



FinAnalytics: A Survey of Coordinated and Self-Validating Multi-Agent Language-Model Systems for Financial Analysis

Sahil S. Patil ^a, Prabhleen Kaur ^{a*}, Anurag Sutar ^a

^a Department of Computer Engineering, Smt. Indira Gandhi College of Engineering, Ghansoli, India

ABSTRACT

Financial analysis draws on several kinds of source, including filings, earnings calls, news, and market data, but the tools that analyse them are separate, so analysts reconcile sentiment, event, fundamental, risk, and forecasting signals by hand. This survey reviews ten core studies (2019–2025) and 32 supporting works on financial language models, event extraction, sentiment-driven forecasting, risk assessment, multi-criteria company ranking, and multi-agent LLM systems. We propose a five-dimensional taxonomy and compare the core studies across eight analytical capabilities. Most of them address one or two tasks in isolation, and in our reading none uses confidence scoring to reconcile conflicting outputs across components. We identify six research gaps and describe how FinAnalytics, a metacognitive multi-agent platform with a Confidence & Safety Checker, answers each one through its architecture, data design, requirements, and evaluation plan. The paper ends with applications, open challenges, best practices, and a roadmap.

Keywords: Financial Intelligence, Multi-Agent Systems, Metacognition, Sentiment Analysis, FinBERT, Stock Forecasting, Risk Assessment, Explainable AI, Literature Review

1. Introduction

1.1 Motivation and Background

Investors, analysts, and researchers work from a steady flow of mixed information: regulatory filings, earnings-call transcripts, financial news, and market prices. Each source is usually examined with its own tool, one for financial metrics, one for sentiment, one for document retrieval, and one for prediction. The analysis ends up fragmented and labour-intensive, and there is no systematic way to check whether the different signals agree.

Work in natural language processing [11], [27], time-series forecasting [15], [16], and large language model (LLM) agents [2], [22], [23] has produced strong solutions to individual tasks. What remains open is how to *coordinate* these specialised components and *validate* their combined output so that the final result can be trusted, explained, and traced to its sources.

1.2 Problem Statement

Three limitations come up repeatedly in current practice. Analysis is **fragmented** across tools, with filings, earnings calls, news, and market data handled separately. Because metrics, sentiment, document retrieval, chatbots, and prediction each sit in a different tool, users face **high manual effort and information overload**. Most systems also target a **single analytical task** instead of an integrated workflow, with little multi-agent coordination and little support for comparing companies.

A worked illustration. An analyst is reviewing a technology company before its earnings call. Recent news reads positive and the price is trending upward, but the valuation ratio is stretched, risk indicators are elevated, and a regulatory event is pending. Each tool reports its own result, and none of them tells the analyst that the signals disagree or how much weight each one deserves. This survey is mainly concerned with detecting that kind of disagreement.

1.3 Research Questions

RQ1: Which analytical tasks and techniques have been applied to financial text and numerical data, and how much of a unified workflow do they cover?

RQ2: How do existing multi-agent and LLM-based systems coordinate specialised agents, and how do they justify their outputs?

RQ3: How are conflicts among sentiment, event, fundamental, risk, and forecast signals detected and handled?

RQ4: Which gaps does a unified platform need to close, and under what cost and hardware constraints can it be built?

1.4 Evolution of Financial Analytics

Computational financial analysis has gone through four broad phases, summarised in Table 1. The date ranges are approximate and overlap.

Phase 1 (before ~2010): Statistical and ratio-based. The dominant tools were fundamental ratio analysis, econometric models such as ARIMA [24], and lexicon-based text scoring.

Phase 2 (~2010–2017): Classical machine learning. SVM-style classifiers, hand-crafted text features, and multi-criteria decision methods such as TOPSIS [25] came into common use. Systematic reviews of text-based market prediction appeared in this period [35].

Phase 3 (~2018–2022): Deep learning. LSTMs [16], deep belief and recurrent networks for risk [6], pretrained transformers [11], and domain models such as FinBERT [1], [13] made sentiment-aware forecasting possible [4], [5], [9]. Deep-learning forecasting from this period is surveyed in [34].

Phase 4 (~2023–present): LLMs and agents. Domain LLMs [18], [19], retrieval augmentation [20], reflection [21], and role-based multi-agent frameworks [2] moved attention from single models to coordinated systems. Surveys of finance-specific LLMs [33] and of general agents [36], [37] document this change.

Each phase changed what was analysed as well as how. Early work used one numeric or lexical signal. Later work added learned representations of text and sequences, and the most recent systems combine several sources and analytical roles. The part that has lagged behind is the layer that decides whether the combined output can be trusted, which Sections 6 and 7 take up.

Table 1 - Summary of the four phases.

Phase	Typical inputs	Typical techniques	Persistent limitation
1	Prices, ratios, word lists	ARIMA, ratio analysis, lexicons	Context-blind; one signal at a time
2	Structured features, hand-crafted text features	SVM-style classifiers, MCDM ranking	Manual feature engineering
3	News, headlines, price series	LSTM, BERT/FinBERT, Prophet	Models used in isolation; weak explanation
4	Multi-source text and tables	LLMs, retrieval, agents, reflection	Cost; coordination and validation under-studied

1.5 Scope and Contributions

Task-wise review: sentiment, event detection, forecasting, risk, fundamental ranking, and multi-agent coordination, with a critique of each (Sections 3–7).

Five-dimensional taxonomy that classifies financial-intelligence systems by data, task, technique, architecture, and output assurance (Section 2).

Capability matrix comparing ten core studies on eight analytical capabilities (Table 13).

Gap analysis: six research gaps, with particular attention to cross-component validation, which has received little study (Section 9).

System-level positioning of FinAnalytics: architecture, agent and checker design, data flow, requirements, database, interface, feasibility, plan, and evaluation (Section 10), followed by applications, challenges, best practices, and a roadmap (Sections 11–14).

1.6 Organization

Section 2 sets out the taxonomy and the survey method. Sections 3–7 review text analytics, forecasting and risk, fundamental ranking, multi-agent systems, and explainability. Section 8 covers data and metrics, Section 9 gives the comparative analysis and the gaps, and Section 10 describes FinAnalytics. Sections 11–16 discuss applications, challenges, future directions, best practices, the limitations of this survey, and our conclusions.

2. Taxonomy and Methodology

2.1 Five-Dimensional Taxonomy

We classify systems along five dimensions (Fig. 1). **D1 Data modality** runs from numerical market data to news text, filings and earnings calls, or a mix of these. **D2 Analytical task** covers sentiment, events, fundamentals, risk, forecasting, and comparison. **D3 Modelling technique** includes statistical methods, classical ML and MCDM, deep learning, pretrained language models, and LLM agents. **D4 System architecture** separates single-task models, sequential pipelines, multi-agent debate, and coordinator-routed agents. **D5 Output assurance** records how results are justified: not at all, by post-hoc explanation, by debate rationale, or by cross-agent validation with confidence scores.

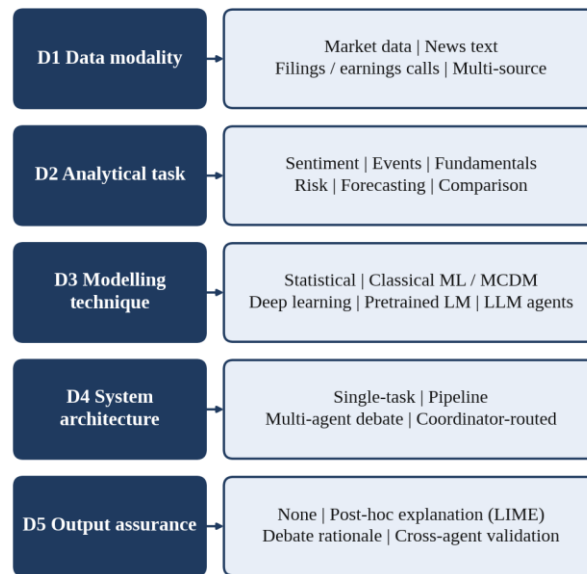


Fig. 1 - Five-dimensional taxonomy of financial-intelligence systems.

The dimensions are independent by design. A system can apply an advanced technique (D3) inside a single-task architecture (D4) with no output assurance (D5), and most of the reviewed studies do exactly that. A simple technique can equally sit inside a coordinated architecture with explicit validation. The taxonomy therefore keeps *what a model can do* apart from *how a system is assembled and justified*, and the second question is the one a unified platform has to answer.

3.2 Survey Methodology

This is a focused survey, and it does not attempt an exhaustive systematic review. We selected ten *core studies*, each covering at least one analytical task within the scope of a unified financial-intelligence platform, and added 32 *supporting works* (models, methods, datasets, related surveys, and theory) needed to read them (Table 2).

