09 through 2024-02) providing ~56-84GB uncompressed
- **Format**: TSV files with columns: prev, curr, type, n
- **Rationale**: This dataset is genuinely large-scale, publicly available, and represents real user navigation patterns requiring distributed processing.

1.7 CONTRIBUTIONS

Dirgha Jivani:

- **Pipeline Development**: Put basic ETL procedures and Spark data ingestion into practice
- **ML Implementation**: Create a K-means clustering algorithm to identify navigation patterns
- **Dashboard**: Create the dashboard's backend integration and data export API.
- **Documentation**: Create technical documentation for configuring and setting up Spark clusters.

Harshith Keshavamurthy:

- **Pipeline Development:** Manage data partitioning and enhance Spark performance
- **ML Implementation**: Use statistical anomaly detection (MAD, Z-scores)
- **Dashboard**: Produce anomaly trend charts and time-series visualizations
- **Documentation**: Create accuracy analyses and model evaluation reports.

Aryaman Jalali:

- **Pipeline Development**: Configure data storage and control processed output formats
- **ML Implementation**: Create historical trend analysis and baseline calculations
- **Dashboard**: Create and construct the primary interactive dashboard user interface.
- **Documentation**: Produce final reports, presentation slides, and user manuals.

1.8 REFERENCES FOR PROPOSAL PHASE

1. Wikimedia Foundation. "Clickstream Data Release." https://dumps.wikimedia.org/other/clickstream/readme.html
2. Apache Spark MLlib - Clustering https://spark.apache.org/docs/latest/ml-clustering.html
3. Apache Spark SQL - Window Functions https://spark.apache.org/docs/latest/api/sql/
4. Wikimedia Pageviews API - For validation https://wikitech.wikimedia.org/wiki/Analytics/AQS/Pageviews