Presentation Outline: Tweet-Driven Alpha — Sentiment & Correlation Strategies

A 6-slide deck covering strategy mechanics, the ticker filtering pipeline, and live backtest results.

Slide 1 — Overview: What Are We Building?

Title: Tweet-Driven Alpha: Extracting Market Signals from Social Data

Key points:

- Hypothesis: public tweets contain statistically meaningful information about near-term price moves
- Two primary signal extraction methods: **Sentiment Analysis** (NLP-based) and **Correlation Discovery** (statistical)
- Supplementary methods: Bag of Words, Embedding similarity, Multi-timeframe scoring, Ensemble
- All methods feed into a unified backtesting engine trading across 13 tickers (SPY, TLT, GLD, etc.)
- Each method is evaluated on a strict out-of-sample test set — the last 30% of all tweet data by time

Visual suggestion: high-level system diagram — tweets → signal methods → backtest engine → portfolio equity curve

---

Slide 2 — Sentiment Strategy: FinBERT-Powered Signal

Title: Sentiment Strategy — How a Tweet Becomes a Trade

Every tweet passes through four sequential filters before a trade is ever placed.

Stage 1 — Tweet Cleaning (`TweetCleaner`)

Raw tweets arrive with HTML markup, URLs, and duplicate noise. Before any NLP runs:

Step	What it does
HTML tag removal	Strips `<p>`, ` `, etc. via regex
Entity decoding	`&` → `&`, `'` → `'`
URL removal	Drops all `http(s)://` and `www.` links
Whitespace normalisation	Collapses multiple spaces, trims
Short tweet removal	Drops any tweet with ≤ 5 characters after cleaning
Deduplication	Removes exact duplicates by `tweet_id` or text

Stage 2 — Importance Filtering (Top 40%)

A composite importance score (0–1) is computed per tweet. **Only the top 40% pass forward.**

Factor	Weight	Logic
Length	20%	Longer tweets are more substantive
Engagement	40%	replies + reblogs + favourites, normalised
Keyword density	30%	Hits on financial/political terms (tariff, fed, inflation, recessio
Uniqueness	10%	Penalises very short or repetitive content

This step eliminates ~60% of raw tweets before any model is run. The effect is fewer but higher-quality signals feeding into FinBERT.

Stage 3 — FinBERT Sentiment Scoring

The surviving tweets are passed through `ProsusAI/finbert`, a BERT transformer fine-tuned on financial filings and news:

- Outputs three probabilities: **positive**, **negative**, **neutral**
- Raw sentiment score = `positive_prob - negative_prob` ∈ [-1, +1]
- Confidence = `abs(positive_prob - negative_prob)` — measures conviction, not just direction
- If the transformer is unavailable, a keyword fallback fires (bullish/bearish word lists)

Stage 4 — Ticker Relevance Weighting

The same tweet can have different relevance for different tickers. Each ticker has a domain keyword list:

Ticker	Relevance keywords

SPY	market, economy, stocks, equities, s&p
TLT	bonds, treasury, interest, fed, rates
GLD	gold, inflation, dollar, currency

A tweet mentioning "fed rate decision" gets higher confidence on TLT than on SPY:

```
adjusted_confidence = base_confidence × (1 + ticker_relevance) [capped at 1.0]
influence_score = sentiment_score × adjusted_confidence × 0.005
```

The `0.005` scalar calibrates output to realistic tweet price impact (±0.5% per trade).

Stage 5 — Influence Score Threshold

Only tweets where `|influence_score| ≥ 0.0003` trigger a trade. Tweets that pass cleaning and importance filtering but produce a weak sentiment signal are silently discarded at this final gate.

Visual suggestion: vertical pipeline diagram — Raw tweets → Clean → Importance filter (drop 60%) → FinBERT → Relevance weight → Threshold gate → Trade

---

Slide 3 — Correlation Strategy: Statistical Feature Discovery

Title: Correlation Strategy — Learning What Features Actually Predict Returns

Correlation goes further than sentiment — instead of classifying text, it learns *which measurable properties of a tweet historically predict price moves*, then scores new tweets against those patterns.

Stage 1 — Same Cleaning & Importance Filter

Identical to Sentiment: HTML cleaning, deduplication, then top 40% by composite importance score. The strategy only ever sees high-quality tweets.