Inclusion criteria: (i) core studies published 2019–2025; (ii) the work addresses sentiment, events, fundamentals, risk, forecasting, comparison, or multi-agent coordination in finance; (iii) it reports a method with experimental evaluation, or it is a recognised survey. **Exclusion criteria:** (i) non-English; (ii) no financial application; (iii) too little technical detail. The core studies use different markets, datasets, and metrics, so we compare their *capabilities and design choices* and do not rank their reported numbers.

Table 2 - Composition of the reviewed corpus.

Category	No.	Refs.
Core studies on FinAnalytics-relevant tasks	10	[1]–[10]
Language models and transformer basics	5	[11]–[13], [27], [38]
Domain LLMs, retrieval, reasoning	4	[18]–[20], [30]
Agent frameworks, reflection, agent surveys	6	[21]–[23], [36], [37], [40]
Forecasting, ranking, risk, portfolio, market theory	7	[15], [16], [24]–[26], [39], [41]
Explainability and calibration	2	[17], [31]
Datasets, lexicons, metacognition theory	5	[14], [28], [29], [32], [42]
Related surveys	3	[33]–[35]
Total	42	

3.3 Summary of Core Studies

Table 3 summarises the ten core studies by data and task, method, and main limitation. Sections 3–7 discuss each in more detail.

Table 3 - Summary of core studies.

Ref.	Data / task	Method	Key limitation
[1]	Financial corpora; text mining	Domain-pretrained BERT, multiple pretraining tasks	Improves the NLP layer only
[2]	Simulated trading decisions	Role-based multi-agent LLM, Bull–Bear debate	No mechanism to reconcile conflicting outputs
[3]	Event sentences; Wikipedia sections	Weak supervision; sentence classifier	Depends on Wikipedia; limited event typing
[4]	NSE prices + 210,000+ headlines	Sentiment features + LSTM; LIME	Model-level explanation only
[5]	Intraday prices + news	FinBERT sentiment + LSTM	Forecasting focus; no risk or events
[6]	Market data incl. seven exchange-rate markets	Deep belief network + RNN	Systemic risk, not company-level
[7]	DJIA company ratios	Entropy–DEMATEL–TOPSIS	Ratios only; no text or events
[8]	Historical stock prices	Prophet vs. ARIMA	Price-only; no other signals
[9]	Asset news + prices	Semantic attention + sentiment-aware LSTM + mean–variance	Return–stability trade-off; better fusion needed
[10]	Ten NLP applications in finance	Literature survey	Proposes no new model

4. Financial Text Analytics

4.1 Why Financial Text Is Difficult

Financial language has its own vocabulary, frequent hedging, and sentiment that depends on context. Loughran and McDonald [42] showed that general-purpose word lists misclassify many words in financial filings: words that read as negative in everyday English are often neutral in finance. Their finding led first to finance-specific lexicons and later to domain-adapted neural models.

4.2 Domain Language Models

BERT [11] and the Transformer [27] made pretrained language models the default for text understanding; [38] gives a broader overview of LLMs. FinBERT [1] pretrains a BERT-style model on large financial corpora with several pretraining tasks and reports that it captures financial language and semantics better than general-purpose models. A different model, also named FinBERT [13], fine-tunes BERT for financial sentiment and is evaluated on the Financial PhraseBank [14]. Very large finance-specific LLMs such as BloombergGPT [18] and the open-source FinGPT [19] take this line further.

DistilBERT [12] compresses BERT by knowledge distillation. Its authors report a model about 40% smaller and 60% faster that keeps about 97% of BERT's language-understanding performance, which makes it a reasonable choice where no GPU is available.

Table 4 - Sentiment analysis approaches.

Approach	Strength	Limitation	Typical role
Lexicon-based (finance lexicon [42], TextBlob)	Fast, transparent, no training	Misses context and negation	Fallback or baseline
Fine-tuned general encoder (DistilBERT [12])	Light; CPU-friendly	Domain mismatch if tuned on general text	Interactive systems
Domain-pretrained encoder (FinBERT [1], [13])	Captures financial semantics	Needs fine-tuning data	Sentiment features [5]
Large finance LLM [18], [19]	Broad capability	Heavy compute; hallucination risk	Generation, QA

4.3 A Practical Sentiment Pipeline

A typical pipeline (Table 5) cleans and segments the text, tokenises it, classifies sentiment with a transformer encoder, and outputs a label with a score. Transformer encoders take inputs of limited length (512 tokens for BERT-family models), so long filings and transcripts are split into passages whose

predictions are then aggregated. A lexicon-based scorer can act as a fallback when the neural model is unavailable. Attaching a *confidence flag* to each result lets a later validation layer weigh sentiment against other evidence.

Table 5 - Stages of a financial sentiment pipeline.

Stage	Operation	Main risk
Cleaning	Remove markup, boilerplate and disclaimers; normalise numbers and tickers	Discarding useful context
Segmentation	Split text into sentences or passages within the model's input limit	Splitting an argument in the middle
Tokenisation	Convert to sub-word tokens; truncate at the model limit	Truncated evidence
Classification	Transformer encoder outputs class probabilities	Domain mismatch
Aggregation	Combine passage-level results into one label per document	Averaging hides mixed polarity
Fallback	Lexicon scorer when the neural model is unavailable	Lower accuracy; must be flagged
Confidence flag	Attach a confidence value to each label	Over-confident scores

Two design choices need care. The first is aggregation. A transcript that is positive in its prepared remarks and negative in the question-and-answer session should not collapse into one neutral label without any note of the disagreement, so passage-level results are best kept next to the document-level summary. The second is domain shift. A general-domain classifier may read hedged or technical phrasing literally. Domain-tuned models [1], [13] and the finance lexicon of Loughran and McDonald [42] improve accuracy for this reason, and a platform should report how sure it is as well as what it predicts.

Critical analysis. The domain-model studies improve the *NLP layer* and stop at text classification. They do not cover forecasting, risk, event analysis, or the integration of several sources [1]. A general-domain model fine-tuned on non-financial sentiment can also mislabel financial phrasing, so its confidence should be reported.

4.4 Financial Event Detection

Ein-Dor et al. [3] derive training labels from relevant Wikipedia sections, a form of weak supervision, so that no hand-built event knowledge base is needed. A sentence-level classifier finds event-related sentences and generalises to companies it has not seen. The method depends on how much Wikipedia content exists and how good it is, offers only coarse event typing, and identifies event sentences without analysing different event types.

Events that matter in practice usually include earnings announcements, mergers and acquisitions, litigation, management changes, product launches, regulatory actions, and changes in guidance; the list is illustrative. When labelled data is scarce, rule-based or keyword-based detectors are a sensible option. They are transparent and cheap, but they lose recall on unusual phrasing and struggle to tell an actual event from a passing mention. A hybrid design, with rules for high-precision categories and learned classifiers for the rest, is the obvious next step.

In such a hybrid, cleaned text is split into sentences and a transparent rule or keyword detector runs first. A learned classifier trained with weak supervision [3] can later pick up categories and phrasings the rules miss. Every detection carries an event type and a confidence score for downstream components to weigh. Table 6 lists example event categories and the lexical cues a rule-based detector might use; these cues are examples and have not been validated as a lexicon.

Table 6 - Illustrative event categories and lexical cues.

Category	Example cues	Downstream use
Earnings	earnings, quarterly results, EPS	Sentiment and forecast context
Mergers and acquisitions	acquire, merger, takeover	Event risk
Litigation / regulatory	lawsuit, probe, fine, approval	Event risk
Management change	appoints, resigns, steps down	Governance signal
Product / partnership	launch, unveil, partnership	Positive-event context
Guidance	raises or cuts guidance, outlook	Forecast context
Supply / operations	disruption, shortage, recall	Event risk

Precision and recall trade off against each other here. A strict keyword rule seldom raises a false alarm but misses paraphrased events, while a permissive rule or a learned classifier finds more events and more spurious ones. A false event can inflate a risk score, and a missed event can hide a real risk, so the detector should report a confidence value and the Checker should treat low-confidence detections as supporting evidence only.

4.5 From Sentiment to Price Signals

Three studies link text sentiment to price prediction. Gite et al. [4] combine NSE price data with over 210,000 news headlines to train an LSTM and use LIME for feature-level explanations. Sidogi et al. [5] pair FinBERT sentiment with an LSTM for intraday prediction, compare it with price-only and generic-BERT variants, and find that domain-specific sentiment helps. Muthivhi and van Zyl [9] combine a semantic attention model for asset-specific sentiment with a sentiment-aware LSTM and a mean–variance portfolio strategy. Their sentiment-aware strategies earned higher average revenue, though the authors note a trade-off between return and stability and call for better fusion methods.

