KDD 2026 TUTORIAL · JEJU, KOREA

The Anatomy of a Systematic Trading Strategy

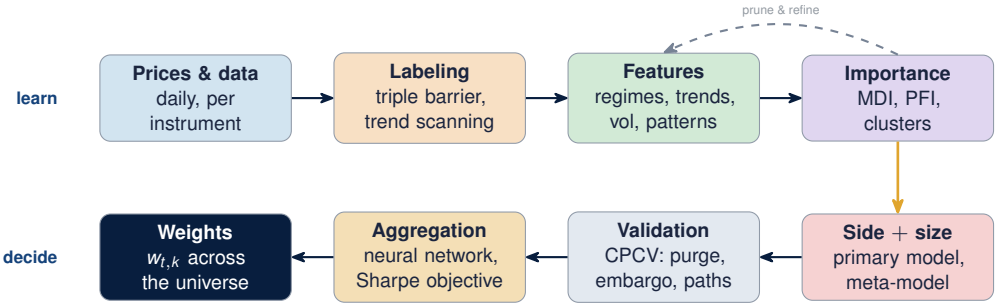
From raw prices to a live futures strategy:

Labels • Features • Importance • Side & Size • Portfolio Networks

Hachem Madmoun • Syllogia • Imperial College Business School • MBZUAI

Outline

The pipeline in one picture



Section 1 walks this diagram, one subsection per stage: this tutorial condenses BUSI70575, **Systematic Trading Strategies with Machine Learning Algorithms**, at Imperial College Business School.
Section 2 presents the implementation: **the Hendaye strategy**, running on a real universe at Alken Asset Management.

SECTION 01

The Systematic Trading Strategy Pipeline

01

Where we are

Labeling defines the learning task

Supervised learning learns $f : X \to Y$ from examples. In trading, **nobody hands us** Y: we manufacture it, and that choice *is* the investment hypothesis.

- The label determines which features are informative: a feature irrelevant under one labeling scheme can be predictive under another.
- Understanding that link is exactly the job of **feature importance analysis** (Subsection 1.3).

Three labeling methods, this subsection

Fixed-time horizon: the common default, and its flaws.

Triple barrier: risk management encoded in the label.

Trend scanning: the data chooses the horizon.

Course anchor

Lecture 1 of BUSI70575, with the trend-scanning programming session.

López de Prado (2018), Ch. 3; BUSI70575, Lecture 1.

Fixed-time horizon: the default, and its two flaws

Grade a position by its return at a single future instant $t + h$:

$$y_i = \begin{cases} +1 & r_{t,\,t+h} > \tau \\ 0 & |r_{t,\,t+h}| \le \tau \\ -1 & r_{t,\,t+h} < -\tau \end{cases} \qquad r_{t,\,t+h} = \frac{p_{t+h}}{p_t} - 1$$

with horizon h and threshold τ fixed in advance.

Two structural flaws

Volatility-blind. The same τ in calm and stressed regimes: a 1% move does not mean the same thing, yet gets the same label.

Path-blind. Only the endpoint matters: a trade stopped out on day 2 and one that sailed to the target can get the same label.

The fixes

Volatility-adjusted thresholds, and **path-aware** labels, next.

The triple-barrier method: risk management inside the label

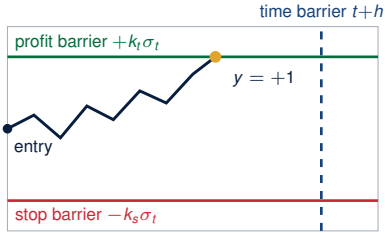
Around each entry, place **three barriers** and label by the **first touch**:

- **Profit-taking** at $+k_t\,\sigma_t \;\Rightarrow\; y = +1$
- **Stop-loss** at $-k_s\,\sigma_t \;\Rightarrow\; y = -1$
- **Time barrier** at $t + h \;\Rightarrow\; y = 0$ or sign of return

Widths scale with local volatility σ_t, so a label means the **same statistical event** in every regime.

Why it matters

The label now encodes **stop-loss discipline and holding limits**: the way the position would actually be managed. It records the first-touch time $t_{i,1}$ too.



The same downward move is a *win* for a short and a *loss* for a long: labels are defined relative to the side of the trade.

Trend scanning: let the data choose the horizon

For each date t, fit a linear trend over a **forward window** of length H,

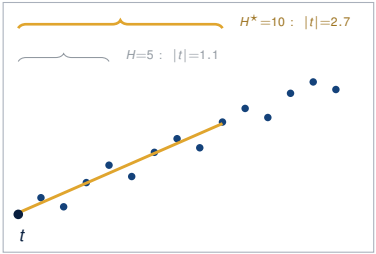
$$x_{t+h} = \beta_0 + \beta_1 h + \varepsilon_{t+h}, \qquad \hat{t}_{\hat{\beta}_1} = \frac{\hat{\beta}_1}{\hat{\sigma}_{\hat{\beta}_1}},$$

compute the t-statistic of the slope for each candidate H, and keep

$$H^\star = \arg\max_H \left|\hat{t}_{\hat{\beta}_1}(H)\right|.$$

The label is the **sign of the most statistically significant trend**.

- No barriers, no fixed horizon: the trend picks its own length: data-driven, no arbitrary thresholds.
- Hold this thought: run **backwards**, the same statistic becomes a *feature* (Subsection 1.2).



Different look-forward windows give different t-stats; the most significant one defines the label.

Which label for which job

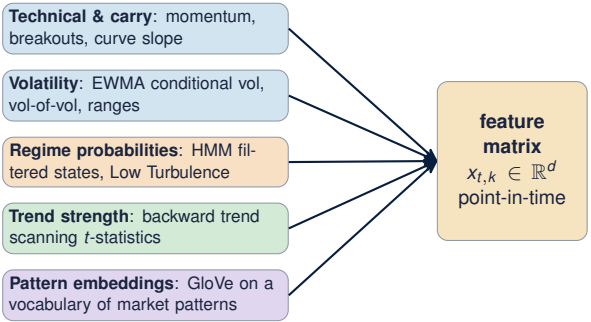
Method	What the label encodes	Typical use
Fixed horizon	return at one instant, fixed τ	quick baselines; papers
Triple barrier	the *outcome of a managed trade*: target, stop, time-out, vol-scaled	grading a strategy's trades $\rightarrow$ **meta-labels** (Subsection 1.4)
Trend scanning	presence and significance of a trend, horizon chosen by the data	directional labels for a **primary model**; backward version as **features** (Subsection 1.2)

The idea to carry forward

A label grades *something*. Grade the **market** and you learn direction. Grade **your own trades** and you learn when to bet. Both appear again in Subsection 1.4.

Where we are

The feature library



This subsection builds the three **model-based** families, the ones the course spends most time on:

- **HMM regime probabilities**,
- **backward trend scanning**,
- **GloVe pattern embeddings**.

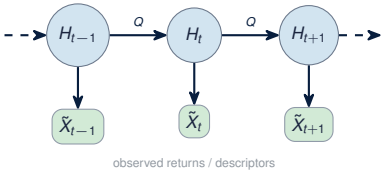
One rule for every column

Computable **at the time**, from past data only, and admitted with an economic rationale.

Regime features: hidden Markov models

Latent states $H_1, \ldots, H_T$ with M possible regimes; observations $\tilde{X}_1, \ldots, \tilde{X}_T$ [10].

- **Markov property**: the hidden states form a first-order Markov chain.
- **Emission independence**: given H_t, the observation $\tilde{X}_t$ is independent of everything else.



Why regimes as features

Markets alternate between persistent environments; a directional model behaves differently across them. The HMM turns that structure into **numbers a model can read**.

Three problems, three tools

Inference (how likely is what we saw?), **learning** (fit the parameters), **prediction** (which regime comes next?). The next slides take them in turn.

BUSI70575, Lecture 3.

Parameterization

Discrete emissions

Observation space $\mathcal{O} = \{o_1, \ldots, o_D\}$.

Initial distribution: $\pi_m = P(H_1 = m)$

Transition matrix: $Q_{ij} = P(H_{t+1} = j \mid H_t = i)$

Emission matrix: $O_{mj} = P(\tilde{X}_t = o_j \mid H_t = m)$

Parameter set: $\theta = (\pi,\ Q,\ O)$

Continuous emissions

One Gaussian per regime:

$$\tilde{X}_t \mid H_t = m \ \sim\ \mathcal{N}_d(\mu_m, \Sigma_m)$$

Parameter set: $\theta = (\pi,\ Q,\ \mu,\ \Sigma)$

This is the version we run on return descriptors: each regime is a cloud with its own mean and covariance.

Both share the **same hidden-state machinery** (π and Q); only the emission model changes. To build intuition we first work the discrete case on a small example, then return to markets.

A worked example: who ate the sandwich?

Dr. Ross Geller, a paleontologist, brings a sandwich every day and stores it in the department refrigerator. It **frequently disappears**. He records the only thing he can observe:

- 0: sandwich is safe • 1: sandwich is missing

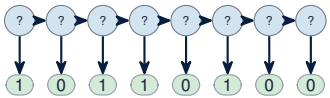
Hidden states (who was around, unobserved):

- State 0: Dr. Donald is present, 90% chance the sandwich is eaten
- State 1: Dr. Charlie is present, 50% chance
- State 2: neither is present, 0% chance

Emission matrix (rows = states, columns = safe/missing):

$$O = \begin{bmatrix} 0.1 & 0.9 \\ 0.5 & 0.5 \\ 1.0 & 0.0 \end{bmatrix}$$

hidden: the visitor sequence



observed: safe / missing, day by day

BUSI70575, Lecture 3.

The example's hidden dynamics

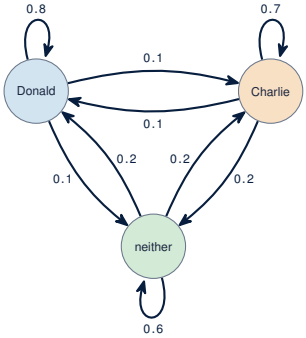
Initial distribution (uniform):

$$\pi = \left[\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right]$$

Transition matrix:

$$Q = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$

The diagonal dominates: **visitors persist**, exactly like market regimes. Sticky dynamics are what make tomorrow predictable from today.



The three questions, concretely

How likely is the observed pattern of missing sandwiches (**inference**)? What are π, Q, O (**learning**)? And, given everything so far: who is around **tomorrow** (**prediction**)?

Learning the parameters: EM

The likelihood sums over **every hidden path**:

$$p_\theta(\tilde{\mathbf{x}}) = \sum_{h_1} \cdots \sum_{h_T} \pi_{h_1} \prod_{t=1}^{T-1} Q_{h_t, h_{t+1}} \prod_{t=1}^{T} p\big(\tilde{x}_t \mid h_t\big)$$

M^T paths: intractable directly. The **Forward algorithm** computes it recursively in $O(TM^2)$.

The E-step, in practice

The posteriors over hidden states, the smoothing probabilities $\psi(t, h)$ and $\phi(t, h, h')$, are computed by the **Forward–Backward algorithm**; the M-step then updates (π, Q, μ, Σ) in closed form.

Algorithm 1 EM for HMMs

Require: observations $\tilde{\mathbf{x}} = \{\tilde{x}_1, \ldots, \tilde{x}_T\}$
Ensure: parameters θ
1: initialise $\theta^{(0)}$
2: **while** not converged **do**
3: **E-step:** $q_{i+1} \in \arg\max_q \mathcal{L}\big(q, \theta_i\big)$
4: **M-step:** $\theta_{i+1} \in \arg\max_\theta \mathcal{L}\big(q_{i+1}, \theta\big)$
5: **end while**
6: **return** $\theta^\star$

Alternating maximisation of the same lower bound $\mathcal{L}(q, \theta)$: each step can only improve the likelihood.

Predicting the next hidden state from past observations

The **filtering probabilities** are the causal beliefs about the current regime:

$$\xi(t,h) := p\big(H_t = h \mid \tilde{X}_1, \ldots, \tilde{X}_t\big),$$

computed by the Forward algorithm, using **only information up to** t. Pushing them through the transition matrix gives tomorrow's regime distribution:

$$p\big(H_{T+1} = h \mid \tilde{X}_{1:T}\big) = \sum_{h'=1}^{M} \underbrace{Q_{h'h}}_{\text{transition}} \underbrace{\xi(T,h')}_{\text{filtered belief}}$$

In the example

The probability that Dr. Donald shows up **tomorrow**, given every safe/missing outcome so far.

In markets

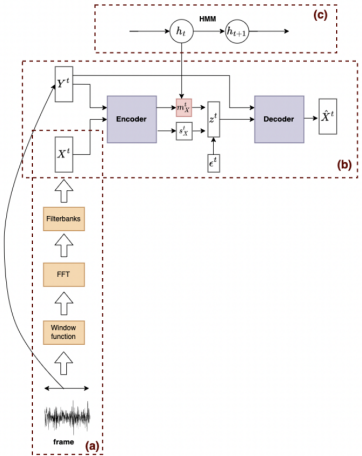
The probability of **each regime tomorrow**, given returns up to today. Strictly ex ante, one number per state per day: exactly what a **feature column** must be.

The Low Turbulence Model

Our production regime model [7] applies this machinery to the **cross-section of returns**:

- **(a) Time–frequency analysis** [8]: each return frame passes through a window function, an FFT and filterbanks, giving **spectral vectors** X^t.
- **(b) Compression** [5]: a variational autoencoder maps X^t to **latent spectral vectors** z^t.
- **(c) Regime detection** [10]: an **HMM** on the latent sequence $h_t \to h_{t+1}$ names the regimes.

Output: the probability of a **turbulent regime**, one causal number per day, which enters the feature matrix like any other column, and Subsection 1.4 shows which model reads it.



Recall: trend scanning looked forward

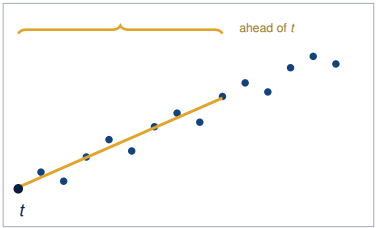
In the labeling subsection, trend scanning fit a regression over a **forward** window, computed the slope's t-statistic $\hat{t}_{\hat{\beta}_1}(H)$ for each candidate H, and kept

$$H^\star = \arg\max_H \left|\hat{t}_{\hat{\beta}_1}(H)\right|,$$

labeling date t with the sign of the most significant **future** trend.

The question

The statistic clearly measures "trendiness". But as written it peeks into the future: usable as a label, forbidden as a feature. How do we reuse the same idea **causally**, so it is available at time t?



The label machinery: regression windows starting at t and reaching forward. The next slide flips the direction.