Stage 2 — 70 / 30 Time-Ordered Train / Test Split

This is the critical difference from sentiment. Before any statistical analysis runs, the dataset is divided **by time**, not randomly:

```
train set = first 70% of tweets (chronological) ← correlations learned here
test set = last 30% of tweets (chronological) ← backtesting happens here only
```

Why time-ordered? Because random splitting would leak future information into the training set. A tweet from 2024-Q4 appearing in the training fold would allow correlations to be computed using data the model "shouldn't know yet". The time split enforces strict causality.

Stage 3 — Safe Feature Extraction

Three categories of features are extracted from each tweet. Price-derived features are **explicitly blocked**:

Category	Allowed?	Examples
Embedding features	Yes	`embedding_pca_0` … `embedding_pca_49` — semantic co
Engagement metrics	Yes	`replies_count`, `reblogs_count`, `favourites_count`, `log_tot
Tweet-derived	Yes	`tweet_word_count`, engagement ratios
Price features	**No**	`ema_*`, `rsi_*`, `macd_*`, `vol_*`, `bb_*`, `atr_*` — blocked

Price features are blocked because they are computed from bars that are timestamped *after* the tweet, meaning including them would mean trading on information that wasn't available at signal time.

Stage 4 — Per-Feature Statistical Filtering

On the training set only, for every (feature, ticker) pair:

1. Align tweet feature values to that ticker's 5-minute forward return
2. Compute **Pearson correlation** and **p-value**
3. Compute **mutual information** to catch non-linear relationships
4. Apply filter: keep if `p_value < 0.05` AND `|correlation| ≥ 0.02`
5. If fewer than 3 features survive, relax to top 10 by p-value with `min_abs_corr × 0.5`
6. Retain top 20 features per ticker, ranked by `|correlation|`

The result: a small, statistically significant feature set per ticker. For SPY this might be "high engagement + specific PCA embedding dimensions". For GLD it might be "reblogs count + a different embedding axis".

Stage 5 — Scoring a New Tweet (Test Set)

For each incoming tweet in the 30% test window:

```
influence_score = Σ (feature_value × correlation_coeff × |weight|)
÷ sqrt(n_matched_features)
```

The `sqrt(n)` dilution is intentional: dividing by `n` would overly punish tweets with many weak features, while dividing by `sqrt(n)` preserves the signal from a single very strong feature match.

Confidence = `min(1.0, n_matched_features / 10.0)` — a tweet matching all 10 top features gets confidence 1.0.

Stage 6 — Influence Score Threshold

Same gate as sentiment but tighter: `|influence_score| ≥ 0.0002`. The correlation method fires fewer trades (26 vs 55 for sentiment) precisely because its threshold requires a statistically grounded signal, not just a positive/negative classification.

Visual suggestion: two-panel diagram — left: training phase (feature selection on 70%), right: inference phase (scoring + threshold on 30%)

Slide 4 — The Full Filtering Pipeline

Title: From Raw Tweet to Executed Trade — 8 Stages

Both strategies share a common outer pipeline. The differences between them emerge inside stages 4 and 5.

Pipeline Overview

```
Raw tweet archive (all tickers, all time)
■
[1] Data availability check
■ Only tickers with a .parquet price file survive
■ ~8–10 of 13 tickers pass
↓
[2] Tweet–ticker alignment
■ tweets_df filtered to matched tickers
↓
[3] Tweet cleaning (TweetCleaner)
■ HTML, URLs, dedup, short-tweet removal
↓
[4] Importance filter (top 40%)
■ Composite score: engagement 40%, keywords 30%, length 20%, uniqueness 10%
■ ~60% of tweets dropped here
↓
[5] Method-specific training (70% of data by time)
■ SENTIMENT: FinBERT scores each tweet — no training data needed
■ CORRELATION: Learns per-ticker feature→return correlations on train set
↓
[6] Influence score threshold
■ Sentiment: |score| ≥ 0.0003
■ Correlation: |score| ≥ 0.0002
■ Weak signals dropped — no trade fired
↓
[7] Regime-adaptive adjustment (optional)
■ Bull / bear / sideways detection
■ Dynamically raises threshold in low-conviction regimes
↓
[8] Daily trade cap + time-of-day filter
■ Max 3 trades per ticker per day
■ Default: 12:00–14:00 ET only
↓
Trade executed
```

Where Sentiment and Correlation Diverge

Stage	Sentiment	Correlation
Training requirement	None — FinBERT is pre-trained	Must learn correlations on 70% train set first

Feature input	Raw tweet text	Embedding PCA vectors + engagement numerics
Scoring mechanism	FinBERT positive/negative probability	Weighted sum of historical feature correlations
Ticker specificity	Keyword-based relevance boost per ticker	Fully separate feature set per ticker
Influence threshold	0.0003	0.0002
Resulting trade count	55 (higher volume)	26 (more selective)

Key Architecture Principle

Stages 1–5 protect data integrity — no look-ahead, no circular logic, no price features leaking into tweet scoring.