Critical analysis. Sentiment adds information beyond price, but all three studies use it only as an *input feature* to a forecaster. None asks how sentiment should be reconciled with fundamentals, risk, or events when they disagree.

5. Forecasting and Risk Assessment

5.1 Statistical and Decomposable Models

ARIMA [24] is still the standard statistical baseline for time-series forecasting. With backshift operator B , an ARIMA(p, d, q) model is

$$\phi(B)(1 - B)^d y_t = \theta(B) \varepsilon_t \quad (1)$$

where ϕ and θ are the autoregressive and moving-average polynomials. Prophet [15] fits an additive decomposition instead:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \quad (2)$$

with trend $g(t)$, seasonality $s(t)$, and holiday effects $h(t)$, plus automatic changepoint detection and uncertainty intervals. Garlapati et al. [8] compare Prophet and ARIMA for stock-price forecasting and find that Prophet captures trend and changepoints while ARIMA gives a statistical reference point. Both use historical prices alone and ignore company metrics, sentiment, events, and risk.

5.2 Deep Sequence Models

LSTMs [16] capture long-range temporal dependencies with gated memory:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (3)$$

$$h_t = o_t \odot \tanh(C_t) \quad (4)$$

LSTMs are the forecasting component in the sentiment-enhanced predictors of [4], [5], and [9]. They usually need more data and tuning than additive models, and a point forecast on its own says little about how reliable it is.

Table 7 - Forecasting models compared.

Model	Inputs	Strength	Limitation
ARIMA [24]	Price history	Statistical baseline; interpretable	Linear; needs stationarity handling
Prophet [15]	Price history	Trend, changepoints, intervals; fast	No exogenous text signals by default
LSTM [16]	Prices (+ features)	Long-range temporal patterns	Data-hungry; opaque; point output
Sentiment + LSTM [4], [5], [9]	Prices + sentiment	Adds textual information	Sentiment as feature only; no validation

5.3 Limits of Prediction

Stock series are noisy and non-stationary, and the efficient-market hypothesis [41] holds that prices already reflect available information. However strongly one accepts the hypothesis, forecasts are best presented as *scenarios with uncertainty*. A trend direction with a confidence interval is a more honest output than a single predicted price, and any accuracy claim should be tested against simple baselines.

A decision-support platform should therefore report a forecast in three parts: the predicted trend direction, the uncertainty interval, and the horizon covered. Prophet [15] produces the interval directly, and a wide interval relative to the forecast is in itself a reason for low confidence. Reporting all three together stops users from reading a point estimate as a promise.

5.4 Risk Assessment

Sun and Li [6] apply deep belief networks and recurrent networks to financial-market data, including seven major exchange-rate markets, to rebuild early-warning indicators of systemic risk, and report better prediction and measurement than traditional models. Their focus is *systemic and exchange-rate risk*;

they do not address company-level risk that combines financial metrics, sentiment, and events. TradingAgents [2] has a dedicated risk agent, but it works inside a trading-decision setting.

Company-level risk has several dimensions (Table 8). The usual quantitative building blocks are log returns, volatility, maximum drawdown (Eq. 18), and value at risk:

$$r_t = \ln(P_t / P_{t-1}) \quad (5)$$

$$\sigma = \sqrt{[(1/(n-1)) \sum (r_t - \bar{r})^2]} \quad (6)$$

$$P(L > \text{VaR}_\alpha) = 1 - \alpha \quad (7)$$

Table 8 - Dimensions of company-level risk.

Dimension	Typical indicators	Source of signal
Market risk	Volatility, drawdown, beta	Price history
Financial risk	Leverage, liquidity, profitability ratios	Company financial data
Sentiment risk	Persistently negative tone	News, calls, filings
Event risk	Litigation, regulatory action, guidance cuts	Event detection
Systemic risk [6]	Cross-market early-warning indicators	Market-wide indicators

To build a company-level score, indicators from each dimension are normalised to a common scale, weighted, and summed into a risk score, which is then mapped to a level such as low, moderate, or high (Table 9). Storing the contributing factors with the score keeps the result explainable. The weights and cut-offs are design parameters, and they should be documented and tested. Since the score mixes numeric, textual, and event inputs, it should also be checked against the forecast and sentiment before it is displayed.

Table 9 works through a hypothetical case. Each indicator is normalised to the range 0 to 1, with 1 meaning highest risk, the weights sum to one, and the weighted sum is mapped to a level with cut-offs at 0.33 and 0.66. The numbers are invented for illustration and do not come from any system.

Table 9 - Illustrative risk-score composition (hypothetical values).

Indicator	Normalised value	Weight	Contribution
Price volatility	0.70	0.30	0.21
Valuation (P/E)	0.55	0.20	0.11
Negative sentiment	0.40	0.25	0.10
Event risk	0.80	0.25	0.20
Risk score		1.00	0.62 (moderate)

Because each indicator's contribution is recorded, a reader can see that volatility and a pending event drive the result, and can check whether the sentiment contribution fits the forecast before accepting the level.

Critical analysis. None of the reviewed risk studies combines these dimensions into a single company-level view, and doing so needs inputs from several analytical agents at the same time.

6. Fundamental Analysis and Multi-Criteria Ranking

Liew et al. [7] combine Entropy, DEMATEL, and TOPSIS to weight financial ratios objectively, reveal causal relations among them, and rank DJIA companies. ROE, current ratio, and debt-to-equity ratio turned out to be the most influential ratios, and the resulting portfolio earned a higher mean return than the DJIA benchmark at minimum risk. Entropy weighting sets each criterion's weight according to its information content:

$$w_j = (1 - e_j) / \sum_k (1 - e_k) \quad (8)$$

and TOPSIS [25] ranks alternative i by how close it is to the ideal solution:

$$C_i = D_i^- / (D_i^+ + D_i^-) \quad (9)$$

where e_j is the entropy of criterion j and D_i^+ , D_i^- are distances to the ideal and anti-ideal solutions. At the portfolio level, mean–variance optimisation [39] chooses weights w that minimise portfolio variance for a target return, the approach used on top of sentiment-aware forecasts in [9]:

$$\min_w w^T \Sigma w \quad \text{subject to} \quad w^T \mu = \mu_{\text{target}} \quad (10)$$

Criteria have to be normalised before weighting because ratios differ in scale and direction. Return on equity is a benefit criterion (higher is better) and debt-to-equity is a cost criterion (lower is better). Min-max normalisation maps every criterion to a common range and inverts the cost criteria, which makes a weighted sum meaningful. The lighter weighted-scoring comparison discussed below rests on the same idea.

Critical analysis. The ranking method is transparent and systematic, but it uses financial indicators only and ignores textual sentiment and events. Lighter schemes based on normalisation, weighted scoring, and ranking are easier to extend with further signals and work well enough for interactive comparison of a few companies. Their weights should still be documented and, where possible, justified with a method such as entropy weighting.

7. Multi-Agent LLM Systems and Metacognition

7.1 General Agent Frameworks

AutoGen [22] and MetaGPT [23] show that role-specialised agents exchanging structured messages can solve tasks that a single prompt handles poorly. ReAct [40] interleaves reasoning with actions, chain-of-thought prompting [30] draws out step-by-step reasoning, and retrieval-augmented generation [20] grounds answers in source documents. Surveys of LLM-based agents [36], [37] catalogue profile, memory, planning, and action modules and point to open problems in reliability and evaluation.

7.2 Multi-Agent Systems in Finance

TradingAgents [2] is modelled on a trading firm, with fundamental, sentiment, technical, research, trading, and risk agents. The agents communicate in a structured format and hold Bull-Bear debates, and the experiments report better returns, Sharpe ratio, and maximum drawdown than baseline strategies. The framework is built for automated trading and simulated investing. The debates produce explanations, but the study does not look at any mechanism for reconciling conflicting outputs across analyses.

7.3 Orchestration Patterns

Three orchestration patterns appear in the literature (Fig. 2). In a **sequential pipeline** each stage feeds the next, as in sentiment-to-forecast systems [4], [5]; errors carry forward and nothing reconciles the stages. In **role-based debate** [2], agents argue opposing positions and a decision-maker settles them, which gives readable rationales but is geared to a trading decision. In **coordinator-routed** systems, a central component interprets the request, picks the agents to run, and sends their outputs to a checker. The set of analyses then follows the query, and every result goes through the same validation.