Trend features: trend scanning, run backwards

Algorithm 2 Backward trend scanning (features at date t)

Require: series $\{x_s\}_{s \le t}$, candidate lengths $\mathcal{L} = \{L_{\min}, \ldots, L_{\max}\}$
Ensure: trend features at date t
for $L \in \mathcal{L}$ **do**
  regress $x_{t-L+1}, \ldots, x_t$ on time: slope $\hat{\beta}_1(L)$
  $\hat{t}(L) \leftarrow \hat{\beta}_1(L) \,/\, \hat{\sigma}_{\hat{\beta}_1}(L)$ ▷ t-stat of the slope
end for
$L^\star \leftarrow \arg\max_{L \in \mathcal{L}} \left|\hat{t}(L)\right|$
return $\left(\hat{t}(L^\star),\ L^\star,\ R^2(L^\star)\right)$

The **mirror image** of the labeling method: same regression, same statistic, but the window looks **back**, so the output is **causal**: a feature, not a label.

- $\hat{t}(L^\star)$: signed **trend strength**: how confident the data is that a trend exists now.
- $L^\star$: the **length** of the dominant trend.
- $R^2(L^\star)$: its **quality**: trendiness vs. noise.

Why it earns its place

It is adaptive (no fixed lookback) and statistically grounded (no arbitrary threshold): the two flaws of standard momentum features, removed.

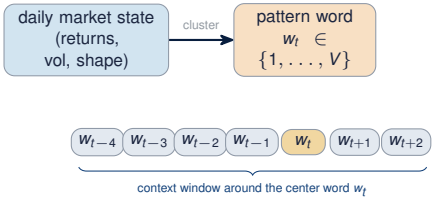
Pattern features I: a vocabulary for markets

Word embeddings rest on one intuition: *a word's meaning is given by the words that frequently appear close by*. We transpose it: **a market pattern's meaning is given by the patterns that tend to surround it in time**.

- **Build the vocabulary**: discretise each day's market state (returns, volatility, shape descriptors) into one of V **pattern words**, e.g. by clustering.
- **Sentences**: the history becomes a sequence of pattern words, one per day.
- **Co-occurrence matrix** $X \in \mathcal{M}_{V,V}(\mathbb{R})$: when word $w[j]$ appears in the context window of $w[i]$, update

$$X[w[i], w[j]] \;+\!=\; \frac{1}{|i-j|},$$

so **closer context counts more**, and the context-size choice matters less.



Since the non-zero counts are large, we work with $\log X$ (after adding 1), which stays sparse.

BUSI70575 optional session; Pennington et al. (2014).

Pattern features II: the GloVe factorization

Approximate the log co-occurrences with a **matrix factorization** [9]: for all (i, j),

$$\log X_{ij} \;\approx\; W_i^\top \tilde{W}_j + b_i + \tilde{b}_j,$$

with embeddings $W, \tilde{W} \in \mathcal{M}_{V,D}(\mathbb{R})$ and biases $b, \tilde{b} \in \mathbb{R}^V$: $2(VD + V)$ parameters in total. Rare co-occurrences are noisy, so weight each term:

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \big(\log X_{ij} - W_i^\top \tilde{W}_j - b_i - \tilde{b}_j \big)^2$$

Trained by **gradient descent** or **alternating least squares** (both derived in the course session).

From embeddings to features

Each pattern word i gets a vector $W_i \in \mathbb{R}^D$; day t inherits the embedding of its word, W_{w_t} (or a context average). D new columns that encode **which environments tend to follow which**: structure no single indicator sees.

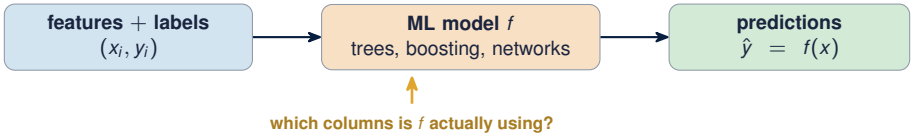
Why factorize, not window-predict

Unlike shallow window-based models (word2vec), GloVe operates **directly on the global co-occurrence statistics**: the whole history speaks at once.

Where we are

We can now train a model. What is it using?

We have a feature matrix x and labels y: any supervised learner fits the map, whether a tree ensemble, gradient boosting, or a neural network.



The question this subsection answers

Is the model relying on **economically meaningful structure**, or on artifacts it memorised? And which feature families **earn their place**, feeding the prune-and-refine loop of the pipeline?

Two complementary methods

MDI, computed inside tree models during training (in-sample), and **PFI**, computed on held-out data for **any** model (out-of-sample). Each answers the question from a different side.

Mean Decrease Impurity (MDI): the in-sample view

Algorithm 3 MDI for a random forest [2]

Require: trained forest with T trees
Ensure: importance scores $\{\mathrm{MDI}_j\}_{j=1}^{d}$

$\mathrm{IMP}_j \leftarrow 0$ for all features j
for each tree $t = 1, \ldots, T$ **do**
  for each internal node n of tree t **do**
    $j \leftarrow$ feature split at n; $w_n \leftarrow$ share of samples at n
    $\mathrm{IMP}_j \leftarrow \mathrm{IMP}_j + w_n \cdot IG_n$ ▷ weighted information gain
  end for
end for
return $\mathrm{MDI}_j = \mathrm{IMP}_j \big/ \sum_k \mathrm{IMP}_k$

Pros

Fast, computed during training; the default in most libraries; normalised, interpretable scores.

Cons

In-sample only: it can assign importance to noise the model merely memorised; biased toward high-cardinality features.

Use it as a **first look**, never as the verdict.

BUSI70575, Lecture 2.

Permutation Feature Importance (PFI): the out-of-sample view

Algorithm 4 Permutation feature importance

Require: fitted model m, validation data D, repetitions K
Ensure: $\{\mathrm{PFI}_j\}$ with standard deviations $\{\sigma_j\}$
$s \leftarrow$ score of m on D ▷ reference performance
for each feature j **do**
  for $k = 1, \ldots, K$ **do**
    shuffle column j of $D \rightarrow$ corrupted $\tilde{D}_{k,j}$
    $\mathrm{scores}_j[k] \leftarrow s - \mathrm{score}\big(m, \tilde{D}_{k,j}\big)$
  end for
  $\mathrm{PFI}_j,\ \sigma_j \leftarrow$ mean, std of scores_j
end for

Pros

Out-of-sample, evaluated on validation data; **model-agnostic**; works with any performance metric.

Cons

Computationally expensive; varies across permutations (hence the K repetitions and σ_j); and sensitive to feature correlation: the next slide's problem.

It reads the causality of the **model**, not of the market.

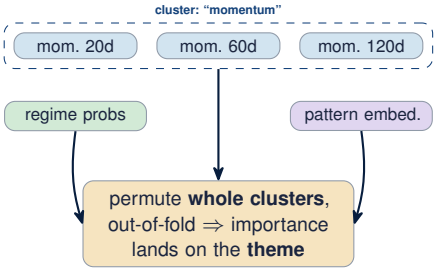
The substitution effect, and the cluster-level fix

The correlation problem

Permute one of two correlated features and its twin covers for it: the performance drop is small, and *both* look unimportant, even when jointly they matter. With momentum computed at five horizons, all five can rank below a mediocre standalone feature. Financial features are highly correlated, so this is not a corner case; it is the default.

Mitigations

Clustering: group correlated features and compute importance at the **cluster level**: permute whole clusters [4].
Dimensionality reduction: PCA or autoencoders to extract uncorrelated components first.



Themes, not columns: the level at which economics actually speaks, and the level we report in production.

BUSI70575, Lecture 2.

Where we are

Why meta-models: two questions, two models

A primary model (trend-following, mean-reversion, or an expert) performs well only in **specific market contexts**, and regimes evolve. Rather than ask one model to do everything, **decouple** [3]:

Which way? (the **side**)
the *primary model* predicts direction: trend detection, mean reversion, an expert's calls

Act on it, and how much? (the **size**)
the *meta-model* learns the conditions under
which the primary's signal can be trusted

- The meta-model can **filter** trades in regimes where the primary underperforms, or **scale** exposure with confidence.
- It is not predicting the market; it is predicting **the performance of a model**. Think of it as a **regime detector on the primary's skill**.
- Applicable to **any** directional model or human expert, which is what makes the architecture reusable.

Meta-labels: grading the primary's trades

Primary signal
side $s_i \in \{-1, +1\}$
at time t_i

→

Triple barrier
on the trade, with
the *side applied*

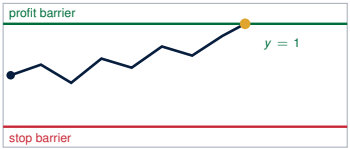
→

Meta-label
$y_i = \mathbf{1}\{\text{profit barrier hit first}\} \in \{0, 1\}$

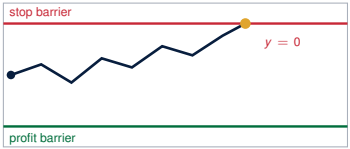
→

Meta-model
$\hat{p}_i = \mathbb{P}(y_i{=}1 \mid x_i)$

primary $+1$ (long)



primary -1 (short)



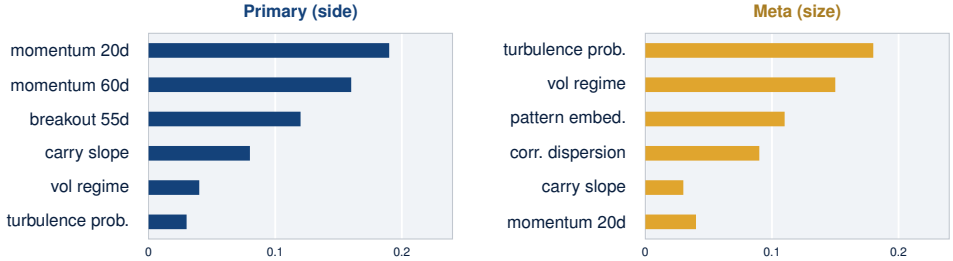
The **same price path**, opposite outcomes: with the side applied, the two barriers **switch roles**. Meta-labels grade the **trade**, not the market, so a long and a short over the same move get opposite labels.

The reframing

A hard 3-class market prediction becomes an **easy binary classification**, "was this signal worth following?", with a probability output.

Different jobs read different features

The primary and the meta-model get **separate feature sets**; using the same ones risks learning redundant information. Cluster-level importance (Subsection 1.3), computed separately, confirms the division:



The side model reads **trend and momentum**; the meta-model reads **regime descriptors**: turbulence probability, volatility state, pattern embeddings. *(Illustrative magnitudes; the split is the point.)*

Evaluating the pair against the primary

Trading recall for precision

The primary alone takes **every** signal: recall $= 1$, precision $=$ base rate. The meta-model rejects low-confidence trades: it gives up recall to gain **precision**, and the trade-off is steered by the decision threshold, with F1 as the balance.

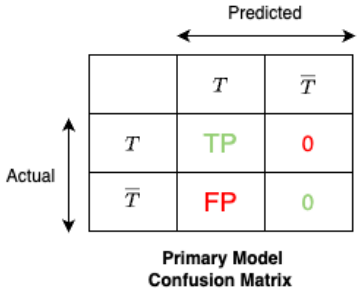
The only fair comparison

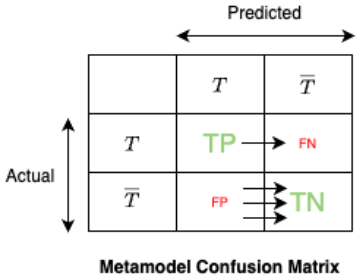
Primary $+$ meta vs. the primary alone, on the same trade pool: false positives avoided vs. true positives missed. The net effect should be positive, and the hit rate should rise meaningfully.

One honesty condition

Every number above is only as trustworthy as the data split behind it. On overlapping financial labels, standard cross-validation quietly cheats. Tuning and validating without fooling ourselves is its own discipline: the next subsection.

How the confusion matrix changes





- The primary alone predicts “trade” **every time**: its column of negatives is empty, so recall is 1 and precision equals the base rate.
- The meta-model moves mass across the matrix: many false positives become true negatives (the gain), a few true positives become false negatives (the price).
- The net effect is the point: **precision and F1 rise**, and the trades that remain are the ones worth taking.

Where we are

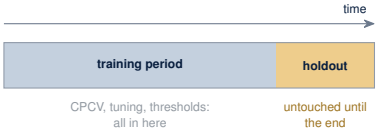
Hold out first; then fix cross-validation

Carve out the **holdout** before anything else: a chronologically-last slice the pipeline never sees until the end. CPCV, hyperparameter selection and thresholds all live on the training period only.

Why standard K-fold fails here

Leakage at the sample level. Triple-barrier labels **span multiple days**, so train and test windows that overlap share information.

One path is an anecdote. A single walk-forward backtest is too noisy to compare model configurations.



The two repairs

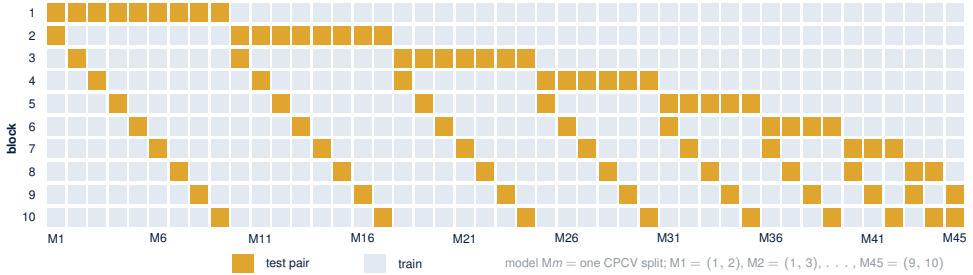
Purge: drop training samples whose label window reaches into the test block.

Embargo: drop a buffer just after each test block, against serial correlation.

López de Prado (2018), Ch. 7; BUSI70575, Lecture 1.

CPCV: a distribution of backtests, not one

Combinatorial Purged Cross-Validation: partition the training period into N=10 blocks and test on **every pair**: one model per column, each trained on 8 blocks and tested on the 2 gold ones.



The arithmetic that buys the distribution

$\binom{10}{2} = 45$ models, purge and embargo applied at every train/test boundary. Each block is tested by $\binom{9}{1} = 9$ of the 45 models, so stitching test blocks together yields **9 backtest paths per observation**: a **distribution** of out-of-sample outcomes instead of one equity curve, which is exactly what selecting between configurations requires [1].

