Sentiment-Driven Market Volatility Index: A Big Data Approach to Financial Market Prediction

Team: Dev Pratap Singh, Darshan Sivarajah, Harsh Soni,
Manit Garg, Mohammad Yafi Akhtar, Aviral Bansal

Course: DS/CMPSC 410 **Semester:** Fall 2025

1 Introduction and Motivation

1.1 Background

Financial markets exhibit complex dynamics driven by both quantitative metrics and qualitative factors such as investor sentiment. Traditional volatility models like VIX rely primarily on options pricing and historical price movements, potentially missing sentiment-driven market shifts that emerge from news cycles, social discourse, and public opinion.

Recent advances in natural language processing and big data analytics enable us to process vast amounts of unstructured text data to extract meaningful sentiment signals. By correlating these sentiment patterns with market volatility, we can potentially identify early indicators of market movements and develop more comprehensive volatility prediction models.

1.2 Key Problem/Question

Research Question: Can we develop a scalable PySpark-based framework that processes large-scale text data from multiple sources to create a sentiment-driven volatility index that correlates with and potentially predicts short-term market fluctuations better than traditional volatility measures?

1.3 Why Big Data is Relevant

- **Volume:** Millions of news articles, forum posts, and financial reports are generated daily, requiring distributed processing capabilities
- **Velocity:** Real-time sentiment changes can precede market movements, necessitating streaming data processing
- **Variety:** Multiple unstructured text formats (news articles, forum posts, earnings calls) require flexible processing pipelines
- **Complexity:** NLP operations on large corpora are computationally intensive and benefit from distributed computing frameworks like PySpark

2 Project Objectives

2.1 Primary Objectives

1. **Data Pipeline Development:** Build a scalable PySpark pipeline for ingesting and preprocessing large-scale financial text data from multiple sources
2. **Sentiment Analysis at Scale:** Implement distributed sentiment analysis using PySpark MLlib and advanced NLP techniques
3. **Volatility Index Construction:** Develop a sentiment-weighted volatility index and compare its predictive power against traditional measures
4. **Performance Evaluation:** Assess both prediction accuracy and computational scalability across different cluster configurations

2.2 Technical Deliverables

- Robust PySpark data ingestion and preprocessing pipeline
- Scalable sentiment analysis framework using distributed NLP
- Interactive dashboard showing sentiment trends vs. market volatility
- Comprehensive performance analysis and scalability benchmarks
- Well-documented codebase and technical report

3 Data Description

3.1 Data Sources

- **Reddit API:** Posts from r/investing, r/stocks, r/SecurityAnalysis (free tier available)
- **Finlight.me API:** Advanced financial news with sentiment analysis and boolean query filtering (free Launchpad plan, WebSockets support)
- **StockData.org API:** Real-time stock data and news from 5000+ sources with sentiment analysis (100 requests/day free)
- **NewsAPI:** Financial news from major outlets (free tier: 1000 requests/day)
- **SEC EDGAR:** 10-K, 10-Q filings and earnings call transcripts (free, comprehensive)
- **Yahoo Finance RSS:** Market news feeds (free, real-time)
- **Google Trends API:** Search volume data for financial keywords (free)
- **Financial Forums:** Scraped data from Seeking Alpha, MarketWatch forums (where legally permissible)

3.2 Expected Data Characteristics

- **Volume:** Approximately 8-12 million text documents over the semester (∼40-80 GB raw text)
- **Format:** Primarily unstructured text with timestamps, sentiment scores, and rich metadata
- **Languages:** Primarily English with some multilingual content (30+ languages via StockData.org)
- **Time Range:** Historical data (2020-present) + real-time feeds during project period
- **Real-time Capabilities:** WebSocket streaming via Finlight.me for live sentiment analysis
- **Enhanced Features:** Built-in sentiment analysis, entity tagging, and advanced filtering

3.3 Data Quality Challenges

- **Noise:** Sarcasm, slang, and informal language in social media
- **Bias:** Platform-specific user demographics and sentiment distributions
- **Spam:** Bot accounts and promotional content requiring filtering
- **Missing Data:** Incomplete timestamps or corrupted text encoding
- **Rate Limits:** API throttling requiring efficient batch processing strategies

4 Technical Approach

4.1 Technology Stack

- **Core Framework:** PySpark (DataFrames, RDDs, MLlib, Spark SQL, Spark Streaming)
- **NLP Libraries:** NLTK, spaCy integrated with Spark UDFs
- **Data Storage:** HDFS with Parquet format for efficient columnar storage
- **Visualization:** Matplotlib, Seaborn, Plotly for interactive dashboards
- **APIs & Integration:**
  - requests, BeautifulSoup, praw (Reddit)
  - finlight-python SDK for advanced financial news queries
  - stockdata.org REST API for multi-source news aggregation
  - newsapi-python for general financial news
  - WebSocket clients for real-time data streaming

4.2 Architecture Overview

1. **Data Ingestion Layer:**
    - Scheduled batch jobs for historical data collection
    - Real-time streaming ingestion using Spark Streaming
    - Data validation and deduplication pipelines
2. **Processing Layer:**
    - Distributed text preprocessing (tokenization, cleaning, normalization)
    - Feature extraction using TF-IDF and n-gram analysis
    - Sentiment scoring using pre-trained models and lexicon-based approaches
3. **Analytics Layer:**
    - Time-series aggregation of sentiment scores
    - Correlation analysis with market volatility indices (VIX, realized volatility)
    - Predictive modeling using Spark MLlib (Linear Regression, Random Forest)
4. **Presentation Layer:**
    - Interactive web dashboard showing real-time sentiment trends
    - Statistical analysis and model performance metrics