7.4 Metacognition and Self-Monitoring

In psychology, metacognition means knowing about and regulating one's own cognitive processes [32]. In AI systems it shows up as self-reflection [21] and, more generally, as *monitoring and control*: deciding what reasoning a task needs and judging how reliable the result is. A financial-intelligence platform can split this into two roles, a **coordinator** that interprets a query and selects the relevant agents, and a **checker** that tests consistency and assigns confidence before results reach the user.

In a query-driven platform, monitoring starts before any analysis runs. The coordinator parses the request, works out whether it concerns one company or several and which analyses it needs, and skips agents whose output the question does not call for. Control comes after the analysis, when the checker decides whether the combined evidence is consistent enough to report with high confidence.

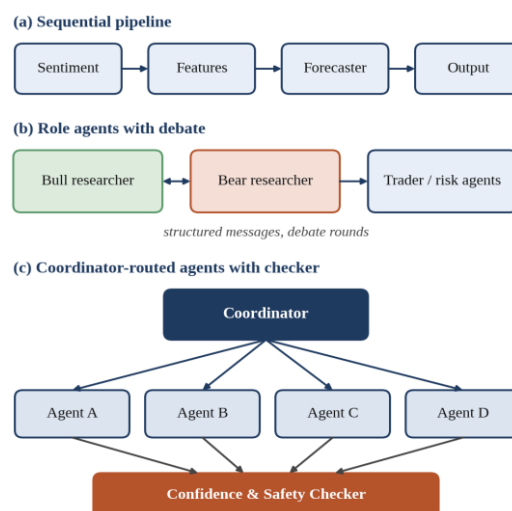


Fig. 2 - Orchestration patterns: (a) pipeline, (b) debate, (c) coordinator-routed with checker.

Table 10 - Architectural patterns for financial analysis.

Pattern	Strength	Limitation
Single-task model [1], [8]	Focused; easy to evaluate	No cross-signal view
Sequential pipeline [4], [5]	Simple integration	Errors propagate; no reconciliation
Role agents with debate [2]	Interpretable rationales; diverse views	Trading focus; no explicit conflict validation
Coordinator-routed agents + checker	Query-driven task selection; confidence-scored output	Needs sensible thresholds; more components

Costs of agent systems. Each added agent increases integration effort, and LLM-based agents also add latency and cost. Agents built from classical models and lightweight encoders avoid much of this cost and are easier to inspect, though they reason less flexibly. Deployment constraints should drive the choice; Section 10 returns to this point.

8. Explainability, Confidence and Validation

Most explainability in the reviewed studies works at the *model level*. Gite et al. [4] use LIME [17] to attribute individual predictions to input features. That helps a reader interpret one prediction, but it does not validate the system as a whole. Du et al. [10] list data quality, privacy, domain specialisation, and interpretability among the open problems of financial NLP and name multi-source integration and reliable, interpretable financial AI as directions for further research.

Confidence is a second requirement. Modern neural networks are often poorly calibrated [31], so raw model scores should not be shown as probabilities without a check. Calibration is usually measured with the expected calibration error:

$$ECE = \sum_m (|B_m| / N) \cdot |\text{acc}(B_m) - \text{conf}(B_m)| \quad (11)$$

where predictions are grouped into bins B_m . A platform that combines several analyses must also deal with *cross-agent conflicts* (Table 11).

Table 11 - Typical cross-signal conflicts.

Conflict	Example	Typical handling
Sentiment vs. fundamentals	Positive news tone, weak ratios	Report both; lower confidence
Forecast vs. risk	Upward trend, elevated risk score	Flag; show forecast interval
Event vs. sentiment	Negative event, positive tone	Surface the event; flag mismatch
Data availability	Live feed unavailable	Use fallback; mark low confidence

We use the term *system-level validation* for handling these conflicts explicitly: flagging them, lowering confidence, and showing both sides. It can be arranged in layers. Per-agent threshold checks catch implausible outputs, cross-agent consistency checks compare signals, confidence is then assigned to each result and to the whole report, and the report carries flags that show users where the evidence is weak.

Validation layers in practice. A threshold check could flag a negative price-to-earnings ratio or a volatility value outside a plausible range. A consistency check could compare a positive sentiment result with a high risk level, or a rising forecast with a recent adverse event. Confidence assignment combines the agent confidences with the amount of disagreement, and flagging makes the report say which results are weak and why. Running these steps after the agents and before the report keeps the checks independent of any one model.

As one possible aggregation, offered here only to illustrate the design space and not taken from any published system, overall confidence can be reduced as disagreement grows:

$$C_{\text{overall}} = (\sum_k w_k c_k) \cdot (1 - \lambda \cdot \rho) \quad (12)$$

where c_k is agent k 's confidence, w_k its weight, ρ the fraction of conflicting signal pairs, and λ a penalty factor to be chosen and tested empirically. Of all the capabilities discussed here, system-level validation is the one the surveyed work most often lacks.

Confidence helps only when users can see where it comes from. A source-backed report therefore lists, for each finding, its data source, retrieval time, and confidence level, and shows the overall confidence together with any flags, so a reader can separate a well-supported conclusion from a tentative one. Given the calibration problems noted above, showing confidence as a graded level instead of a bare percentage also avoids false precision.

9. Data Sources and Evaluation

9.1 Data Sources

Financial-intelligence systems use both numerical and textual sources (Table 12). Open and free-tier sources keep research systems cheap to run, though each comes with limits on coverage, licensing, and request rates.

Table 12 - Common data sources.

Source	Content	Used for	Caveat
yfinance	Prices, company data	Numeric agents	Unofficial wrapper over Yahoo Finance; rate limits and terms apply
SEC EDGAR [28]	Annual and periodic filings	Fundamentals, events	Long, unstructured text
Financial PhraseBank [14]	Labelled financial sentences	Sentiment evaluation	Small; English only
FiQA [29]	Financial opinions and QA	Sentiment, QA	Limited size
Earnings calls	Transcripts	Sentiment, events	Availability and licensing
Wikipedia labels [3]	Weak event labels	Event training	Depends on page coverage

Evaluation Pitfalls

Financial machine learning has three common pitfalls. **Look-ahead bias** happens when information that was not available at prediction time leaks into features or labels; splits must be time-ordered, never random. **Survivorship bias** appears when only companies that are still listed are studied. **Overfitting to a single period** inflates results, so evaluation should cover different market regimes. Backtests should also state their transaction-cost assumptions.

9.2 Evaluation Metrics

Classification (sentiment, events) is measured with precision, recall, and F1, and macro-averaged F1 is the better choice when classes are imbalanced:

$$P = TP / (TP + FP), \quad R = TP / (TP + FN) \quad (13)$$

$$F1 = 2PR / (P + R) \quad (14)$$

Forecasting is measured with RMSE and MAPE against a baseline such as ARIMA [24], plus interval coverage when the model produces intervals:

$$RMSE = \sqrt{[(1/n) \sum (y_t - \hat{y}_t)^2]} \quad (15)$$

$$MAPE = (100/n) \sum |y_t - \hat{y}_t| / |y_t| \quad (16)$$

Strategy-level studies such as [2] report the Sharpe ratio [26] and maximum drawdown:

$$\text{Sharpe} = (R_p - R_f) / \sigma_p \quad (17)$$

$$MDD = \max_i (Peak_i - V_i) / Peak_i \quad (18)$$

System-level evaluation should also measure latency, cache effectiveness, conflict-detection accuracy, and calibration (Eq. 11).

Comparability Problem

The core studies differ in market (NSE, DJIA, exchange-rate markets), horizon, and metric, so their reported numbers cannot be compared directly. We recommend shared benchmarks and the reporting of baselines, and return to this in Section 14.

10. Comparative Analysis and Research Gaps

10.1 . Capability Matrix

Table 13 shows which of eight capabilities each core study addresses, judged from the main focus each study describes. ● marks primary coverage, ○ partial or discussion-only coverage, and – no coverage.

Table 13 - Capability coverage of core studies.

Study	Sent	Evt	Fund	Fcst	Risk	Cmp	MAS	Val
[1] FinBERT	●	—	—	—	—	—	—	—
[2] TradingAgents	●	—	●	○	●	—	●	○
[3] Event extraction	—	●	—	—	—	—	—	—
[4] Explainable sentiment	●	—	—	●	—	—	—	○
[5] FinBERT + LSTM	●	—	—	●	—	—	—	—
[6] DL risk	—	—	—	—	○	—	—	—
[7] Entropy–TOPSIS	—	—	●	—	—	●	—	—
[8] Prophet vs. ARIMA	—	—	—	●	—	—	—	—
[9] Sentiment + price fusion	●	—	—	●	—	—	—	—
[10] NLP in finance (survey)	○	—	—	○	○	—	—	○
FinAnalytics (proposed)	●	●	●	●	●	●	●	●

Sent = sentiment; *Evt* = event detection; *Fund* = fundamental metrics; *Fcst* = forecasting; *Cmp* = multi-company comparison; *MAS* = multi-agent coordination; *Val* = cross-component validation.