Two-stage selection: statistical first, economic second

A recommended two-stage workflow:

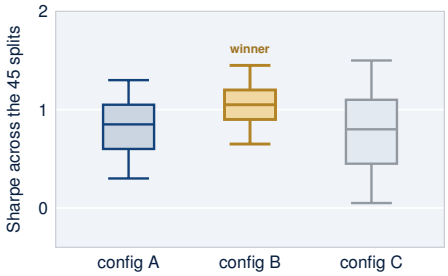
Stage 1: statistical loss, low variance

Rank every configuration by its **mean AUC across the 45 CPCV models**; keep the top $K = 3$.

Stage 2: economic loss, the tiebreaker

For each survivor, plot the **distribution of Sharpe ratios across the 45 splits**. Pick the configuration with the **highest median and tightest spread**, not the single best path.

Stage 1 narrows the field cheaply; Stage 2 decides on the quantity we care about, with its uncertainty visible.



Illustrative. All three survived Stage 1 on AUC; config B wins Stage 2 on the median and the spread.

Where this leaves us

Per instrument and per day: a side $s_{t,k} \in \{-1, 0, +1\}$ and an **out-of-fold** meta-probability $\hat{p}_{t,k} \in [0, 1]$. The last subsection turns this collection into portfolio weights.

Where we are

Combining signals: learn the portfolio directly

So far, every signal is **per instrument**. A natural question closes the pipeline:

> *What if we gather everything we know about every instrument (the primary side, the meta-probability, the features) and learn a **portfolio** directly?*

This is the program of a recent paper,[1]

- The method is **agnostic to the features**: it does not care whether $x_{t,k}$ holds raw technicals or our meta-signals.
- It is a recipe for **portfolio construction**: map per-asset feature windows to positions, and train the whole map by **maximizing the Sharpe ratio**.

Setup and notation

K instruments; days $t = 1, \ldots, T$; per-asset features $x_{t,k} \in \mathbb{R}^d$; lookback windows $X_{t,k} = [x_{t-L+1,k}, \ldots, x_{t,k}] \in \mathbb{R}^{L \times d}$; daily returns $r_{t,k}$.

[1] A. Saly-Kaufmann, K. Wood, J.-P. Calliess, S. Zohren, *Deep Learning for Financial Time Series: A Large-Scale Benchmark of Risk-Adjusted Performance*, University of Oxford.

Trading signal generation, and sizing to a common risk

Two-stage map, shared across architectures. A sequence model g_ϕ (LSTM, TFT, ...) compresses the window into a state, and a bounded head turns it into a **directional conviction**:

$$h_{t,k} = g_\phi(X_{t,k}), \qquad \hat{y}_{t,k} = \tanh\left(w_{\text{lin}}^\top h_{t,k} + b_{\text{lin}}\right) \in [-1, 1].$$

A ticker embedding lets the shared model specialise per instrument. Only g_ϕ changes across the study; head, sizing and loss stay fixed, which makes comparisons fair.

From conviction to weight. Estimate each asset's ex-ante volatility $\hat{\sigma}_{t,k}$ with an EWMA (information up to t only), then scale conviction to a common risk target:

$$\boxed{w_{t,k} = \hat{y}_{t,k} \cdot \frac{\sigma_{\text{tgt}}}{\hat{\sigma}_{t,k}}}$$

Low-vol assets are scaled **up**, high-vol assets **down**: each contributes comparable risk. The aggregate daily portfolio return is

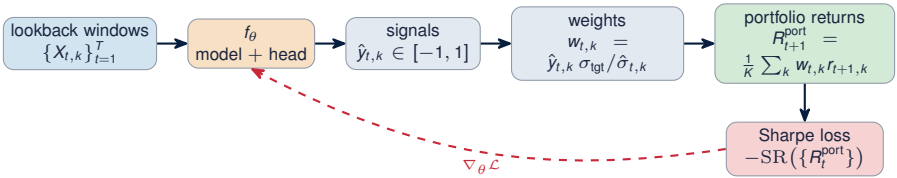
$$R_{t+1}^{\text{port}} = \frac{1}{K} \sum_{k=1}^{K} w_{t,k}\, r_{t+1,k}.$$

The objective: maximize the Sharpe ratio, end to end

No supervised target anywhere: the network is scored on the **whole return path** it generates. With sample moments $\hat{\mathbb{E}}[R]$ and $\widehat{\mathrm{Var}}[R]$ over the training window,

$$\mathcal{L}(\theta) = -\frac{\hat{\mathbb{E}}[R]}{\sqrt{\widehat{\mathrm{Var}}[R] + \epsilon}}\sqrt{252}$$

the negative annualized Sharpe ratio, computed on **pooled** returns across the batch and minimized by gradient descent.



One forward pass produces a return for **every** day in the window; the Sharpe is computed once on the pooled series, so gradients flow from the whole path back into f_θ. At inference there is no loss: window in, conviction out, weight out.

The bridge: meta-signals as input channels

Make the features concrete: feed the network **the outputs of our own pipeline**. Per instrument and day,

$$x_{t,k} = \Big[\underbrace{\text{features}}_{d \text{ cols}},\ \underbrace{s_{t,k}}_{\text{side}},\ \underbrace{\hat{p}_{t,k}}_{\text{meta prob.}}\Big] \in \mathbb{R}^{d+2}.$$

- $s_{t,k} \in \{-1, 0, +1\}$ carries **direction**; $\hat{p}_{t,k} \in [0, 1]$ is **direction-agnostic** by construction.
- Feeding them **separately** lets the network learn how trust should modulate direction, rather than forcing a fixed product $s \cdot \hat{p}$.

The one trap

Use the **out-of-fold** meta-probabilities. In-sample $\hat{p}$ is over-confidently correct and poisons the network with lookahead.

What the network now learns

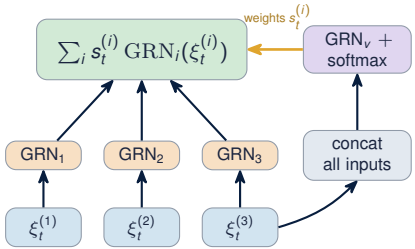
Sizing *and* cross-asset allocation, jointly, against the metric we actually care about, with the meta-model acting as a clean, compressed "trust" channel.

Inside g_ϕ (i): the Variable Selection Network

Financial inputs are many and mostly noise, and which ones matter **changes over time**. The VSN selects features **at every time step**:

- each input $\xi_t^{(i)}$ is transformed by its own **Gated Residual Network** (GRN);
- a parallel GRN on the concatenated inputs emits **softmax selection weights** $s_t^{(i)}$;
- the output is the weighted combination $\sum_i s_t^{(i)}\,\mathrm{GRN}_i(\xi_t^{(i)})$.

The weights $s_t^{(i)}$ are a **built-in, per-step feature importance**: interpretability by architecture, not post-hoc.



Built from scratch in the course's VSN programming session.

BUSI70575, Lectures 6–7; Lim et al. (2021).

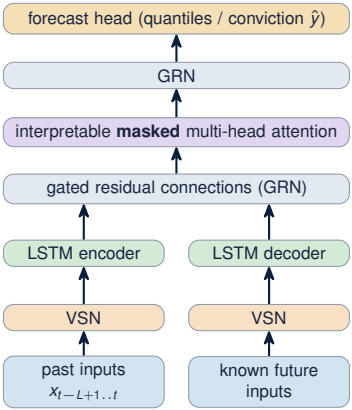
Inside g_ϕ (ii): the Temporal Fusion Transformer

The TFT [6] assembles these building blocks into one forecaster:

- **VSNs** select relevant inputs at each step: static, known-future, and observed-past features.
- An **LSTM encoder–decoder** captures local sequential dynamics.
- **GRN gating** with residual connections regularises every stage.
- **Masked multi-head attention** looks across time for context (masked, so no future leaks), and its weights are **readable**: which past dates drove the forecast.

In this pipeline

One of the strongest candidates for g_ϕ in the aggregation recipe: gated, interpretable, and built for exactly this kind of mixed, noisy input.



Sketch of the main path; static-covariate encoders omitted for clarity.

BUSI70575, Lecture 7.

SECTION 02

The Hendaye Strategy

Where we are

The universe

Hendaye runs the pipeline of Section 1 on **12 futures markets** across four asset classes:

Equity	ES S&P 500 · NQ Nasdaq 100
FX	6A Australian dollar · 6J Japanese yen
Energy	CL WTI crude · HO heating oil · NG natural gas · RB gasoline
Metals	GC gold · HG copper · PL platinum · SI silver

- Deep, liquid markets; long and short cost the same: the natural home for signed signals.
- Daily frequency, matching every model in this tutorial; each market is traded through its **continuous back-adjusted series**.

The strategy, factually

Built by **Syllogia** and run in partnership with **Alken Asset Management**. It is the pipeline you have just seen, implemented end to end, instrument by instrument, with nothing exotic added on the way to production.

From research to production

The strategy runs on **institutional infrastructure**: point-in-time data pipelines, out-of-fold evaluation, a held-out period reserved for the final assessment, and daily monitoring of every model output.

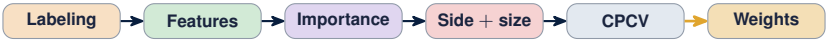
Where we are

1. The Systematic Trading Strategy Pipeline

2. The Hendaye Strategy
   - The Universe
   - The Strategy
   - The Performance Metrics

What is implemented, stage by stage

The pipeline of Section 1, end to end:



Pipeline stage	In Hendaye
Labeling	triple-barrier labels on managed-trade definitions; trend-scanning labels for the directional models
Feature creation	full library per instrument: technical & carry, volatility, **HMM regimes incl. Low Turbulence**, backward trend scanning, **GloVe pattern embeddings**
Feature importance	cluster-level, out-of-fold; reported per model, per instrument
Side and size	per-instrument primary models; one meta-model over their signals, trained on triple-barrier meta-labels
Aggregation	for now, an equal risk budget per instrument with volatility scaling to a 10% target; neural portfolio construction on the channels $[x,\ s,\ \hat{p}\,]$ is under exploration
Validation	purged and combinatorial cross-validation; a held-out period reserved for the final assessment

Where we are

Performance metrics

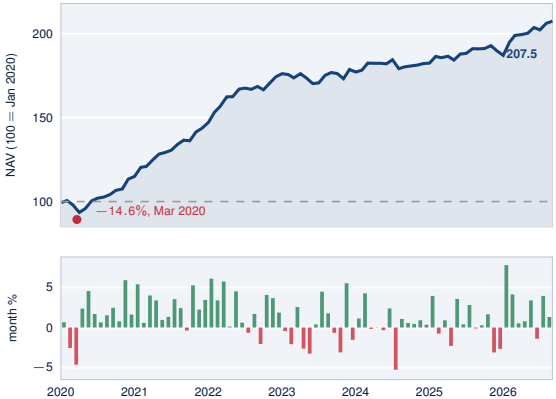
With daily returns r_t, $t = 1, \ldots, N$, equity curve NAV_t, Y years of data and $A = N/Y$ observations per year:

Total return	$\dfrac{\mathrm{NAV}_N}{\mathrm{NAV}_0} - 1$: the growth of capital over the whole period.
CAGR	$\left(\dfrac{\mathrm{NAV}_N}{\mathrm{NAV}_0}\right)^{1/Y} - 1$: the constant yearly rate that compounds to the same endpoint.
Annualised volatility	$\sigma = \sqrt{A}\ \mathrm{std}(r_t)$: the typical size of yearly fluctuations, the risk dial.
Sharpe ratio	$\mathrm{SR} = \sqrt{A}\,\dfrac{\mathrm{mean}(r_t)}{\mathrm{std}(r_t)}$: return earned per unit of risk taken.
Sortino ratio	$\mathrm{SO} = \sqrt{A}\,\dfrac{\mathrm{mean}(r_t)}{\sigma_\downarrow}$, with $\sigma_\downarrow = \sqrt{\mathrm{mean}\left(\min(r_t, 0)^2\right)}$: only downside moves count as risk.
Max drawdown	$\min_t \left(\dfrac{\mathrm{NAV}_t}{\max_{s \le t} \mathrm{NAV}_s} - 1\right)$: the worst peak-to-trough fall; *longest underwater* is the longest run with NAV_t below its running peak.

The four questions

How much did it earn (CAGR)? How bumpy was the ride (volatility)? Was the earning worth the bumps (Sharpe, Sortino)? And how bad did it get (drawdown, underwater)?

Results: January 2020 to August 2026



Headline statistics, 6.6 years	
total return	+107.5%
CAGR	11.7%
annualised volatility	10.0%
Sharpe	1.63
Sortino	2.39
max drawdown	−14.6%
longest underwater	187 days
positive months	74%
best / worst month	+7.8% / −5.3%

Daily P&L, 3 Jan 2020 to 6 Aug 2026, stated on the trading level that puts realised volatility at the 10% design target; Sharpe and Sortino do not depend on that choice.

What to remember

1. **The label is the hypothesis**: triple barrier grades managed trades; trend scanning picks the horizon.
2. **Features can be models**: HMM regimes, backward trend-scanning statistics, GloVe pattern embeddings.
3. **Importance needs care**: MDI in-sample, PFI out-of-sample; correlation forces the **cluster** level.
4. **Separate side from size**: a primary model owns direction; a meta-model, on different features, sizes the trust.
5. **Tune on distributions**: CPCV gives 45 purged splits, 9 paths per observation; select on the median Sharpe.
6. **Aggregate by learning**: a neural network maps all signals to weights, trained on the Sharpe ratio.

The course behind this hour

BUSI70575, *Systematic Trading Strategies with Machine Learning Algorithms*, Imperial College Business School. Everything shown today is taught there in full, and runs in Hendaye with Alken.

Thank you.

Hachem Madmoun

Imperial College Business School · Syllogia · MBZUAI

References I

[1] D. H. Bailey, J. M. Borwein, M. L. de Prado, and Q. J. Zhu. The probability of backtest overfitting. *Journal of Computational Finance*, 20 (4):39–69, 2017.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] M. L. de Prado. *Advances in Financial Machine Learning*. John Wiley & Sons, 2018.

[4] M. L. de Prado. *Machine Learning for Asset Managers*. Cambridge University Press, Elements in Quantitative Finance, 2020.

[5] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[6] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[7] H. Madmoun. *Creating Investment Strategies Based on Machine Learning Algorithms*. PhD thesis, École des Ponts ParisTech, 2022.

[8] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[9] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[10] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2): 257–286, 1989.