4.3 Scalability Considerations

- **Partitioning Strategy:** Data partitioned by date and source for optimal query performance
- **Caching:** Intermediate results cached in memory for iterative ML algorithms
- **Resource Management:** Dynamic resource allocation based on workload requirements
- **Fault Tolerance:** Checkpointing for long-running jobs and data lineage tracking

5 Expected Outcomes

5.1 Primary Deliverables

1. **Sentiment-Volatility Index:** A novel composite index combining sentiment scores with traditional volatility measures
2. **Predictive Model:** ML model that can forecast short-term market volatility (1-5 days) using sentiment indicators
3. **Scalable Pipeline:** Production-ready PySpark pipeline capable of processing 100GB+ datasets
4. **Performance Dashboard:** Interactive visualization showing real-time sentiment trends and predictions

5.2 Success Metrics

- **Prediction Accuracy:** Correlation coefficient > 0.6 between sentiment index and next-day volatility
- **Scalability:** Linear speedup with increasing cluster nodes (efficiency $> 80\%$)
- **Processing Throughput:** Handle $> 1.5M$ documents per hour on the available cluster
- **Model Performance:** RMSE improvement of $> 15\%$ over baseline volatility models
- **Real-time Latency:** Process streaming sentiment updates within < 5 seconds via WebSocket integration
- **Data Quality:** Sentiment accuracy $> 75\%$ validated against manually labeled sample data

6 Work Plan & Timeline

Week	Tasks and Milestones
1-2	Literature review, API setup, data source validation Define sentiment lexicons and baseline models
3-4	Implement data ingestion pipeline for Reddit, NewsAPI Set up HDFS storage and initial data collection
5-6	Develop PySpark preprocessing pipeline Text cleaning, tokenization, and feature extraction
7-8	Implement distributed sentiment analysis Integrate multiple NLP approaches and validation
9	Construct sentiment-volatility index Correlation analysis with market data
10	Build predictive models using MLlib Develop interactive dashboard and visualizations
11	Performance testing and scalability analysis Model validation and backtesting
12	Final report preparation and presentation Code documentation and project delivery

Table 1: Detailed Work Plan and Timeline

7 Division of Labor

- **Data Acquisition & Infrastructure Lead (2 members):**
  - API integration and web scraping implementation
  - HDFS setup and data pipeline architecture

- **NLP & Analytics Lead (2 members):**
  - Sentiment analysis model development
  - Statistical correlation analysis and model validation
- **Visualization & Frontend Lead (1 member):**
  - Dashboard development and user interface design
  - Performance metrics visualization
- **Documentation & Project Management Lead (1 member):**
  - Technical documentation and report writing
  - Project coordination and timeline management

8 Potential Challenges & Mitigation Strategies

8.1 Technical Challenges

- **Challenge:** API rate limiting and quota management across multiple services
- **Mitigation:** Implement intelligent caching, use multiple free-tier accounts, prioritize high-quality APIs like Finlight.me and StockData.org
- **Challenge:** Integrating real-time WebSocket streams with batch processing pipelines
- **Mitigation:** Use Spark Streaming for real-time ingestion, implement proper buffering and check-pointing mechanisms
- **Challenge:** Noisy text data affecting sentiment accuracy
- **Mitigation:** Leverage built-in sentiment analysis from APIs, implement ensemble methods, manual validation on sample data
- **Challenge:** Spark cluster resource optimization with diverse data sources
- **Mitigation:** Profile jobs extensively, implement source-specific partitioning strategies, use appropriate caching
- **Challenge:** Maintaining data quality consistency across different API formats and schemas
- **Mitigation:** Implement robust schema validation, create unified data models, extensive API response testing

8.2 Data Quality Challenges

- **Challenge:** Inconsistent data quality across different sources
- **Mitigation:** Implement comprehensive data validation pipelines and quality scoring
- **Challenge:** Market regime changes affecting model validity
- **Mitigation:** Use rolling window validation and implement model retraining mechanisms

9 References

- Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1), 1-8.
- Zhang, X., Fuehres, H., & Gloor, P. A. (2011). Predicting stock market indicators through Twitter sentiment analysis. *Procedia-Social and Behavioral Sciences*, 26, 55-62.
- Preethi, G., Santhi, B., Gokulnath, C., & Rajesh, L. (2017). Sentiment analysis for stock price prediction. In *Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems* (pp. 1604-1609).
- **Technical Documentation:**
  - Apache Spark Documentation: `https://spark.apache.org/docs/latest/`
  - Reddit API (PRAW): `https://praw.readthedocs.io/`
  - Finlight.me API Documentation: `https://docs.finlight.me/`
  - StockData.org API Documentation: `https://www.stockdata.org/documentation`
  - NewsAPI Documentation: `https://newsapi.org/docs`
  - SEC EDGAR Database: `https://www.sec.gov/edgar`
  - Yahoo Finance API: `https://python-yahoofinance.readthedocs.io/`
- **Datasets & Resources:**
  - VADER Sentiment Analysis: `https://github.com/cjhutto/vaderSentiment`
  - Financial sentiment lexicons from Loughran-McDonald
  - Historical VIX data from CBOE: `https://www.cboe.com/tradable_products/vix/`