Observations. Of the nine technical studies, five have a single analytical objective ([1], [3], [6], [7], [8]), three pair sentiment with forecasting ([4], [5], [9]), and only [2] covers several roles through agents. Sentiment and forecasting are well covered. Events, comparison, multi-agent coordination, and validation are each the main subject of one study at most. No core study offers cross-component validation as a primary contribution.

10.2 Comparison with Related Surveys

Several surveys cover nearby ground (Table 14). Du et al. [10] survey NLP applications in finance broadly, Li et al. [33] survey LLMs in finance, Sezer et al. [34] review deep-learning forecasting, Nassirtoussi et al. [35] review text-based market prediction, and Xi et al. [36] and Wang et al. [37] survey LLM-based agents in general. As far as we know, none of them organises the field around the coordination and validation of different financial analyses, which is the focus of this survey.

Table 14 - Related surveys.

Survey	Primary scope	Finance-specific	Agent coordination
Du et al. [10]	NLP applications in finance	Yes	Not a focus
Li et al. [33]	LLMs in finance	Yes	Limited
Sezer et al. [34]	Deep-learning forecasting	Yes	No
Nassirtoussi et al. [35]	Text-based market prediction	Yes	No
Xi et al. [36]; Wang et al. [37]	LLM-based agents (general)	No	Yes
This survey	Coordinated, validated financial intelligence	Yes	Yes

10.3 Research Gaps

G1 Fragmented analysis: metrics, sentiment, events, and prediction each have their own tools, and the reviewed work rarely covers more than two tasks.

G2 No cross-component validation: conflicting outputs are neither reconciled nor given confidence scores. The multi-agent study [2] does not examine such a mechanism either.

G3 Limited company-level risk: risk research targets systemic or exchange-rate risk [6] and does not combine company-level signals from metrics, sentiment, and events.

G4 Point forecasts without uncertainty: many studies report predictions with no interval or trend direction, which overstates certainty given the limits set out in Section 4.3.

G5 Explanations at model level only: LIME-style attributions [4] neither tie the whole report to its sources nor show where evidence conflicts.

G6 Heavy infrastructure: large LLMs and GPU requirements put these systems out of reach for many education and research settings.

11. The FinAnalytics Approach

11.1 Overview

FinAnalytics is a metacognitive multi-agent platform for financial intelligence, designed to address the gaps above. A user enters a company or ticker and a natural-language query in a React dashboard. The Flask backend hands the request to a **Metacognitive Coordinator**, which decides which analyses are needed and routes the query to specialised agents for financial data, sentiment, events, risk, prediction, and comparison. A **Confidence & Safety Checker** then runs threshold-based and cross-agent consistency checks, reconciles conflicts, and assigns confidence scores. The outputs are compiled into a Financial Intelligence Report, displayed with Chart.js, and can be exported as a PDF with ReportLab (Fig. 3).

The architecture (Fig. 3) has layers. The presentation layer (React) collects the ticker and query. The API layer (Flask, with CORS enabled) exposes the /api/analyze endpoint. The Metacognitive Coordinator handles query parsing, agent selection, and task routing, six specialised agents run the analyses, the Confidence & Safety Checker validates and scores their outputs, and a report generator builds a JSON report for the dashboard and an optional PDF. External inputs (market data, financial text, and the DistilBERT, TextBlob, and Prophet models) feed only the agents, so a data source can be swapped without changing the layers above.

11.2 Objectives and Scope

The platform has five objectives: (i) a metacognitive multi-agent system in which a central coordinator interprets and processes financial queries; (ii) analysis of sentiment, events, financial metrics, and multi-company comparison from several data sources; (iii) risk assessment and AI-assisted forecasts with confidence intervals; (iv) confidence and safety checks that cross-check outputs, reconcile conflicts, and flag low-confidence results; and (v) explainable reports, tied to their sources, with visualisations and PDF export. In scope are natural-language queries on supported companies, integration of real-time and historical data, comparative analysis, and an interactive dashboard, all for educational and research use.

11.3 Agents and Their Roles

Metacognitive Coordinator: interprets the query, works out which analyses are needed, and routes work to the relevant agents.

Financial Data Agent: fetches market and company data through yfinance, with caching and request throttling.

Sentiment Agent: classifies financial text with a DistilBERT model and falls back to TextBlob.

Event Agent: detects financial events using rules and keywords.

Risk Agent: combines financial metrics, sentiment, and events into a risk assessment.

Predictive Agent: forecasts price trends with Prophet and reports confidence intervals.

Comparative Agent: normalises metrics, applies weighted scoring, and ranks companies.

Confidence & Safety Checker: runs threshold-based and cross-agent consistency checks, reconciles conflicts, and assigns confidence scores.

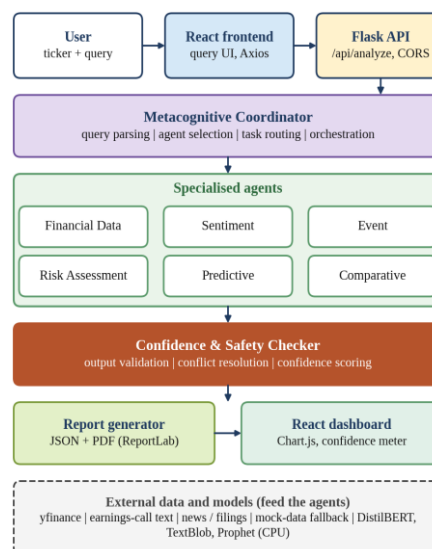


Fig. 3 - FinAnalytics layered architecture.

11.4 Agent Design Details

Table 15 describes each component as an input–process–output unit, matching the functional analysis in Table 17. All agents return results in the same form (agent name, raw output, and confidence), so the Checker can treat them uniformly and the database can store them in one agent-result entity (Table 21). Agents that do not depend on each other's outputs can run in parallel; the Risk Agent waits for the metric, sentiment, and event results.

Table 15 - Design of the FinAnalytics components.

Component	Input	Method	Output
Coordinator	Company, query	Company extraction; query parsing; agent routing	Required analyses; agents to invoke
Financial Data	Ticker	yfinance retrieval with caching and throttling	Price, P/E, volatility, market cap
Sentiment	Financial text	DistilBERT classification; TextBlob fallback	Label, score, confidence
Event	Financial text	Keyword / pattern detection	Event type, confidence
Risk	Metrics, sentiment, events	Weighted risk scoring	Risk level, score, factors
Predictive	Historical prices	Prophet forecasting	Forecast, trend, interval
Comparative	Multi-company metrics	Normalisation, weighted scoring, ranking	Ranking, best performer
Checker	All agent outputs	Threshold and cross-agent checks	Validated outputs, confidence

11.5 Confidence and Safety Checker Design

The Checker is what separates FinAnalytics from a set of independent tools chained together. It receives every agent's output with its confidence and applies the staged procedure in Table 16, which implements the four validation layers of Section 7. Thresholds and weights are configurable. Their values will be calibrated during the evaluation described in Section 10.15, so this paper does not fix them.

Table 16 - Stages of the Confidence & Safety Checker.

Stage	What is checked	Result
Threshold validation	Each value against plausible ranges and a minimum agent confidence	Pass, reject, or flag
Consistency check	Signal pairs: sentiment–risk, forecast–risk, event–sentiment	List of conflicts
Reconciliation	Each conflicting pair	Both views kept; confidence lowered
Confidence scoring	Agent confidences and conflict fraction (cf. Eq. 12)	Per-result and overall confidence
Flagging	Scores below a configured cut-off; fallback data in use	Low-confidence flags in the report

Two behaviours are central. The Checker never quietly resolves a conflict in favour of one signal; the report shows both and explains why confidence dropped. Results based on fallback (mock) data are labelled, so a missing live feed lowers confidence and does not go unnoticed.

11.6 Processing Flow

Fig. 4 shows the end-to-end flow. The React frontend accepts the ticker and query and sends them to the Flask endpoint /api/analyze; input is validated and parsed; the Coordinator selects agents; agents retrieve and process data independently, with mock data as a fallback when live data is unavailable; the Checker validates and scores outputs; and the compiled report returns to the frontend through Axios for display, with optional PDF export.