LLMs for Financial Markets

From Next-Token Prediction to Verifiable Financial Reasoning

Part of the KDD 2026 Tutorial "Generative AI and LLMs for Financial Markets"

Salem Lahlou, MBZUAI Abu Dhabi
salem.lahlou@mbzuai.ac.ae https://lahlou.org

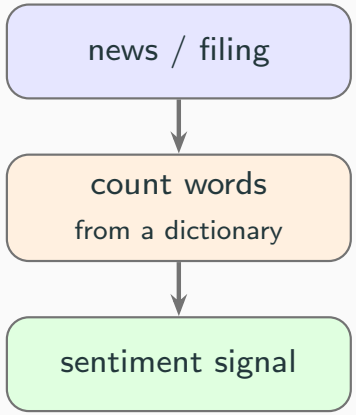
August 2026

2007: Text Was Already Moving Markets

Tetlock (2007), J. Finance

Count negative words in the WSJ "Abreast of the Market" column (1984–1999), using the Harvard psychology dictionary.

High pessimism $\Rightarrow$ downward pressure on the Dow the next day, then partial reversal.



Which of these is *negative* in a 10-K?

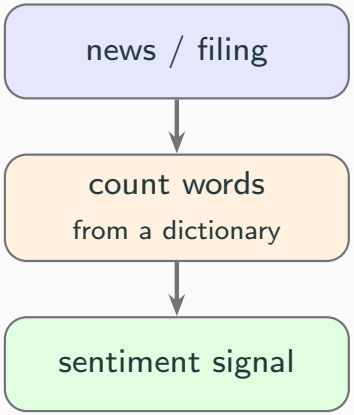
liability *tax* *restructuring* *cancer*

2007: Text Was Already Moving Markets

Tetlock (2007), J. Finance

Count negative words in the WSJ "Abreast of the Market" column (1984–1999), using the Harvard psychology dictionary.

High pessimism ⇒ downward pressure on the Dow the next day, then partial reversal.



Which of these is *negative* in a 10-K?

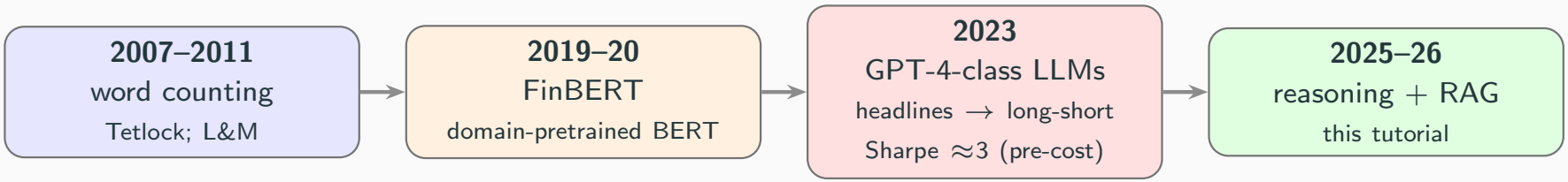
liability *tax* *restructuring* *cancer*

Loughran & McDonald (2011), J. Finance

∼3/4 of Harvard "negative" word counts in 10-Ks are not negative in finance. Built 6 finance-specific word lists. (The one that counts: *restructuring*.)

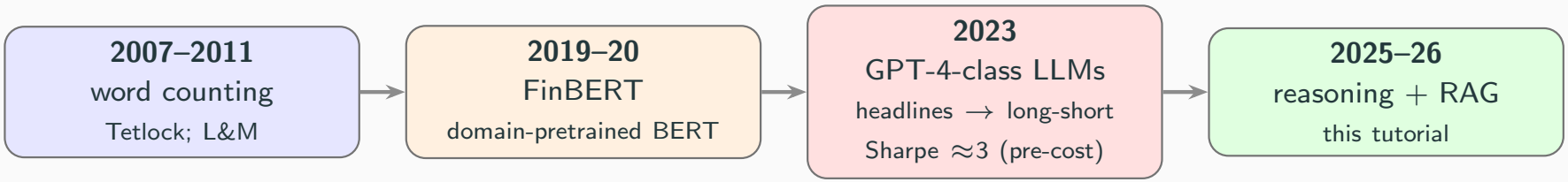
Text moved markets before any neural network. And general-purpose tools misread financial text.

The Same Question, New Tools



Huang, Wang & Yang (2023); Lopez-Lira & Tang (2023).

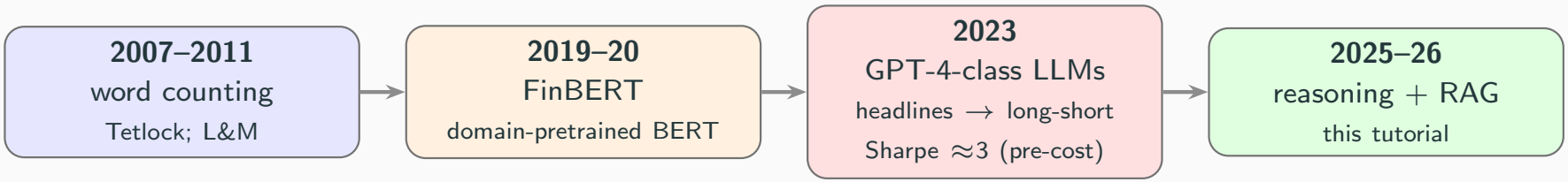
The Same Question, New Tools



Huang, Wang & Yang (2023); Lopez-Lira & Tang (2023).

BERT (2018): the first mainstream *pre-trained* language model: a neural network trained once on generic text, then fine-tuned for your task. **FinBERT**: BERT pre-trained on financial text; it beats the Loughran-McDonald dictionary on analyst-report sentiment, ∼88% vs ∼62% (Huang, Wang & Yang, 2023). Lopez-Lira & Tang also find BERT-class models *cannot* predict returns from headlines; GPT-4-class can: the ability *emerges* with scale (Part 1).

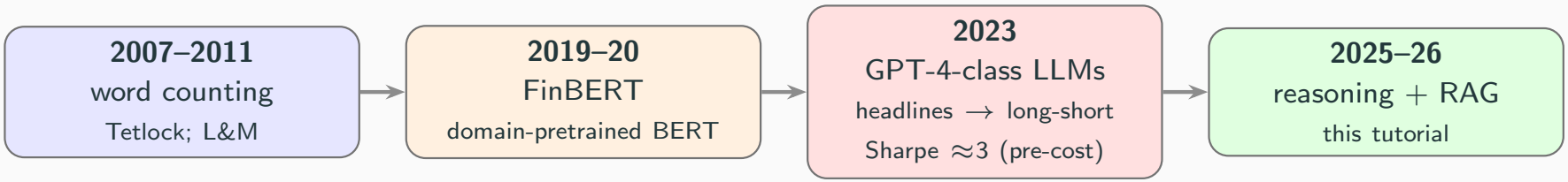
The Same Question, New Tools



Huang, Wang & Yang (2023); Lopez-Lira & Tang (2023).

BERT (2018): the first mainstream *pre-trained* language model: a neural network trained once on generic text, then fine-tuned for your task. **FinBERT**: BERT pre-trained on financial text; it beats the Loughran-McDonald dictionary on analyst-report sentiment, ∼88% vs ∼62% (Huang, Wang & Yang, 2023). Lopez-Lira & Tang also find BERT-class models *cannot* predict returns from headlines; GPT-4-class can: the ability *emerges* with scale (Part 1).

The Same Question, New Tools



Huang, Wang & Yang (2023); Lopez-Lira & Tang (2023).

BERT (2018): the first mainstream *pre-trained* language model: a neural network trained once on generic text, then fine-tuned for your task. **FinBERT**: BERT pre-trained on financial text; it beats the Loughran-McDonald dictionary on analyst-report sentiment, ∼88% vs ∼62% (Huang, Wang & Yang, 2023). Lopez-Lira & Tang also find BERT-class models *cannot* predict returns from headlines; GPT-4-class can: the ability *emerges* with scale (Part 1).

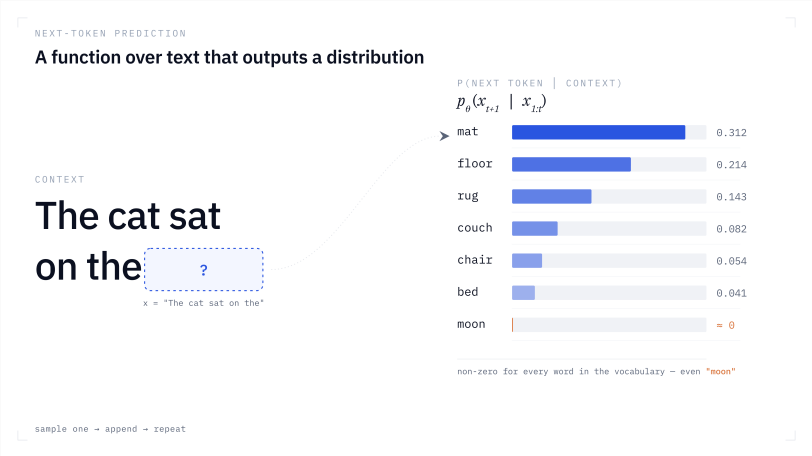
The plan, in five parts:

1. **What is inside an LLM?** Next-word prediction, scale, feedback, tools.
2. **Can it reason?** Four cases, and a taxonomy you build yourself.
3. **Grounding it: RAG** (with a notebook demo on an earnings call).
4. **Many models: debate, search, orchestration.**
5. **Measuring it in finance: benchmarks and FinChain.** (if time permits)

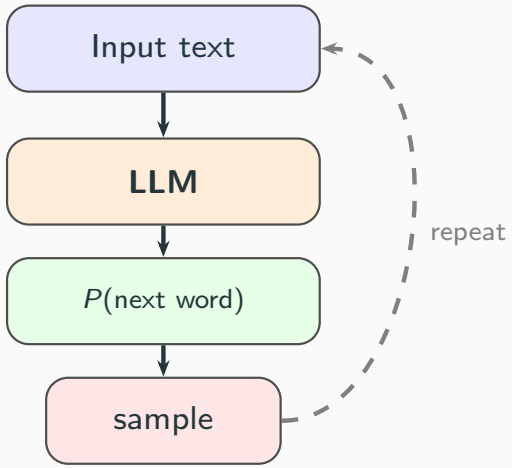
Part 1: What Is Inside an LLM?

Three simple ideas, and what they cost in finance

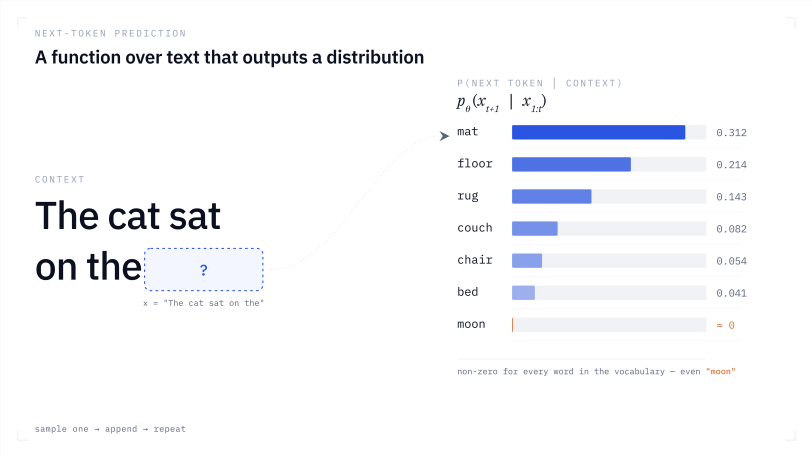
Idea 1: Predict the Next Word

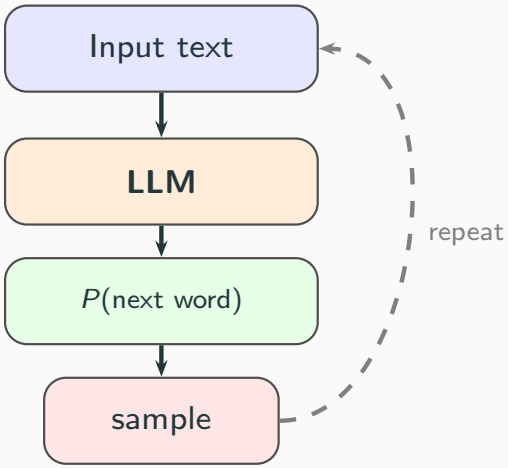


Modern AI is built on a function that takes text and outputs a probability for every word in the dictionary.



Idea 1: Predict the Next Word

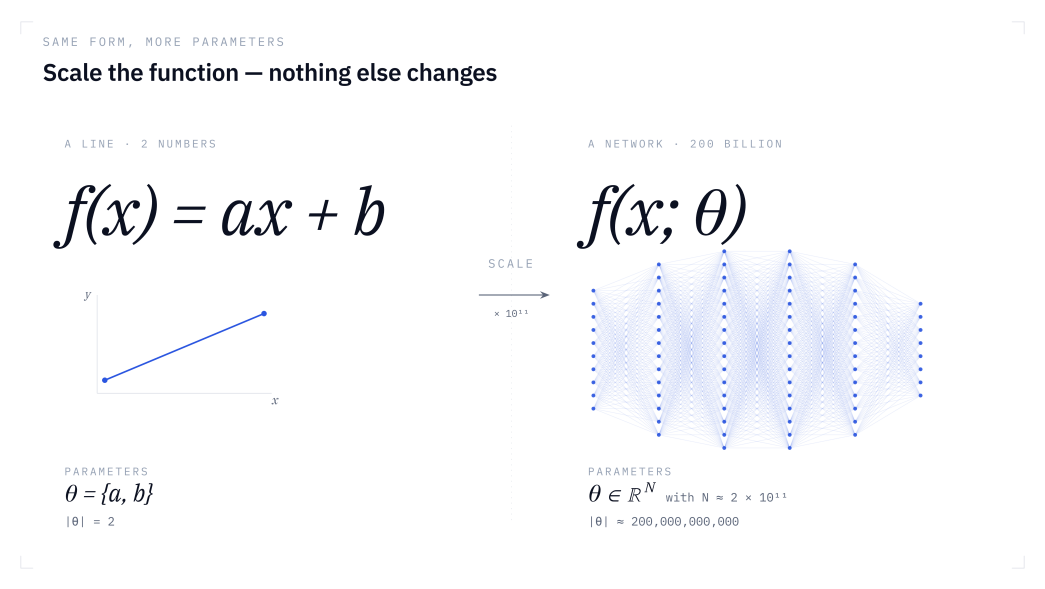




Modern AI is built on a function that takes text and outputs a probability for every word in the dictionary.

