Technical Documentation (Project Market Signal)

Github repo: https://github.com/vyomthakkar/market-signal

Data collection

X scraper built with Playwright for automated browser control. Targets hashtag-based tweet collection.

Anti-detection strategies:

1. Browser Fingerprint Masking
    - Disabling automation flags (AutomationControlled)
    - Realistic user agent (Chrome 120, macOS)
    - Viewport: 1920×1080
2. Mimicing Human-Like Behavior
    - Random scroll timing: 2-4 second delays
    - Variable delays between hashtags: 5-10 seconds
    - Multiple login strategies with fallbacks (role selector → text selector → Enter key)
3. Rate Limiting Architecture
    - Token Bucket Algorithm + Adaptive Rate Limiter

I tried scraping initially with nitter, snscrape and twscrape but they were not as resilient as playwright browser approach to X's bot detection algorithms

Optimizations Involving Time and Space Complexity:

1. O(1) Deduplication (TweetCollector): Custom dual data structure using a list for ordered storage and a set for constant-time membership testing, eliminating O(n) linear search overhead (1000x speedup for large datasets).

2. TF-IDF: Leverages NumPy/SciPy matrix operations for batch term frequency calculations, replacing iterative loops with BLAS-optimized vector operations

3. Lazy RoBERTa Model Loading: Defers 500MB sentiment model initialization until first use, reducing startup time from ~8s to <1s and enabling analysis of non-sentiment features without memory overhead.

4. Batch Processing in Analysis Pipeline: Vectorized Pandas DataFrame operations.

Concurrent Processing Implementation:

Implemented multiprocessing-based parallelization for sentiment analysis pipeline, achieving 4-8x speedup on multi-core systems and demonstrating scalability for 10x larger datasets.

Made the decision to not parallelize the data collection/scraping because concurrent requests might trigger rate limiting/bot detection from X.

Memory-efficient data handling for large datasets:

1. Parquet Storage Format: reduces storage and better performance
2. Vectorized Batch Processing

Text-to-Signal Conversion Pipeline

Multi-stage pipeline converting unstructured tweet text into quantitative trading signals with confidence-weighted reliability scoring.

Input: Raw tweet text + engagement metrics (likes, retweets, replies, views)
Output: Signal score ∈ [-1, +1], categorical label, confidence ∈ [0, 1], uncertainty interval

Input: Raw tweet text + engagement metrics (likes, retweets, views)

3-Stage Pipeline:

1.1 Sentiment Analysis (Primary Signal)
- Model: cardiffnlp/twitter-roberta-base-sentiment-latest (125M parameters)
- Process: Tweet → Tokenization → Transformer encoding → Softmax(logits) → 3-class probabilities
- Output: P(negative), P(neutral), P(positive)
- Conversion: sentiment_score = P(pos) × 1 + P(neu) × 0 + P(neg) × (-1) ∈ [-1, +1]
- Confidence: max(P(negative), P(neutral), P(positive))

1.2 Domain-Specific Keyword Enhancement
- Bullish keywords: {rally, breakout, uptrend, buy, calls, strength, target hit, ...}
- Bearish keywords: {crash, breakdown, downtrend, sell, puts, weakness, stop loss, ...}
- Boost calculation: keyword_boost = clip((bullish_count - bearish_count) / 3, -1, +1) × 0.3
- Purpose: Correct general sentiment model for finance-specific language

1.3 TF-IDF Vectorization
- Algorithm: Scikit-learn TfidfVectorizer with 1000 features
- Parameters: unigrams + bigrams, min_df=2, max_df=0.8

- Output: Top-10 terms per tweet ranked by TF-IDF score
- Purpose: Identify distinguishing terms for content quality assessment

1.4 Engagement Metrics → Virality Score
- engagement_rate = (likes + retweets + replies) / views × 1000
- virality_ratio = retweets / likes (high retweets = viral content)
- reply_ratio = replies / total_engagement (discussion indicator)

Stage 2: Confidence Scoring (Reliability Assessment)

2.1 Content Quality (40% weight)
- Base score: min(finance_term_density × 10, 1.0)
- Spam penalty: 70% reduction if keywords like {telegram, subscribe, free, channel} detected
- Technical boost: 30% increase if {support, resistance, breakout, RSI, MACD} present
- Range: [0, 1]

2.2 Sentiment Strength (30% weight)
- Source: RoBERTa model confidence from Stage 1
- Interpretation: How certain the model is about sentiment direction
- Typical values: 0.5-0.9 for clear sentiment, 0.3-0.5 for ambiguous

2.3 Social Proof (30% weight)
- Source: Virality score from Stage 1.4
- Rationale: High-engagement tweets = crowd validation
- Limitation: Can amplify noise (viral spam)

Stage 3: Signal Generation with Confidence Dampening

3.1 Base Signal Calculation
base_signal = (sentiment_score + keyword_boost) × (0.5 + virality_score × 0.5)

Interpretation:
- Sentiment provides direction (-1 to +1)
- Virality amplifies signal (0.5x to 1.0x multiplier)
- High virality = stronger conviction

3.2 Confidence Dampening
final_signal = base_signal × confidence

Effect:
- High confidence (0.8): Signal mostly preserved

- Medium confidence (0.5): Signal halved
- Low confidence (0.3): Signal heavily dampened → labeled IGNORE

3.3 Signal Classification

Condition	Label	Action
confidence < 0.3	IGNORE	Insufficient reliability, skip
final_signal ≥ 0.5	STRONG_BUY	High conviction long
0.2 ≤ final_signal < 0.5	BUY	Moderate long position
-0.2 < final_signal < 0.2	HOLD	No clear direction
-0.5 < final_signal ≤ -0.2	SELL	Moderate short position
final_signal ≤ -0.5	STRONG_SELL	High conviction short

3.4 Confidence Intervals (Uncertainty Quantification)

margin = (1 - confidence) × 0.5
lower_bound = max(final_signal - margin, -1.0)
upper_bound = min(final_signal + margin, 1.0)

Interpretation:

- Low confidence → Wide interval (high uncertainty)
- High confidence → Narrow interval (precise estimate)

Stage 4: Aggregation

Consensus Classification:

- Bullish ratio > 70% → STRONG_BULLISH consensus
- Bullish ratio > 50% → BULLISH consensus
- Mixed signals → MIXED consensus (market indecision)

Risk Indicators:

- signal_volatility = std(signals) → Measures disagreement
- High volatility + low confidence → Unreliable market sentiment