Stages 6–8 protect trade quality — only high-conviction signals, no overtrading, no off-hours noise.

Visual suggestion: side-by-side funnel — one column per strategy, tweets narrowing to trades, showing where each strategy diverges

---

Slide 5 — Results: Equity Curve & Performance Metrics

Title: Backtest Results — Out-of-Sample Performance (Last 30% of Data)

Performance table (all methods, out-of-sample):

Method	Total Return	Final Capital	Trades	Win Rate	Avg Return/Trade
Sentiment	**+6.37%**	$106,372	55	58.2%	1.02%
Correlation	**+4.14%**	$104,142	26	61.5%	1.78%
Ensemble	+3.19%	$103,188	34	55.9%	1.37%
Multi-timeframe	+2.57%	$102,572	19	68.4%	2.38%
Embedding	+2.39%	$102,388	33	51.5%	1.24%
Bag of Words	+0.49%	$100,492	26	53.8%	0.47%

Key observations:

- Sentiment leads on raw return (+6.37%) with the most trades (55) — high throughput, moderate precision
- Correlation leads on per-trade quality (1.78% avg return, 61.5% win rate) — fewer but more selective trades
- Multi-timeframe has the highest win rate (68.4%) but lowest trade count (19) — very selective filter
- Bag of Words is weakest — proves that surface-level keyword matching without semantic understanding barely works
- All 6 methods beat a flat baseline, validating the hypothesis that tweet signals are exploitable

Visual suggestion: overlaid equity curves for all 6 methods from `output/tweet_ticker_methods/all_methods_comparison.png`

Slide 6 — Risk-Adjusted Returns & Key Takeaways

Title: Risk-Adjusted Analysis & What We Learned

Risk-adjusted lens:

- **Sentiment:** highest absolute return but most trades — examine drawdown and Sharpe via `quantstats_report.html`
- **Correlation:** best return-per-trade ratio (1.78%) — signal is selective and high-conviction by design; the statistical filtering is doing real work
- **Multi-timeframe:** highest win rate (68.4%) with lowest trade count — extremely conservative entry criteria, essentially only trades when multiple time horizons agree
- Core trade-off: *volume* (sentiment, 55 trades) vs *precision* (correlation/multi-timeframe, 19–26 trades)
- Monthly heatmaps available per method: `output/tweet_ticker_methods/{method}/monthly_heatmap.png`

Key takeaways:

1. The importance filter (top 40%) is load-bearing — without it, weaker tweets dilute the signal
2. Semantic understanding (FinBERT, embeddings) strongly outperforms surface keyword matching (Bag of Words: +0.49%)
3. The 70/30 time split is not conservative padding — it is the mechanism that prevents the correlation method from cheating
4. The ensemble underperforms the top individual methods — likely signal dilution from weaker sub-strategies being included
5. Next steps: live forward-test correlation and sentiment, explore confidence-scaled position sizing

Visual suggestion: bar chart of total return by method + 2×2 quadrant of win rate vs avg return/trade showing precision/volume trade-off

---

Assets Available for Each Slide

All output files are in `output/tweet_ticker_methods/{method}/`:

File	Use in slide
`equity_curve.png`	Slide 5 — individual equity curves
`equity_curve_simple.png`	Slide 5 — clean version for presentation
`all_methods_comparison.png`	Slide 5 — overlaid comparison chart
`monthly_heatmap.png`	Slide 6 — return distribution by month
`metrics.txt`	Slides 5–6 — source of all numeric metrics
`quantstats_report.html`	Slide 6 — full Sharpe, Sortino, max drawdown
`trades.csv`	Slide 4 — trade-level detail if needed
`top_movers_dump.csv`	Slides 2/3 — examples of top signal tweets