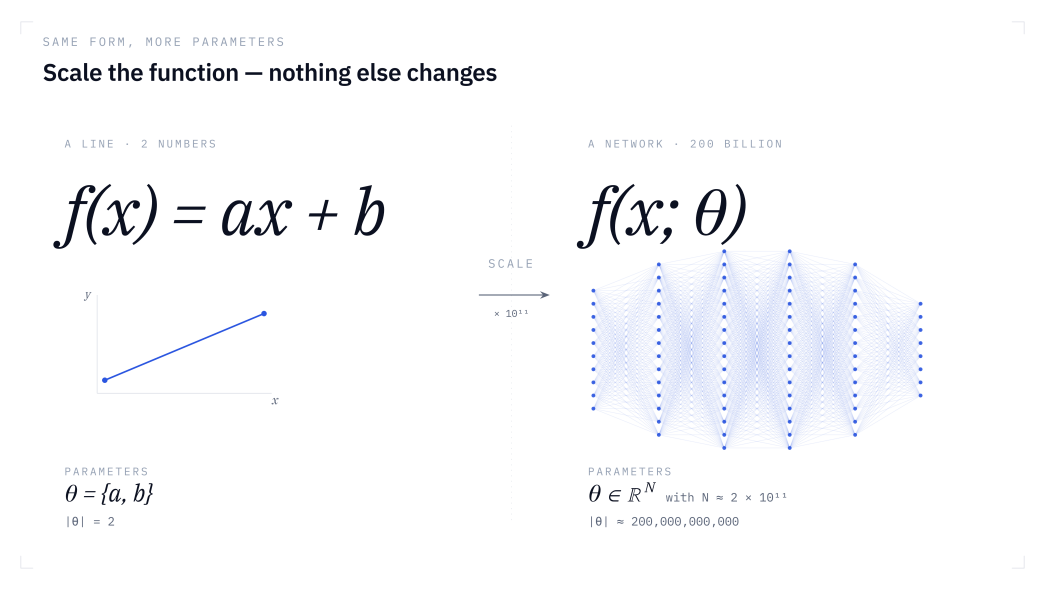
We sample from that distribution to pick the next word, and repeat. Same prompt, different answers: ask for a bond's fair value twice, nothing guarantees the same number twice.

A Very Large Function



$f(x) = ax + b$ has 2 parameters. A modern LLM has the same shape, with **hundreds of billions**. We feed it the internet; every time it puts low probability on the word that came next, we tweak the parameters.

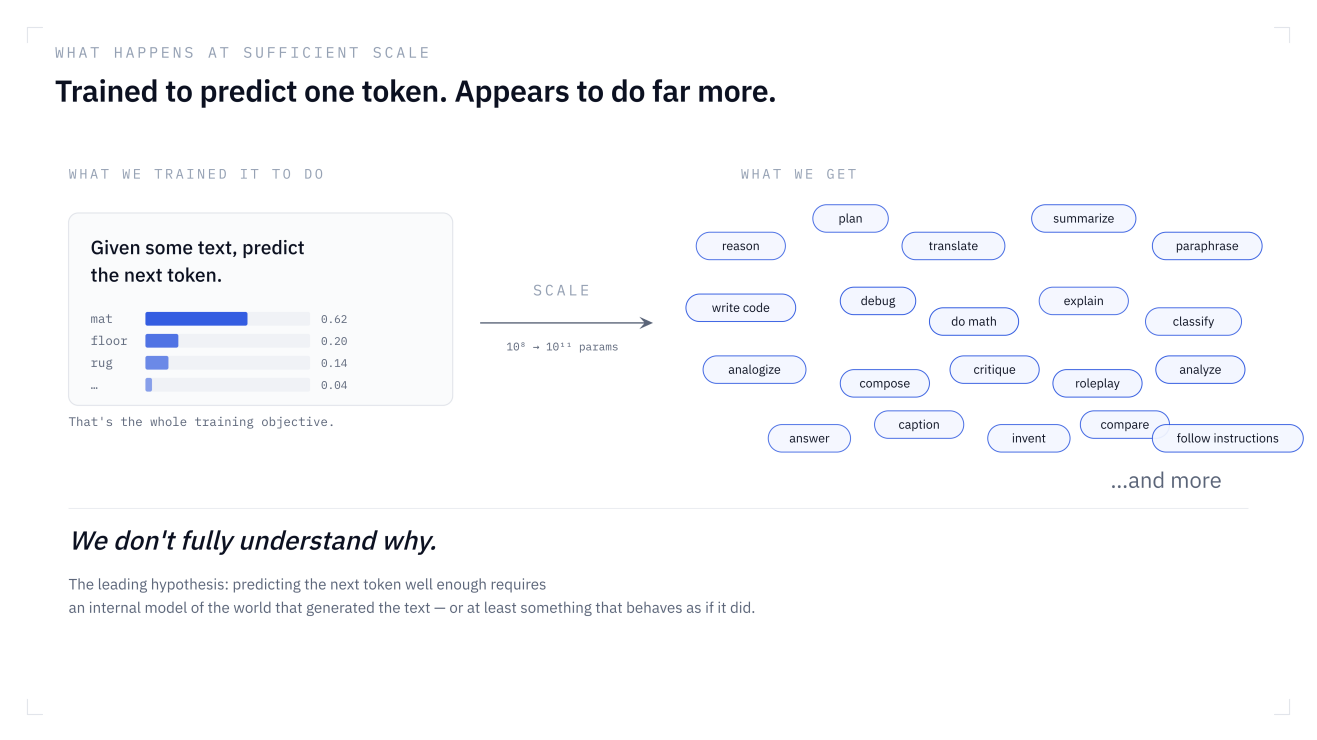
A Very Large Function



$f(x) = ax + b$ has 2 parameters. A modern LLM has the same shape, with **hundreds of billions**. We feed it the internet; every time it puts low probability on the word that came next, we tweak the parameters.

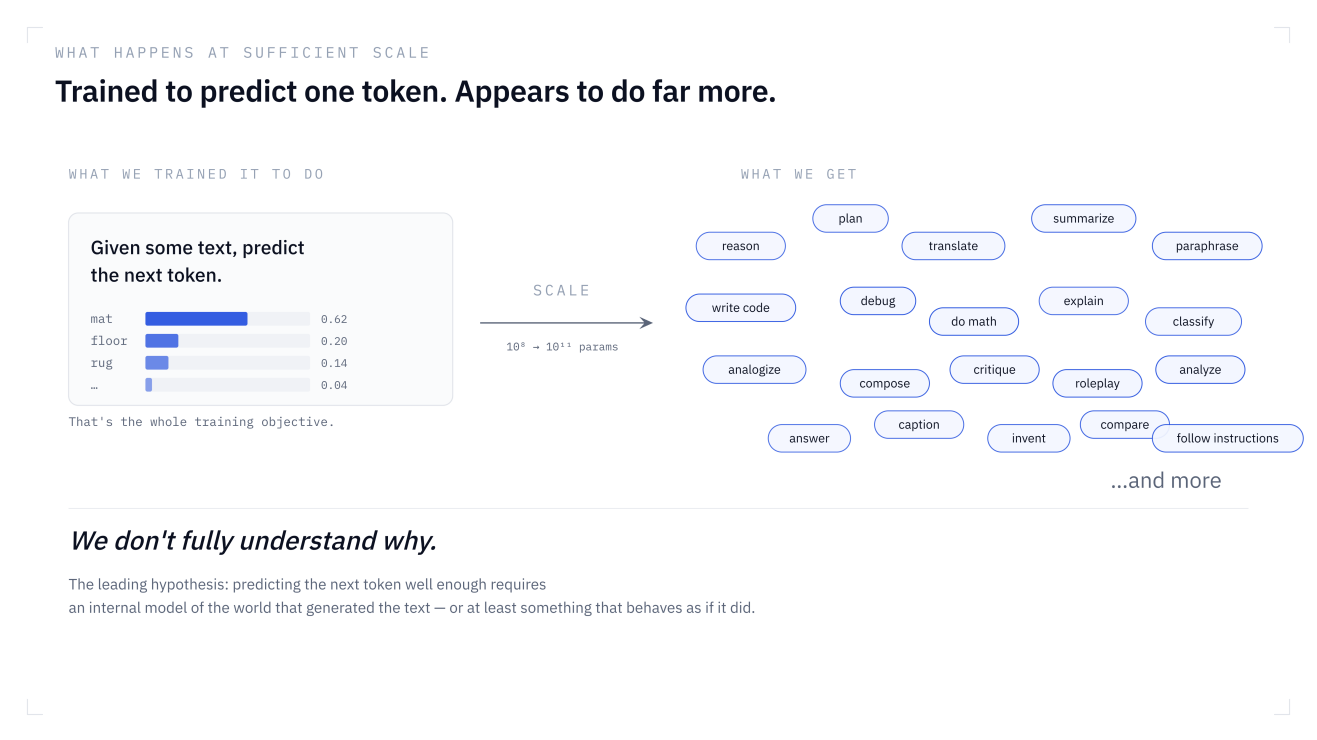
Model	Parameters	Note
BERT (2018)	110M	the base of FinBERT
GPT-3 (2020)	175B	
BloombergGPT (2023)	50B	363B tokens of financial text
Frontier models (2025–26)	>1T (est.)	undisclosed

What Happens When You Scale Up?



At sufficient scale, the model starts to *appear* to reason, plan, and know things. We don't fully understand why. Predicting text well enough seems to force the model to build some internal representation of the world that generated it.

What Happens When You Scale Up?

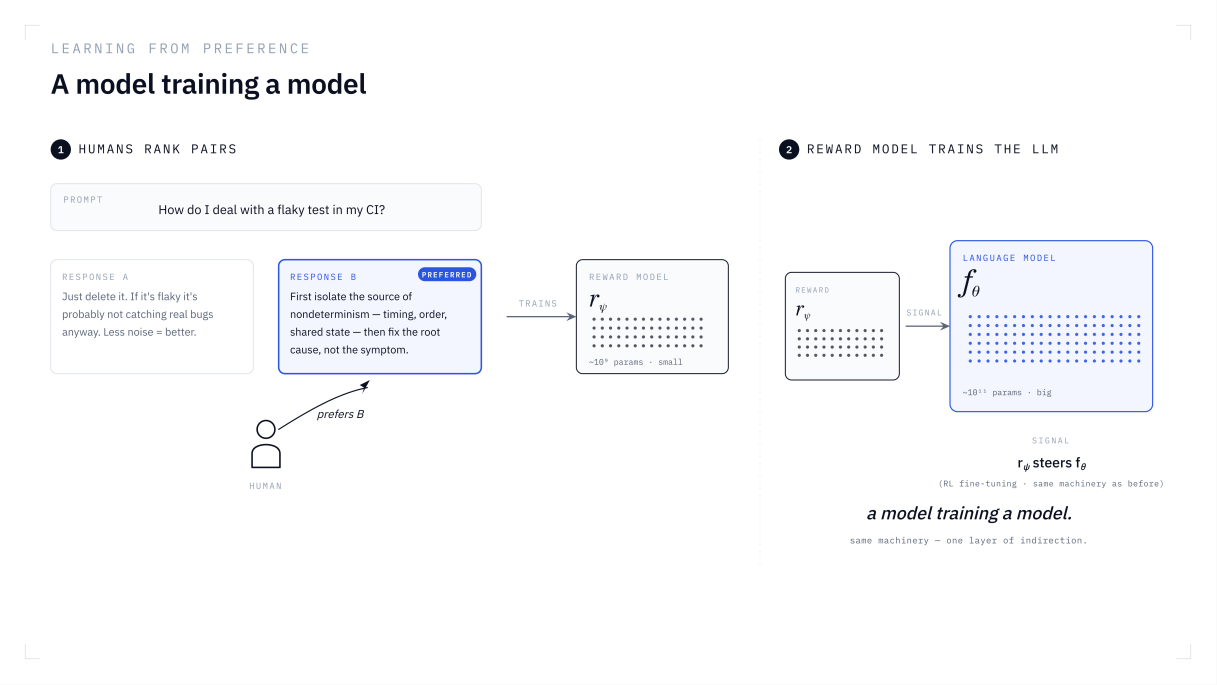


At sufficient scale, the model starts to *appear* to reason, plan, and know things. We don't fully understand why. Predicting text well enough seems to force the model to build some internal representation of the world that generated it.

Finance ran this bet early: **BloombergGPT**, 50B params trained on 363B tokens of financial text, beat same-size open models on financial tasks without losing general ability (Wu et al., 2023).

Idea 2: Steer with Human Preferences

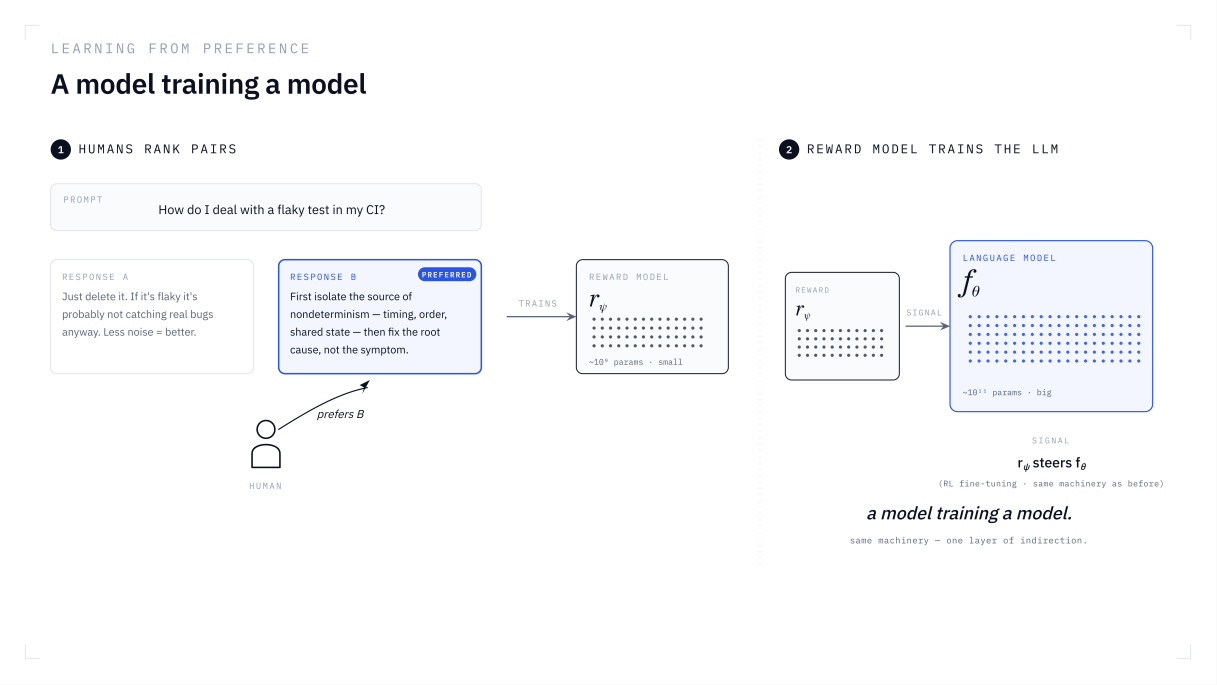
A next-word predictor trained on the internet will also predict harmful or false text. So we steer it.



