CS 6350: Big Data Management and Analytics

Assignment 3

Names of students in your group:

1. Kushal Choudhary (kxc240000)

Number of free late days used: 0

Part 1: Spark Streaming with Kafka

Code: https://github.com/Kushal3121/1. Spark Streaming with Kafka

1. Data Source

The data for this project was obtained from the **NewsAPI**, a live REST-based feed providing real-time headlines from major global news outlets.

The Python-based producer script (newsapi_producer.py) fetches new articles periodically from the API using the endpoint https://newsapi.org/v2/top-headlines.

Each news record contains fields such as:

- title - headline text
- author - article author (if available)
- source.name - news outlet
- publishedAt - timestamp of publication

These headlines serve as continuous text input for Named Entity Recognition (NER). The producer publishes each news headline as a JSON message into the **Kafka topic topic1**, simulating a live data stream.

2. Processing and Visualization

A PySpark Structured Streaming job (stream_ner_to_counts.py) reads live messages from Kafka (topic1) and applies SpaCy's NER model (en_core_web_sm) to detect entities such as:

- Organizations (e.g., Apple, Microsoft)
- Persons (e.g., Elon Musk, Joe Biden)
- Locations (e.g., United States, China)

The Spark job maintains a running count of how many times each entity appears and writes these results to Kafka topic topic2.

Logstash then consumes messages from topic2, parses them as JSON, and indexes the entity counts into Elasticsearch under the index name ner-entities.

Finally, Kibana visualizes these results:

- A bar chart displays the top 10 most frequent entities at any given time.
- A line chart tracks entity frequency changes over time (5-minute intervals).

3. Results and Interpretation

The visualization dashboards show that certain entities, such as “Apple,” “AI,” “Trump,” and “United States,” consistently appear among the most mentioned.

This pattern indicates their strong presence in global news discussions and trending topics during the streaming period.

The bar chart highlights static entity dominance at a single point in time, while the line chart demonstrates temporal variation, some entities rise or fall in mentions as new headlines arrive.

Overall, the results confirm:

- Successful real-time ingestion and NER extraction from live sources.
- Proper aggregation and indexing in Elasticsearch.
- Continuous visual updates in Kibana, verifying an end-to-end streaming analytics pipeline.

4. Conclusion

This project demonstrates how Apache Kafka, Spark Streaming, and the Elastic Stack can be integrated to perform real-time NLP analytics.

By automating entity extraction from live news, it provides insights into which people, companies, or countries dominate current news cycles, enabling scalable and continuously updating text analytics.

5. Appendix: Screenshots

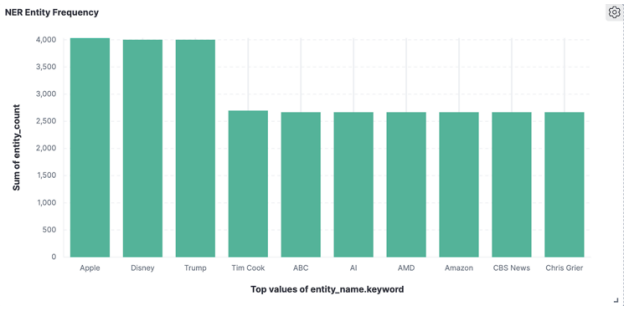

Fig. 1. Top 10 Entities (15 min)

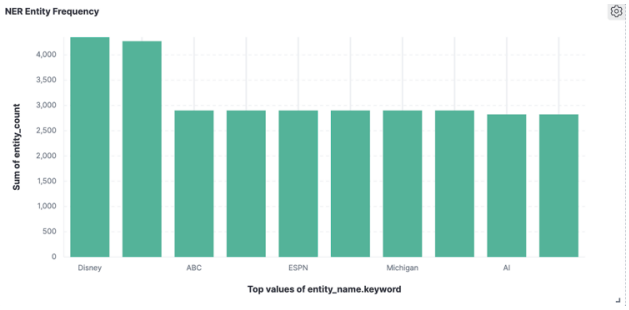

Fig. 2. Top 10 Entities (30 min)