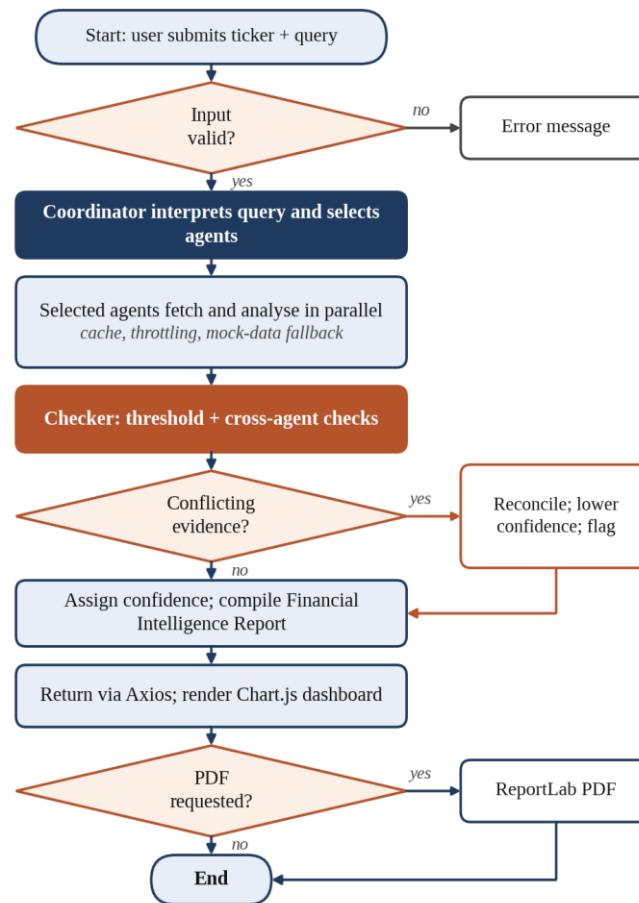


Fig. 4 - FinAnalytics system flowchart.

Viewed as messages between components, the flow is as follows. Once the query reaches the Flask API (steps 1–2), the Coordinator sends it to the selected agents (3). They fetch and analyse data (4) and pass their outputs to the Checker (5), which validates, reconciles, and scores them (6). The verified results go to the report generator (7), which returns a JSON report through Flask (8) for the dashboard to render (9). Five-minute caching and request throttling cut repeated calls to the data provider, and mock data keeps the system usable if a live feed fails.

11.7 Data Analysis

Sources. yfinance provides real-time and historical financial data, earnings-call text is used for sentiment and event analysis, SEC filings and financial news add supporting information, and mock data serves as a fallback. The current version supports eight selected companies. **Preprocessing** validates and parses the query, cleans the financial data and fills or handles missing values, and prepares text for DistilBERT sentiment and keyword-based event detection. At the context level (Level 0 of the data-flow analysis), the system has one user, who sends a ticker and query and receives a validated report, while market data and financial text arrive as external inputs.

The design leads to three data-handling rules. Market data is cached briefly so that repeated queries do not cause repeated provider calls. Text is cleaned and validated before it reaches the language model. Any result that depends on mock data is labelled. With these rules the data layer stays replaceable, and a different market-data provider or text source can be plugged in without changing the agents or the Checker.

The functional analysis takes the form of an input–process–output (IPO) model with two flows (Table 17): data ingestion and analysis (rows 1–9), and validation, modelling output, and reporting (rows 10–12).

Table 17 - IPO model of the FinAnalytics workflow.

Input	Process	Output
Ticker + natural-language query	JSON packaging → Flask POST → validation	Accepted company and query
Company and query	Company extraction and query parsing (Coordinator)	Analysis requirements
Analysis requirements	Metacognitive agent routing	Agents to invoke
Company ticker	yfinance retrieval with caching / throttling	Price, P/E, volatility, market cap

Input	Process	Output
Financial text	DistilBERT classification + fallback	Sentiment label, score, confidence
Financial text	Keyword / pattern event detection	Events with type and confidence
Metrics + sentiment + events	Weighted risk-scoring model	Risk level, score, factors
Historical price data	Prophet forecasting	Forecast, trend, confidence interval
Multi-company metrics	Normalisation + weighted scoring + ranking	Comparative ranking
All agent outputs	Threshold and cross-agent consistency check; reconciliation	Validated outputs and confidence
Validated outputs	Orchestration and report compilation	Unified JSON report
JSON / compiled report	Chart.js + React rendering; ReportLab rendering	Interactive dashboard; downloadable PDF

11.8 Non-Functional Requirements

Table 18 - Non-functional requirements and how they are met.

Requirement	Design provision
Security	Flask-CORS and controlled data processing protect data and control access
Reliability	Confidence & Safety Checker validates agent outputs before results are reported
Efficiency	Five-minute caching and request throttling reduce redundant API calls; independent agents can run in parallel
Performance	CPU-based DistilBERT and pre-loaded models reduce per-query processing time
Maintainability	Modular agents can be updated independently without affecting other modules
Scalability	Agent-based architecture lets new agents and companies be added with minimal redesign

11.9 Technical Requirements

Table 19 - Software requirements.

Technology	Version	Purpose	Note
Python	3.11+	Backend and ML/NLP	Model dependencies
JavaScript	ES6	Frontend	React components
Flask	2.3.2	Backend and REST API	Serves data and predictions
React	18.2.0	Dashboard UI	Interactive visualisation
Transformers	4.31.0	Hugging Face inference	Sentiment model loading
Torch	2.0.1	Deep-learning framework	Needed by Transformers
DistilBERT	SST-2 fine-tuned	Sentiment classification	Core sentiment model
TextBlob	Latest stable	Sentiment fallback	Basic backup check
yfinance	0.2.18	Financial data	Prices and company info
Prophet	1.1.5	Time-series forecasting	Historical trend predictions
Chart.js	4.3.0	Dashboard charts	All on-screen charts
ReportLab	Latest stable	PDF generation	Saves user reports

Table 20 - Hardware requirements.

Component	Minimum	Recommended	Purpose
Processor	Intel Core i5 / AMD Ryzen 5	Intel Core i7 / AMD Ryzen 7	Backend, agents, ML
RAM	8 GB	16 GB	DistilBERT and Prophet
Storage	500 MB free	1 GB+ free	Dependencies, model cache
Network	Stable internet	Broadband	yfinance, Hugging Face
OS	Windows 10/11, macOS, Linux	Latest stable	Flask and React

The React frontend, Flask backend, DistilBERT, and Prophet all run on the CPU of the application machine, with no dedicated GPU. The user device connects to it over the internet, and the machine calls external APIs for market data and models.

Requests travel from the user's browser over the network to the application machine, which calls the CPU-based models and the external data sources; no dedicated GPU is needed. Because the models live on the application machine, the user only needs a modern browser, and because external sources sit behind the backend, API keys and rate limits are managed in one place.

A note on the sentiment model. The DistilBERT model specified here is fine-tuned on SST-2, a general-domain sentiment corpus, and has not been trained on financial text. Given the domain mismatch discussed in Section 3, we accept this for CPU deployment but treat it as a known limitation. The TextBlob fallback and confidence reporting reduce its impact, and Section 13 lists a domain-tuned replacement as a priority.

11.10 Database Design

The data model (Table 21) centres on the *company* entity. Analysis results are stored per agent, and each final report links to a confidence-validation record.

Table 21 - Principal database entities.

Entity	Principal attributes
Company	company_id, ticker, name, sector
Financial data	price, P/E ratio, volatility, market cap, timestamp
Financial text	source type, content, timestamp
Sentiment result	label, score, confidence
Detected event	event type, confidence
Risk assessment	risk level, risk score, factors
Prediction	forecast, trend, confidence interval
Comparative analysis	company ids, ranking, best performer
Agent result	agent name, raw output, confidence
Analysis report	query, generated at, PDF path
Confidence validation	overall confidence, conflicts resolved

Fig. 5 gives the logical entity-relationship view. Company is the central entity: financial data, financial text, risk assessments, predictions, agent results, and reports reference it through *company_id*, while sentiment results and detected events reference financial text through *text_id*, and the confidence-validation record references its report.