Show humans pairs of responses; train a smaller model to predict their preferences; use it to steer the big one. (**RLHF**: *a model training a model.*)

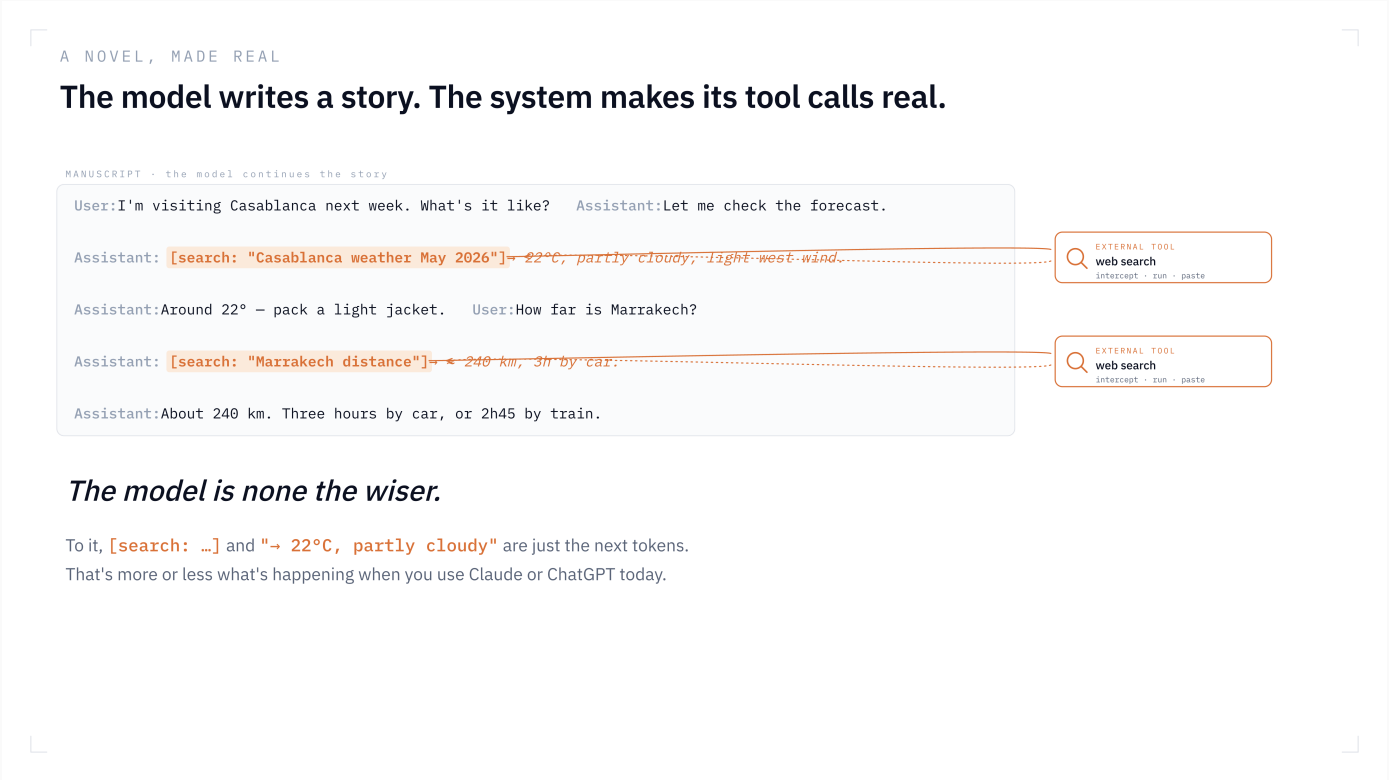
Idea 2: Steer with Human Preferences

A next-word predictor trained on the internet will also predict harmful or false text. So we steer it.



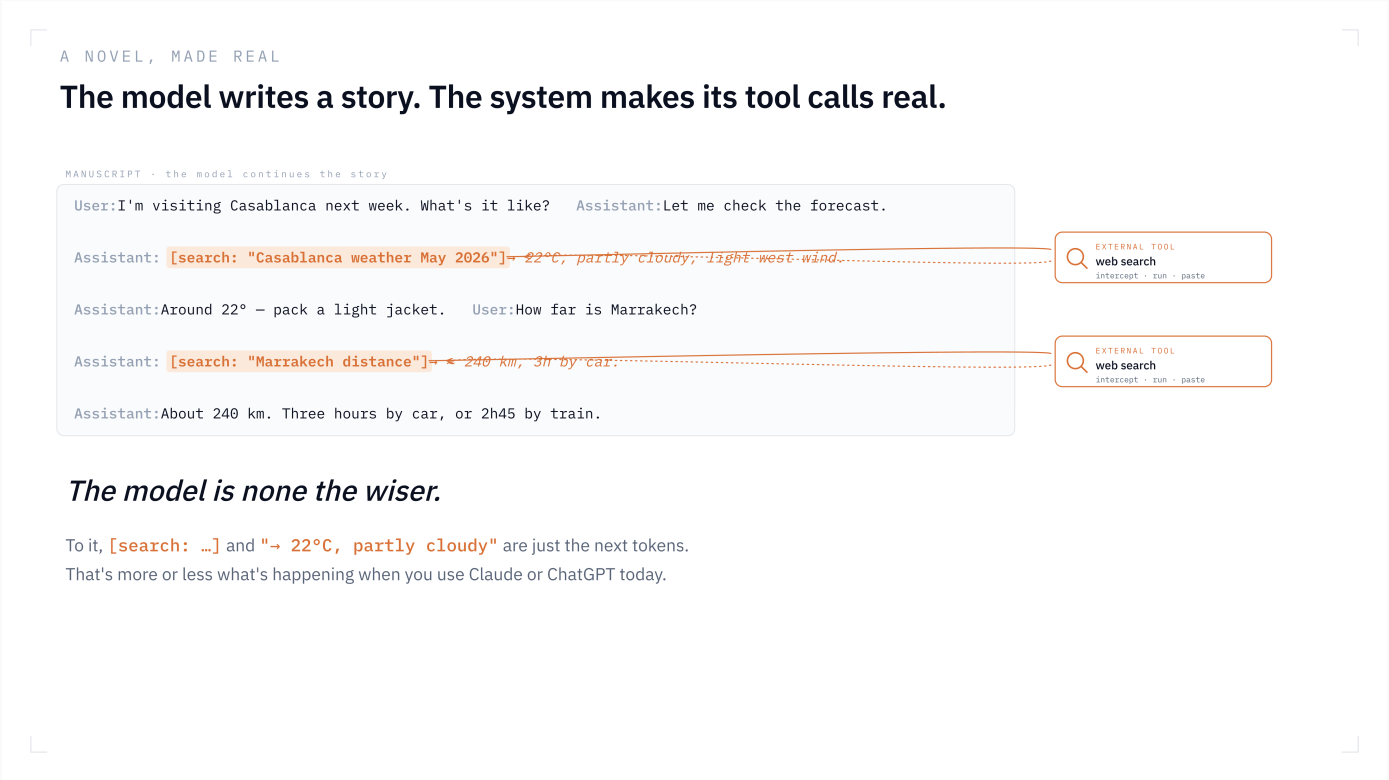
Show humans pairs of responses; train a smaller model to predict their preferences; use it to steer the big one. (**RLHF**: *a model training a model.*) Raters can be finance experts: "calibrated = good, invented numbers = very bad" is a trainable signal.

Idea 3: A Novel Made Real



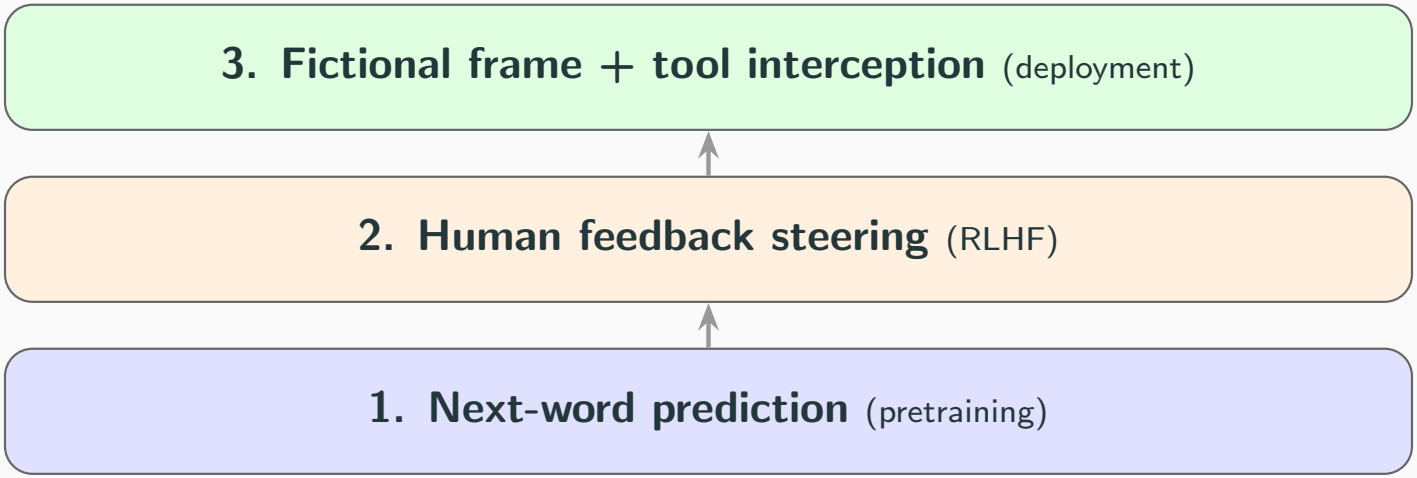
Imagine a novel whose main character is a supremely intelligent AI assistant with tools. Ask a language model to *continue that novel*. When the fictional AI writes a tool call, a real program intercepts those tokens and makes it real.

Idea 3: A Novel Made Real



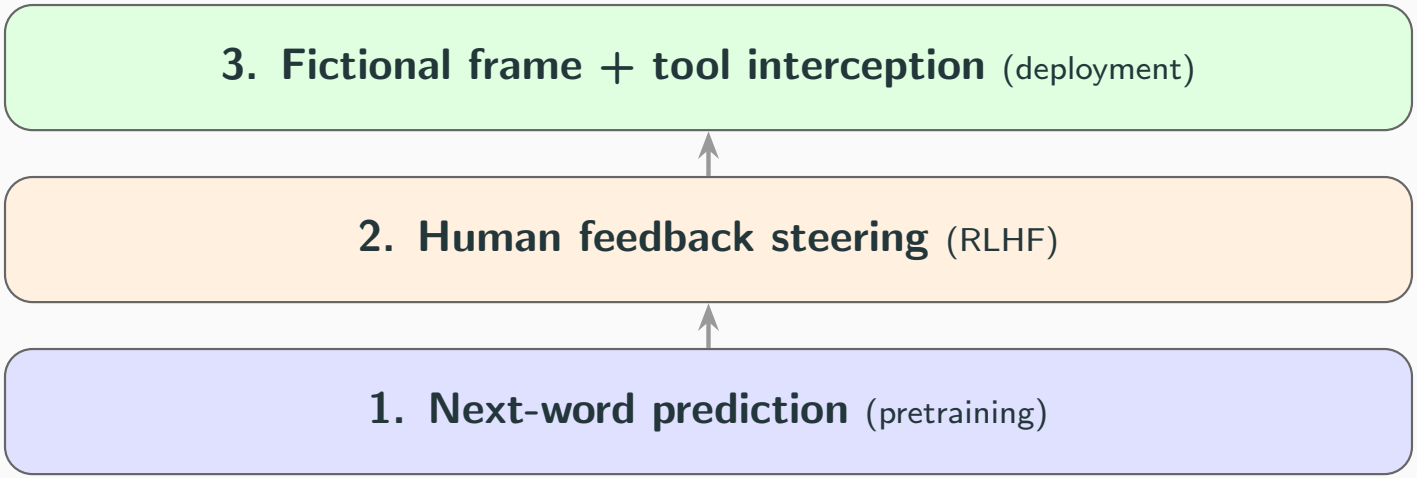
Imagine a novel whose main character is a supremely intelligent AI assistant with tools. Ask a language model to *continue that novel*. When the fictional AI writes a tool call, a real program intercepts those tokens and makes it real. Every "agent" in this tutorial is this mechanism.

That's the Entire Stack



No logic engine, no knowledge base, no valuation model inside. Just a very large function, clever training signals, and a lot of compute.

That's the Entire Stack



No logic engine, no knowledge base, no valuation model inside. Just a very large function, clever training signals, and a lot of compute.

The surprising part is not that it works. It's that **this is what was sufficient**.

What Does It Cost to Read a 10-K?

Models read tokens, not words:

- "EBITDA" $\approx$ 3 tokens; "the" $= 1$
- One 10-K $\approx$ **50K–200K tokens**
- Context windows: 200K–1M tokens

What Does It Cost to Read a 10-K?

Models read tokens, not words:

- "EBITDA" $\approx$ 3 tokens; "the" $= 1$
- One 10-K $\approx$ **50K–200K tokens**
- Context windows: 200K–1M tokens

Why it matters:

- One filing: cents. *Every* filing, *every* quarter, re-run daily: a budget line.
- Output tokens cost $\sim 5\times$ input.
- This asymmetry is why **retrieval** exists (Part 3).

Cost of *reading* one 10-K (input only):

Model tier	$/M input tokens	One 10-K
Small (Haiku 4.5)	$1	$0.05–0.20
Mid (Sonnet 5)	$3	$0.15–0.60
Large (Opus 5)	$5	$0.25–1.00
Frontier (Fable 5)	$10	$0.50–2.00

Anthropic API list prices, mid-2026. Other providers are in the same range.

One Wrong Number Is Not a Typo Here

Q: How did the company's cloud segment perform last quarter?

A: Cloud revenue grew 22% year-over-year to $8.4B, ahead of consensus estimates of $8.1B, driven by strong enterprise demand. Management raised full-year guidance accordingly.

Spot the problem.

One Wrong Number Is Not a Typo Here

Q: How did the company's cloud segment perform last quarter?

A: Cloud revenue grew 22% year-over-year to $8.4B, ahead of consensus estimates of $8.1B, driven by strong enterprise demand. Management raised full-year guidance accordingly.

Spot the problem. (You can't: the number is invented, plausible, and well-formatted.)

- Measured hallucination rates on grounded tasks: **a few % to ∼20%**; higher on open-ended factual recall.
- In finance, an invented figure in client communication can be a "misleading communication" (ESMA): a regulatory category, not a UX bug.