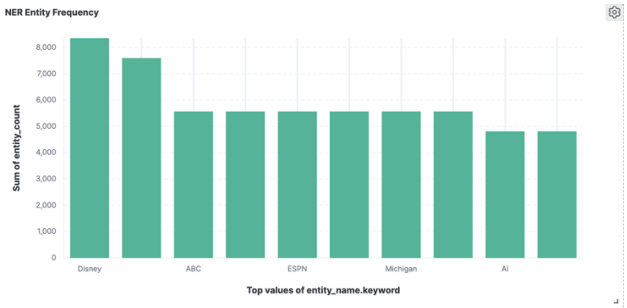

Fig. 3. Top 10 Entities (45 min)

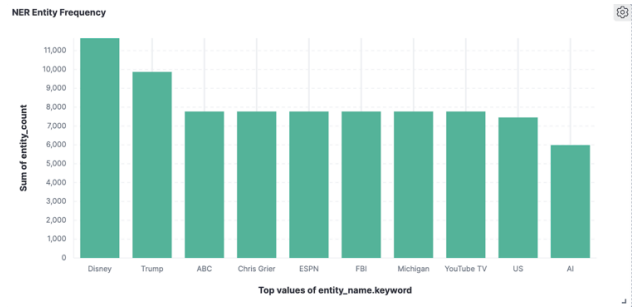

Fig. 4. Top 10 Entities (60 min)

Kushal Choudhary (kxc240000)

Part 2: Analyze Networks with GraphFrame

Code: https://github.com/Kushal3121/2. Analyze Networks with GraphFrame

1. Data Source

- The dataset used for this analysis is the Epinions "Who-trusts-whom" social network, publicly available from the Stanford SNAP repository.
- It represents a directed trust relationship among users, where each edge (src, dst) means user src trusts user dst.
- The graph consists of approximately 75,879 nodes and 508,837 directed edges, making it well-suited for social network analysis using GraphFrames.

2. Processing and Visualization

The dataset was loaded into Apache Spark and analyzed using the GraphFrames API.
The following algorithms and queries were executed:

- Outdegree and Indegree Analysis: to identify the most active and most trusted users.
- PageRank: to measure overall influence or importance of users.
- Connected Components: to detect groups of users connected directly or indirectly.
- Triangle Count: to find tightly connected communities (mutual trust circles).

The computation was performed on a local Spark environment (Spark 3.5.0, GraphFrames 0.8.3) using Python and PySpark.

3. Results and Interpretation

Metric	Key Findings
Outdegree	Node 645 had the highest outdegree (1,801), meaning it trusted the most users.
Indegree	Node 18 had the highest indegree (3,035), indicating it was the most trusted user in the network.
PageRank	Node 18 again ranked highest, confirming it as a central and influential node.
Connected Components	Almost all users (75,877 out of 75,879) were part of a single large component, showing a highly connected trust network.
Triangle Count	Nodes 645, 18, 27, 634, and 44 had the most triangles, forming small clusters of mutual trust.

These results make logical sense in a trust network: a few nodes emerge as highly influential "hubs," while most users form part of a large, interconnected community. The overlap between high PageRank and indegree nodes validates the consistency of centrality measures.

4. Conclusion

This analysis confirms that:

- A small fraction of users dominate trust relationships.

- The network exhibits **scale-free** behavior, a few nodes have very high connectivity.
- The existence of triangles indicates strong local clustering (trusted friend groups).
- Nearly all nodes belong to a single connected component, showing a cohesive community structure.

Overall, the results are both statistically and conceptually consistent with expectations for online social trust networks. GraphFrames proved efficient for analyzing large-scale graph data, enabling computation of influence and structure metrics with minimal code.