11.11 User Interface Design

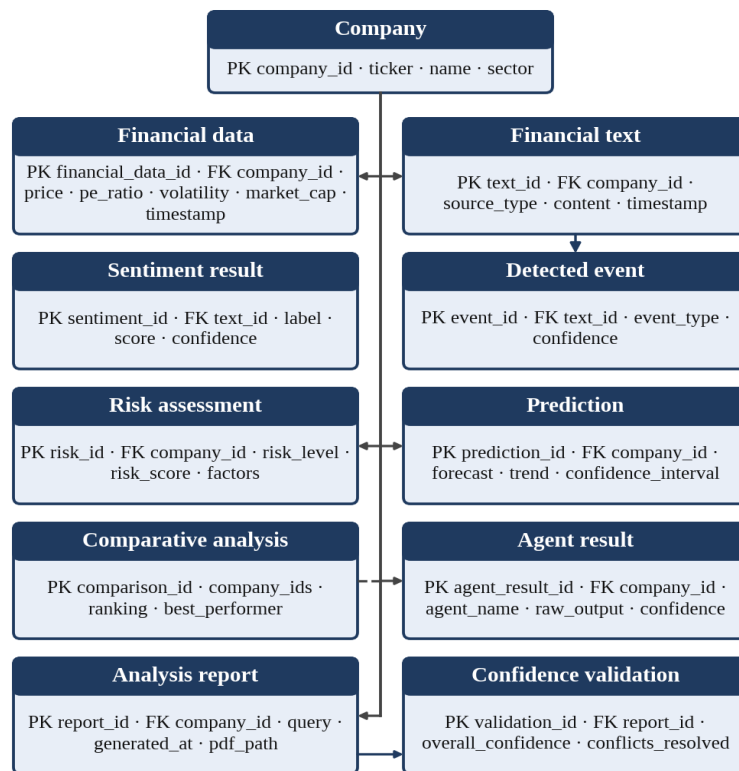


Fig. 5 - Logical entity-relationship diagram (key attributes).

The interface is a React dashboard with three views. The home view takes a company or ticker and a natural-language query, offers shortcuts to performance, sentiment, risk, and event analyses, and lists recent analyses. The analysis dashboard shows cards for price, valuation, volatility, sentiment, risk, detected events, and a forecast with its uncertainty band, next to an overall confidence meter. The report view gives a comparison across companies, key findings, guidance worded as decision support and not investment advice, the sources used, and a button to download the PDF report.

Feasibility

Table 22 - Feasibility assessment.

Type	Assessment
Technical	Established technologies (Python, Flask, React, Hugging Face, yfinance, Prophet, ReportLab, Chart.js); lightweight DistilBERT runs on CPU; caching, throttling, and pre-loading improve response time
Economic	Open-source software and pretrained models; free or free-tier data; no paid APIs or specialised infrastructure for the current scope; student-accessible hardware
Operational	Simple dashboard needing only a ticker and query; automates several analyses in one platform; provides confidence-scored results, visualisations, and PDF reports
Legal	Decision-support and educational tool, not investment advice; confidence scores and source-backed findings for transparency; forecasts carry intervals and trend direction; data not stored beyond required caching

Project Plan

The project runs on a 12-week plan in three phases (Fig. 6): initiation (weeks 1–3), planning and requirements (weeks 4–7), and design and development (weeks 8–12). Development moves from requirements and planning through backend and financial data, sentiment and event analysis, risk, prediction and comparison, coordinator and validation, dashboard and visualisation, and PDF reporting, ending with testing and documentation.

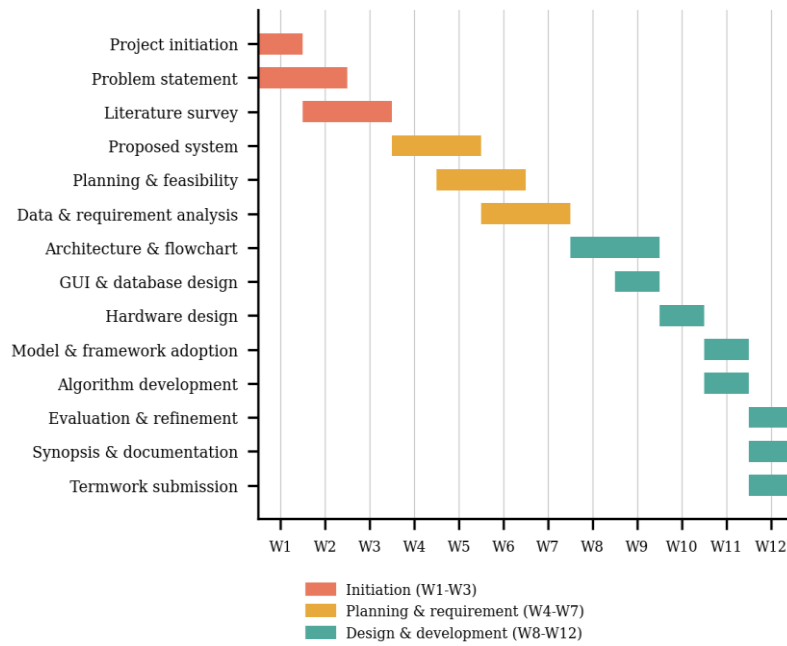


Fig. 6 - Twelve-week project plan.

Gap-to-Design Mapping

Table 23 - Research gaps and the FinAnalytics response.

Gap	FinAnalytics design response
G1	Single platform with six specialised agents behind one query interface
G2	Confidence & Safety Checker: threshold and cross-agent consistency checks, confidence scores, low-confidence flags
G3	Risk Agent combining financial metrics, sentiment, and events
G4	Prophet forecasts with confidence intervals and trend direction
G5	Source-backed report with confidence levels, charts, and PDF export
G6	CPU-based DistilBERT, free-tier data, caching and throttling; no paid APIs

Proposed Evaluation Plan

Table 24 lists the experiments we propose for validating the design. They use the metrics of Section 8 and avoid the pitfalls listed in Section 8.2. This paper does *not yet report* results.

Table 24 - Proposed evaluation plan.

Component	Experiment	Metric / baseline
Sentiment	Evaluate on Financial PhraseBank [14]; compare DistilBERT with the lexicon fallback	Accuracy, macro-F1
Forecasting	Time-ordered backtest on the eight supported companies	RMSE, MAPE vs. ARIMA [24]; interval coverage
Validation	Inject cross-signal conflicts (Table 11); run with and without the Checker	Conflict-detection rate, false flags
Confidence	Compare reported confidence with observed correctness	Calibration (ECE)
Efficiency	Measure end-to-end latency with and without caching	Latency, cache hit rate

Testing also follows the architecture. Each agent is tested alone on fixed inputs, the /api/analyze path is tested end to end, fallback behaviour is tested by switching off the live data feed, and the dashboard and PDF output are checked against the validated report. Checker tests should include inputs that conflict on purpose, so that its flags are verified directly.

Two comparisons are likely to be the most informative. The first is an ablation in which the same queries are run with and without the Checker, and the reports are compared on whether conflicts were surfaced and how closely stated confidence follows correctness. The second uses seeded conflicts: an input is altered on purpose (for example, strongly positive text paired with poor metrics), so the expected flag is known beforehand and the detection rate and false-flag rate can be measured directly. Neither comparison depends on any claim about investment returns.

12. Emerging Applications

Education and research. Students and researchers can study how sentiment, events, fundamentals, and forecasts relate for one company in a single place, with sources and confidence shown, without moving between tools. **Retail and individual investors.** A natural-language interface makes multi-signal analysis easier to access, as long as outputs are presented as decision support and not as advice. **Analyst workflows.** Automatic first-pass summaries and multi-company comparisons can save manual work and bring conflicting evidence to light early. **Report generation.** Downloadable reports that cite their sources keep an audit trail of what the system concluded and how confident it was.

Teaching and demonstration. Each agent is modular and its output is visible, so the platform can be used in class to show coordination, validation, and uncertainty. **Research testbed.** Researchers can replace one component, such as the sentiment model with a domain-tuned one, and observe the effect on conflict detection and calibration without rebuilding the rest of the system.

13. Challenges and Ethical Considerations

Market unpredictability: prices are noisy and non-stationary, so forecasts should be shown as scenarios with uncertainty and never as guarantees.

Data quality and availability: free data feeds may be delayed, incomplete, or rate-limited, which makes fallbacks and staleness warnings necessary.

Model limitations: lightweight sentiment models and keyword-based event rules give up accuracy for speed, and LLM-style components can make unsupported statements unless they are grounded in sources [20].

Legal and ethical use: these systems are decision-support and teaching tools and do not give investment advice. Outputs should state their confidence and sources, and users need to understand the limits.

Privacy and data governance: keep data only as long as needed (short-lived caches, for example) and follow each data provider's terms.

Over-reliance: high-confidence displays can lead users to trust results uncritically, so low-confidence results should be flagged clearly.

Latency and resource limits: transformer and forecasting models running on CPU limit throughput, so caching, pre-loaded models, and request throttling are needed for acceptable interactive response times.

Bias and coverage: a small set of companies and English-only sources can skew results, and the coverage limits should be stated.

Reproducibility: free data feeds change over time and model versions get updated, so each analysis should record the data retrieval time and library versions needed to reproduce it.