Two cures, two parts of this talk

1. Make it reason (fewer errors of logic) → Part 2. **2. Ground every number in a document** → Part 3 (RAG).

Part 2: Can It Reason?

Four cases first, then we look inside the reasoning models

Is THIS Reasoning? (1/4)

Q: What is the CAPM formula for the expected return of an asset?

A: $\mathbb{E}[R_i] = R_f + \beta_i \, (\mathbb{E}[R_m] - R_f)$: the risk-free rate plus the asset's beta times the market risk premium.

Is this reasoning?

Is THIS Reasoning? (1/4)

Q: What is the CAPM formula for the expected return of an asset?

A: $\mathbb{E}[R_i] = R_f + \beta_i\,(\mathbb{E}[R_m] - R_f)$: the risk-free rate plus the asset's beta times the market risk premium.

Is this reasoning?

No. This is **retrieval**: the formula appears verbatim, thousands of times, in the training data. Nothing was derived, combined, or checked. Fluent recall $\neq$ reasoning.

Is THIS Reasoning? (2/4)

Q: A project costs \$1,000 today and pays \$600 at the end of each of the next 2 years. Discount rate: 10%. Is the NPV positive?

A: Step 1: $600/1.10 = 545.45$. Step 2: $600/1.10^2 = 495.87$.
Step 3: $545.45 + 495.87 = 1{,}041.32 > 1{,}000$. Step 4: NPV $= +\$41.32$. Positive.

Is this reasoning?

Is THIS Reasoning? (2/4)

Q: A project costs \$1,000 today and pays \$600 at the end of each of the next 2 years. Discount rate: 10%. Is the NPV positive?

A: Step 1: $600/1.10 = 545.45$. Step 2: $600/1.10^2 = 495.87$.
Step 3: $545.45 + 495.87 = 1{,}041.32 > 1{,}000$. Step 4: NPV $= +\$41.32$. Positive.

Is this reasoning?

Yes: deductive. The conclusion is *guaranteed* by the premises. And every step is **mechanically checkable**: a 3-line script can verify it. Hold on to that word: *verify*.

Is THIS Reasoning? (3/4)

Observed: Stock X fell after each of the last three Fed rate hikes.

This morning: a fourth hike is announced.

Model: "X will likely fall today."

Is this reasoning?

Is THIS Reasoning? (3/4)

Observed: Stock X fell after each of the last three Fed rate hikes.

This morning: a fourth hike is announced.

Model: "X will likely fall today."

Is this reasoning?

Yes: inductive. Generalizing from instances; *probable*, never guaranteed. It is also what ML training is, and what half of sell-side commentary is. The job is to *quantify its uncertainty*.

Last One: What KIND of Reasoning Is This? (4/4)

Clues: volume spikes to 5× average; the CEO sold a large block last week; the quarterly filing is delayed.

Model: "The most plausible explanation: insiders anticipate bad news."

What kind of reasoning?

Last One: What KIND of Reasoning Is This? (4/4)

Clues: volume spikes to 5× average; the CEO sold a large block last week; the quarterly filing is delayed.

Model: "The most plausible explanation: insiders anticipate bad news."

What kind of reasoning?

Abductive: inference to the *best explanation*. Diagnosis, debugging, detective work, and interpreting market moves. Also where LLMs are most seductive: fluent explanations, with or without evidence.

Three Kinds of Reasoning: The Four You Just Saw

Deductive

Conclusions *guaranteed* by premises.

The NPV computation.

Math, proofs, pricing formulas. **Mechanically checkable.**

Three Kinds of Reasoning: The Four You Just Saw

Deductive

Conclusions *guaranteed* by premises.

The NPV computation.

Math, proofs, pricing formulas. **Mechanically checkable.**

Inductive

Generalize from observations; *probable*, not certain.

3 hikes, 3 falls $\Rightarrow$ *4th hike: likely fall.*

Science, ML training, backtests.

Abductive

Best *explanation* for the evidence.

Volume spike + CEO sale $\Rightarrow$ *insider concern.*

Diagnosis, debugging, analysis.

Three Kinds of Reasoning: The Four You Just Saw

Deductive

Conclusions *guaranteed* by premises.

The NPV computation.

Math, proofs, pricing formulas. **Mechanically checkable.**

Inductive

Generalize from observations; *probable*, not certain.

3 hikes, 3 falls $\Rightarrow$ *4th hike: likely fall.*

Science, ML training, backtests.

Abductive

Best *explanation* for the evidence.

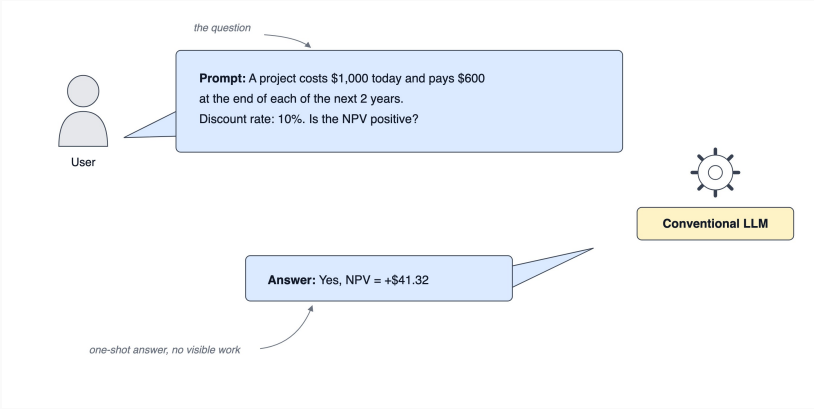
Volume spike + CEO sale $\Rightarrow$ *insider concern.*

Diagnosis, debugging, analysis.

Most LLM "reasoning" is **informal** (inductive + abductive). Deduction is the exception, and the only one we can *verify*: that fact powers everything on the next slides.

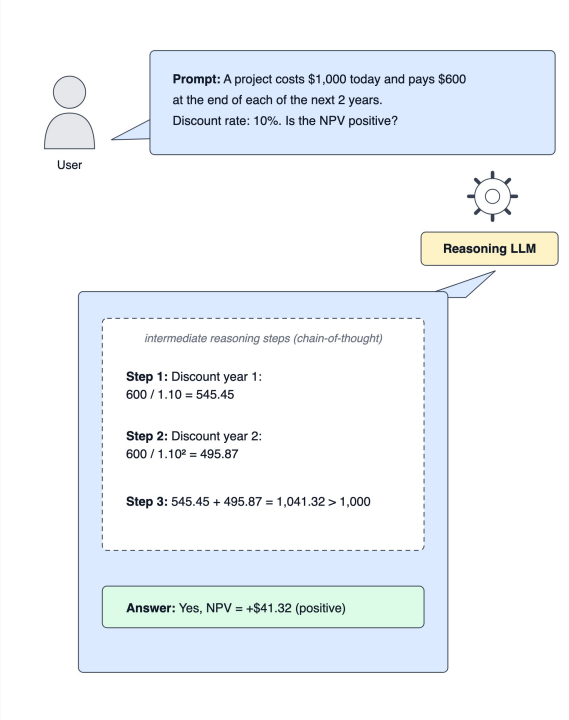
Standard vs. Reasoning LLMs

Standard LLM



Responds directly with a short answer

Reasoning LLM

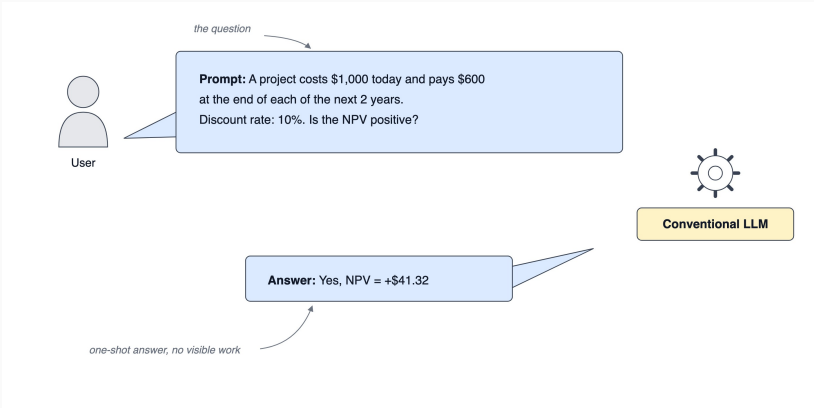


Generates intermediate thinking steps first

Layout after Raschka, *Build a Reasoning Model (From Scratch)*; example: the Part-2 present-value question.

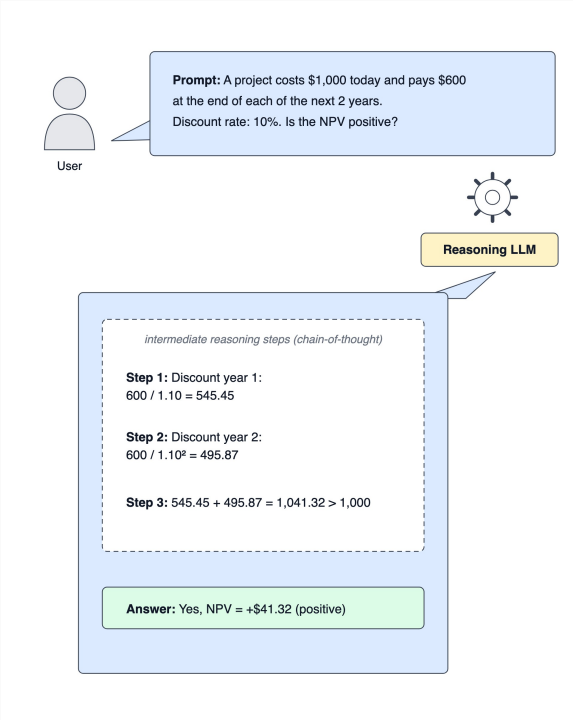
Standard vs. Reasoning LLMs

Standard LLM



Responds directly with a short answer

Reasoning LLM



Generates intermediate thinking steps first

Layout after Raschka, *Build a Reasoning Model (From Scratch)*; example: the Part-2 present-value question.

A reasoning LLM generates intermediate steps before answering. This often **dramatically improves accuracy**, and financial arithmetic is exactly this shape.

Chain-of-Thought: Just Ask It to Think

Standard prompting

Q: \$10,000 at 6% annual, compounded quarterly, for 2 years. How much interest?

A: $10{,}000 \times 6\% \times 2 = \mathbf{\$1{,}200}$. ✗ (simple interest)

Chain-of-thought prompting

A: Quarterly rate = 1.5%; 8 quarters; $10{,}000 \times 1.015^8 = 11{,}264.93$. Interest = **\$1,264.93**. ✓

Same model, same weights. Only the prompt changed.

Chain-of-Thought: Just Ask It to Think

Standard prompting

Q: \$10,000 at 6% annual, compounded quarterly, for 2 years. How much interest?

A: $10{,}000 \times 6\% \times 2 = \mathbf{\$1{,}200}$. ✗ (simple interest)

Chain-of-thought prompting

A: Quarterly rate = 1.5%; 8 quarters; $10{,}000 \times 1.015^8 = 11{,}264.93$. Interest = **\$1,264.93**. ✓

Same model, same weights. Only the prompt changed.

The trick (Wei et al., 2022):

- Intermediate steps unlock multi-step problems

Zero-shot CoT (Kojima et al., 2022):

- Just append *"Let's think step by step"*

Result

CoT alone: a 0.6B model goes 15.2% → **40.6%** on MATH-500 (Raschka).

Chain-of-Thought: Just Ask It to Think

Standard prompting

Q: \$10,000 at 6% annual, compounded quarterly, for 2 years. How much interest?

A: $10{,}000 \times 6\% \times 2 = \mathbf{\$1{,}200}$. ✗ (simple interest)

Chain-of-thought prompting

A: Quarterly rate = 1.5%; 8 quarters; $10{,}000 \times 1.015^8 = 11{,}264.93$. Interest = **\$1,264.93**. ✓

Same model, same weights. Only the prompt changed.

The trick (Wei et al., 2022):

- Intermediate steps unlock multi-step problems

Zero-shot CoT (Kojima et al., 2022):

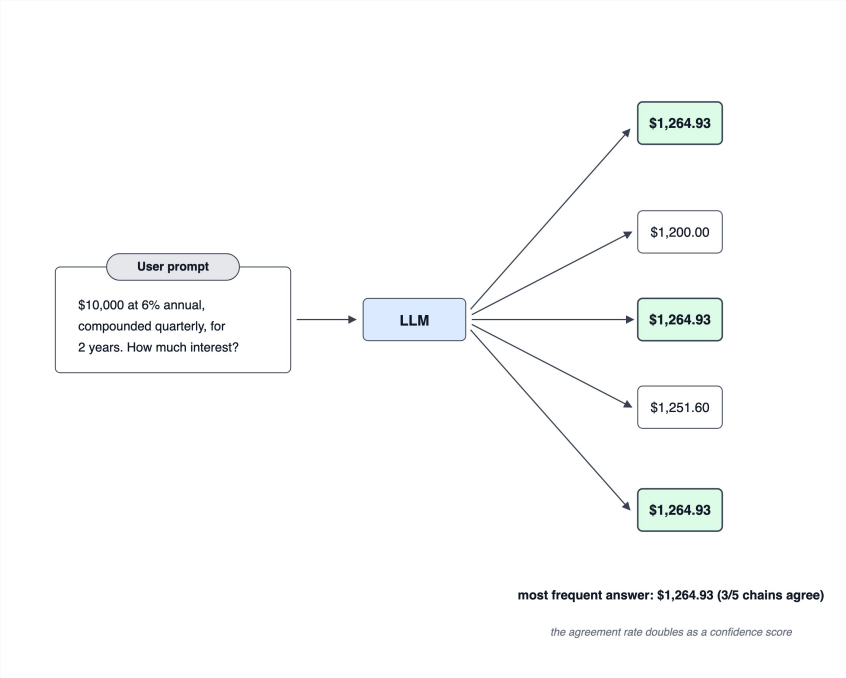
- Just append *"Let's think step by step"*

Result

CoT alone: a 0.6B model goes 15.2% → **40.6%** on MATH-500 (Raschka).

In finance: FinQA/ConvFinQA, multi-step numerical QA over real filings (Chen et al., 2021; 2022). Writing the steps as *runnable code* (Program of Thoughts) adds ∼12% over CoT across math+financial QA (Chen et al., 2023).