Explanation gap: showing sources and flags helps, yet a fluent summary can still sound more certain than the evidence allows, so the wording should match the confidence level.

14. Future Directions

The directions below are qualitative and are not ordered by expected impact. We avoid numeric projections on purpose.

Short term: swap general sentiment encoders for domain-tuned financial models [1], [13], add learned event classifiers trained with weak supervision [3], and carry out the evaluation plan of Section 10.15.

Medium term: retrieval over filings and transcripts with citations [20], learned confidence calibration [31], backtesting of forecasts and rankings, and extension to more companies and to Indian markets (NSE/BSE).

Long term: portfolio-level reasoning over agent outputs, along the lines of [9], reflection-style self-correction [21], and human review of low-confidence results.

Product maturity: saved analysis history, more companies, and user-adjustable weights for the risk and ranking models, with each change documented so results stay explainable.

Cross-cutting: larger shared benchmarks for coordinated financial analysis, so systems can be compared on conflict handling and calibration as well as accuracy.

Coverage: support more companies and add regional-language and non-English sources, lowering confidence where the data is thinner.

15. Best Practices

For researchers: evaluate on shared datasets and report baselines; use several metrics (Section 8); document data sources, time windows, and preprocessing; split data by time; run ablation studies; and analyse failure cases such as conflicting signals.

For practitioners: keep agents modular; cache and throttle external calls; show confidence and sources with every result; provide fallback data paths; test thresholds on realistic conflicts; and make clear that outputs support professional judgement and do not replace it.

For educators and students: use open data and reproducible environments, pin library versions, and use forecast outputs to teach uncertainty, not as trading signals.

For system designers: keep the Checker independent of any single agent, log every flag and confidence value so behaviour can be audited, and version thresholds and weights as configuration.

16. Limitations of This Survey

The survey is deliberately narrow. It reviews ten core studies and their supporting works and does not cover financial AI as a whole. The capability matrix reflects each study's main focus as we read it; we did not re-implement the studies. Their data and metrics differ, so we do not compare numeric results across them. The FinAnalytics evaluation plan has not been carried out yet, and the claimed benefits are design arguments until it is.

Two more caveats apply. Table 13 records each study's main focus as its authors describe it, and a study may implement more than it emphasises. The proposed platform is a design, and its benefits are hypotheses still to be tested. The gap analysis is best read as a map of where evidence is thin, and it should not be taken as a ranking of the reviewed studies.

17. Conclusion

Financial NLP, forecasting, risk modelling, multi-criteria ranking, and LLM agents have each matured, yet the reviewed literature mostly treats them separately. Of the nine technical studies, one covers multiple roles through agents and none offers cross-component validation as a primary contribution, so analysts are left to reconcile conflicting signals by hand.

We proposed a five-dimensional taxonomy, a capability matrix, and six research gaps, and presented FinAnalytics, a coordinator-routed multi-agent platform with a Confidence & Safety Checker, as a design that addresses these gaps with open-source tools on modest hardware. The next steps are a careful evaluation of each agent and of the validation layer, then richer event extraction, retrieval over filings, and calibrated confidence. The goal is clear, explainable analysis, tied to its sources, that supports human decision-making.

Acknowledgements

We thank our internal guide, Dr. K. T. Patil, and the Department of Computer Engineering, Smt. Indira Gandhi College of Engineering, for their guidance and support.

References

- [1] Z. Liu, D. Huang, K. Huang, Z. Li, and J. Zhao, "FinBERT: A Pre-trained Financial Language Representation Model for Financial Text Mining," in *Proc. IJCAI*, 2020.
- [2] Y. Xiao, E. Sun, D. Luo, and W. Wang, "TradingAgents: Multi-Agents LLM Financial Trading Framework," MARW Workshop, AAAI, 2025.
- [3] L. Ein-Dor *et al.*, "Financial Event Extraction Using Wikipedia-Based Weak Supervision," in *Proc. EACL, ACL*, 2019, pp. 10–15.
- [4] S. Gite, H. Khatavkar, K. Kotecha, S. Srivastava, P. Maheshwari, and N. Pandey, "Explainable Stock Prices Prediction from Financial News Articles Using Sentiment Analysis," *PeerJ Computer Science*, vol. 7, 2021.
- [5] T. Sidogi, R. Mbuva, and T. Marwala, "Stock Price Prediction Using Sentiment Analysis," in *Proc. IEEE SMC*, 2021, pp. 46–51.
- [6] Y. Sun and J. Li, "Deep Learning for Intelligent Assessment of Financial Investment Risk Prediction," *Computational Intelligence and Neuroscience*, 2022, Art. 3062566.
- [7] K. F. Liew, W. S. Lam, and W. H. Lam, "Financial Network Analysis on the Performance of Companies Using Integrated Entropy–DEMATEL–TOPSIS Model," *Entropy*, vol. 24, no. 8, p. 1056, 2022.

-
- [8] A. Garlapati *et al.*, "Stock Price Prediction Using Facebook Prophet and ARIMA Models," in *Proc. 6th Int. Conf. Convergence in Technology (I2CT)*, IEEE, 2021, pp. 1–7.
- [9] M. Muthivhi and T. L. van Zyl, "Fusion of Sentiment and Asset Price Predictions for Portfolio Optimization," in *Proc. 25th Int. Conf. Information Fusion (FUSION)*, IEEE, 2022, pp. 1–8.
- [10] K. Du, Y. Zhao, R. Mao, F. Xing, and E. Cambria, "Natural Language Processing in Finance: A Survey," *Information Fusion*, vol. 115, 102755, 2025.
- [11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2019.
- [12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv:1910.01108, 2019.
- [13] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," arXiv:1908.10063, 2019.
- [14] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, "Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts," *J. Assoc. Inf. Sci. Technol.*, vol. 65, no. 4, 2014.
- [15] S. J. Taylor and B. Letham, "Forecasting at Scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.
- [16] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [17] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?: Explaining the Predictions of Any Classifier," in *Proc. ACM KDD*, 2016.
- [18] S. Wu *et al.*, "BloombergGPT: A Large Language Model for Finance," arXiv:2303.17564, 2023.
- [19] H. Yang, X.-Y. Liu, and C. D. Wang, "FinGPT: Open-Source Financial Large Language Models," arXiv:2306.06031, 2023.
- [20] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Proc. NeurIPS*, 2020.
- [21] N. Shinn *et al.*, "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Proc. NeurIPS*, 2023.
- [22] Q. Wu *et al.*, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," arXiv:2308.08155, 2023.
- [23] S. Hong *et al.*, "MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework," in *Proc. ICLR*, 2024.
- [24] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. Wiley.
- [25] C.-L. Hwang and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications*. Springer, 1981.
- [26] W. F. Sharpe, "The Sharpe Ratio," *J. Portfolio Management*, vol. 21, no. 1, pp. 49–58, 1994.
- [27] A. Vaswani *et al.*, "Attention Is All You Need," in *Proc. NIPS*, 2017.
- [28] L. Loukas *et al.*, "EDGAR-CORPUS: Billions of Tokens Make the World Go Round," in *Proc. EONLP*, 2021.
- [29] M. Maia *et al.*, "WWW'18 Open Challenge: Financial Opinion Mining and Question Answering," in *Companion Proc. The Web Conf.*, 2018.
- [30] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Proc. NeurIPS*, 2022.
- [31] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *Proc. ICML*, 2017.
- [32] J. H. Flavell, "Metacognition and Cognitive Monitoring: A New Area of Cognitive–Developmental Inquiry," *American Psychologist*, vol. 34, no. 10, pp. 906–911, 1979.
- [33] Y. Li, S. Wang, H. Ding, and H. Chen, "Large Language Models in Finance: A Survey," in *Proc. 4th ACM Int. Conf. AI in Finance (ICAIF)*, 2023.
- [34] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005–2019," *Applied Soft Computing*, vol. 90, 2020.
- [35] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo, "Text Mining for Market Prediction: A Systematic Review," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7653–7670, 2014.
- [36] Z. Xi *et al.*, "The Rise and Potential of Large Language Model Based Agents: A Survey," arXiv:2309.07864, 2023.
- [37] L. Wang *et al.*, "A Survey on Large Language Model based Autonomous Agents," *Frontiers of Computer Science*, vol. 18, 2024 (arXiv:2308.11432).
- [38] W. X. Zhao *et al.*, "A Survey of Large Language Models," arXiv:2303.18223, 2023.
- [39] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.
- [40] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models," in *Proc. ICLR*, 2023.

[41] E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work," *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[42] T. Loughran and B. McDonald, "When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks," *The Journal of Finance*, vol. 66, no. 1, pp. 35–65, 2011.