Self-Consistency: Ask Ten Times, Take a Vote



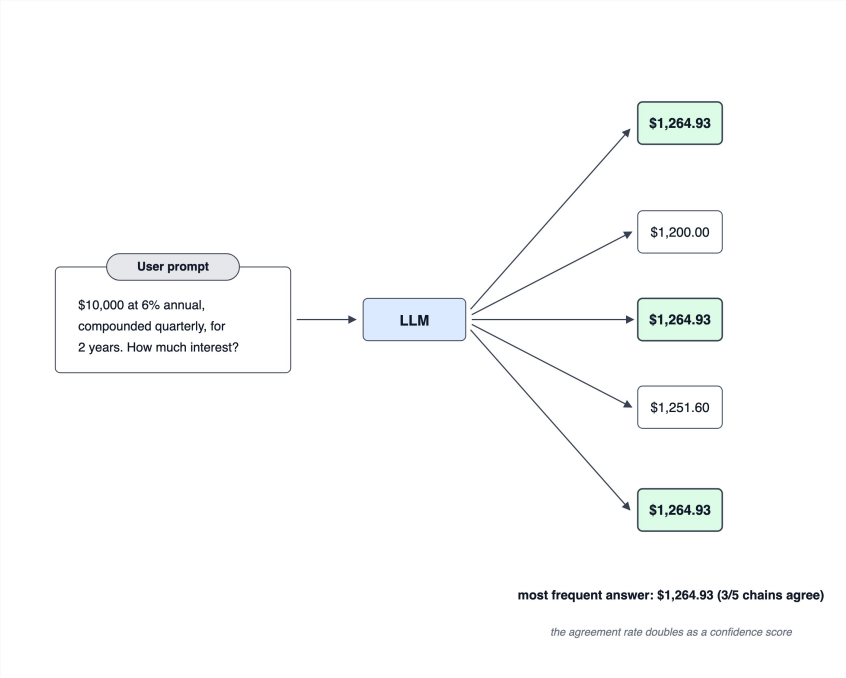
Self-consistency with majority voting, on the compound-interest question

Recipe: sample N chains $\rightarrow$ extract answers $\rightarrow$ majority vote.

Result

CoT + self-consistency (n=10): 15.2% $\rightarrow$ **52.0%** on MATH-500 (0.6B model), at $\sim$85$\times$ compute.

Self-Consistency: Ask Ten Times, Take a Vote



Self-consistency with majority voting, on the compound-interest question

Recipe: sample N chains $\rightarrow$ extract answers $\rightarrow$ majority vote.

Result

CoT + self-consistency (n=10): 15.2% $\rightarrow$ **52.0%** on MATH-500 (0.6B model), at $\sim$85$\times$ compute.

Free confidence signal: 9/10 chains agree $\Rightarrow$ high confidence; 4/10 $\Rightarrow$ abstain. The cheapest calibration in the LLM toolbox; our demo uses it.

What It Costs: Accuracy vs. Compute

0.6B model, MATH-500 (Raschka):

Method	**Acc.**	**Rel. cost**
Greedy	15.2%	1×
+ CoT	40.6%	8×
+ Self-consistency (n=10)	**52.0%**	85×

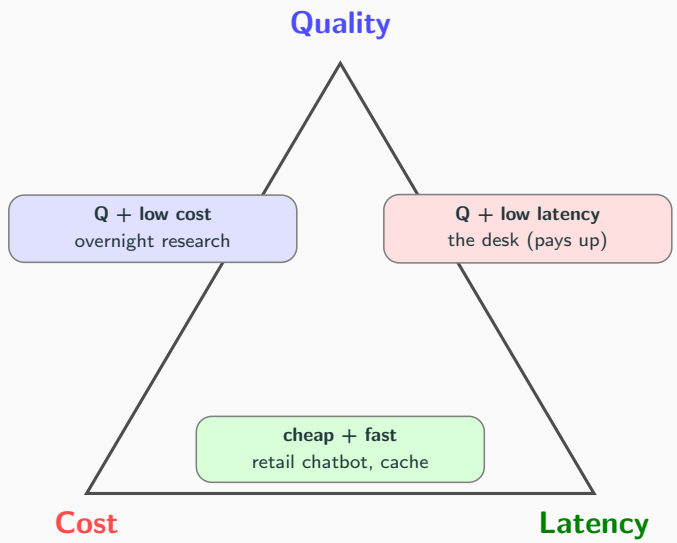
Gains are real, and you pay per query.

What It Costs: Accuracy vs. Compute

0.6B model, MATH-500 (Raschka):

Method	Acc.	Rel. cost
Greedy	15.2%	1×
+ CoT	40.6%	8×
+ Self-consistency (n=10)	**52.0%**	85×

Gains are real, and you pay per query.



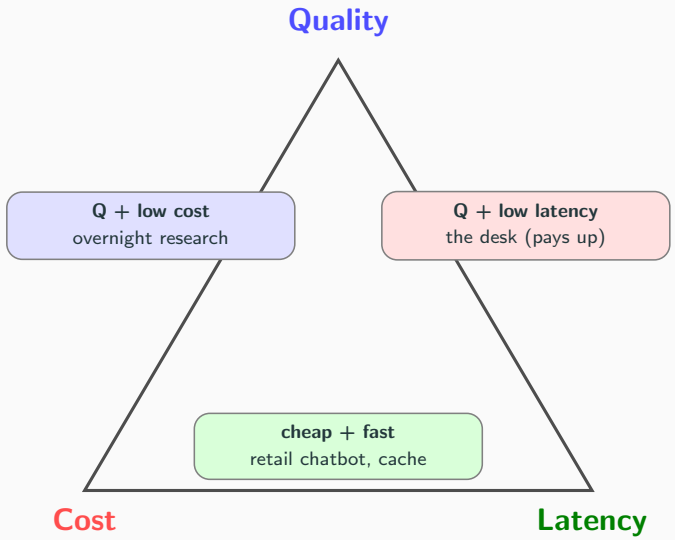
Pick two. Then **route** and **cache**.

What It Costs: Accuracy vs. Compute

0.6B model, MATH-500 (Raschka):

Method	Acc.	Rel. cost
Greedy	15.2%	1×
+ CoT	40.6%	8×
+ Self-consistency (n=10)	**52.0%**	85×

Gains are real, and you pay per query.

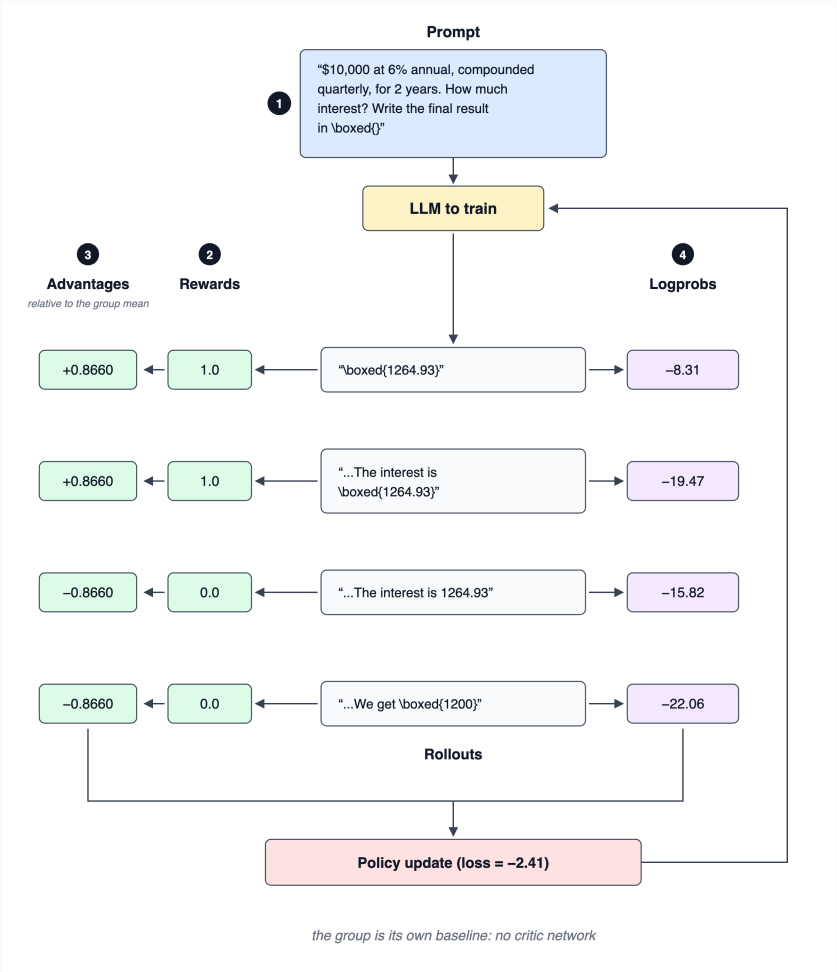


Pick two. Then **route** and **cache**.

The question: instead of *renting* reasoning per query, can we *train it into the weights*?

Training It In: RL with GRPO

Text generation as RL: state = text so far; action = next token; reward at the end (answer checked).



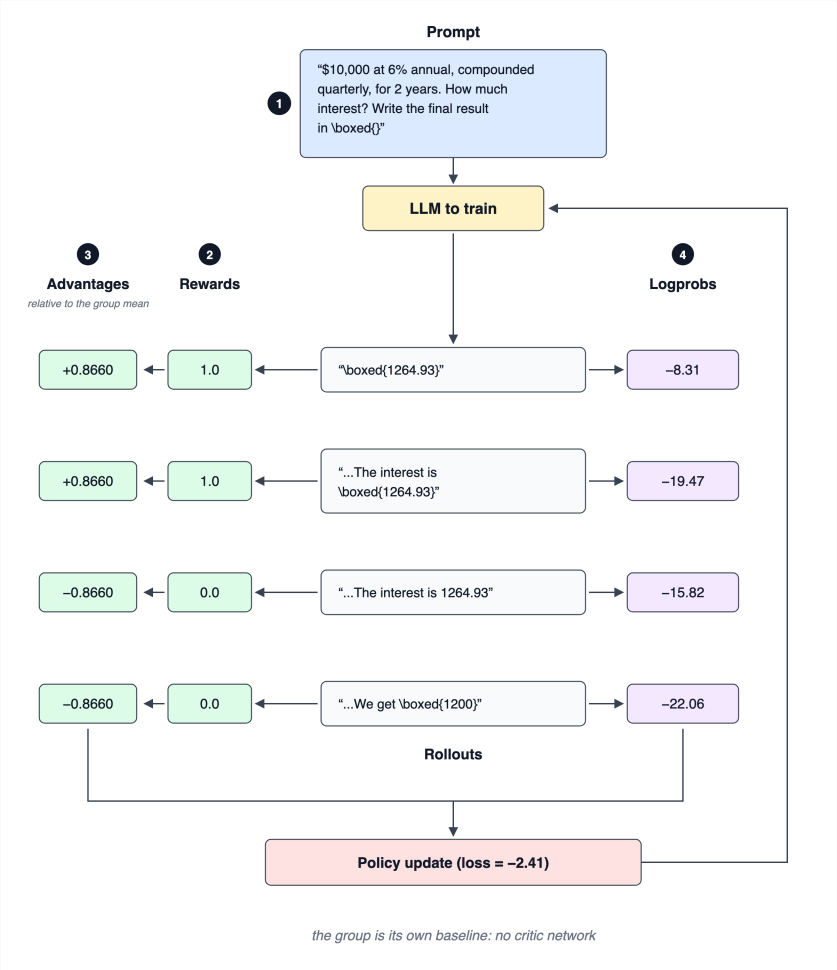
GRPO on one worked prompt (illustrative values)

Training It In: RL with GRPO

Text generation as RL: state = text so far; action = next token; reward at the end (answer checked).

GRPO (DeepSeek, 2024):

- Sample N solutions per problem
- Advantage = **relative to group mean**
- Reinforce the better-than-average ones
- **No critic network** (unlike PPO)



GRPO on one worked prompt (illustrative values)

Training It In: RL with GRPO

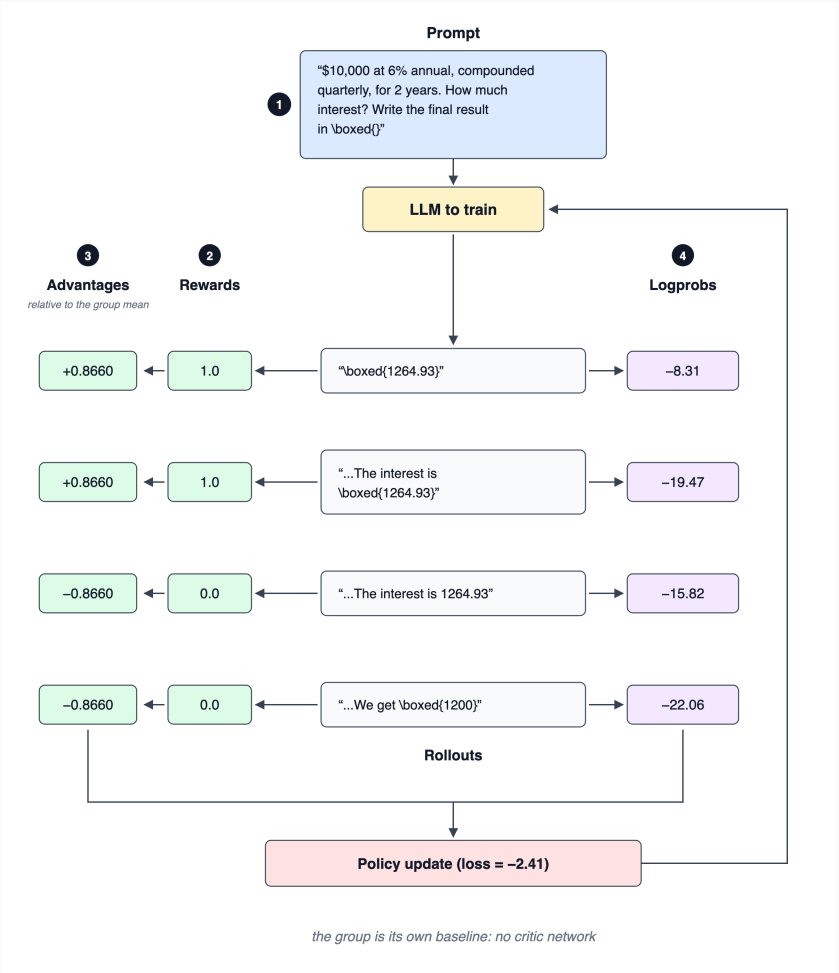
Text generation as RL: state = text so far; action = next token; reward at the end (answer checked).

GRPO (DeepSeek, 2024):

- Sample N solutions per problem
- Advantage = **relative to group mean**
- Reinforce the better-than-average ones
- **No critic network** (unlike PPO)

Result

50 GRPO steps: 15.2% $\rightarrow$ **47.4%** on MATH-500 (0.6B). Inference cost: 1×.

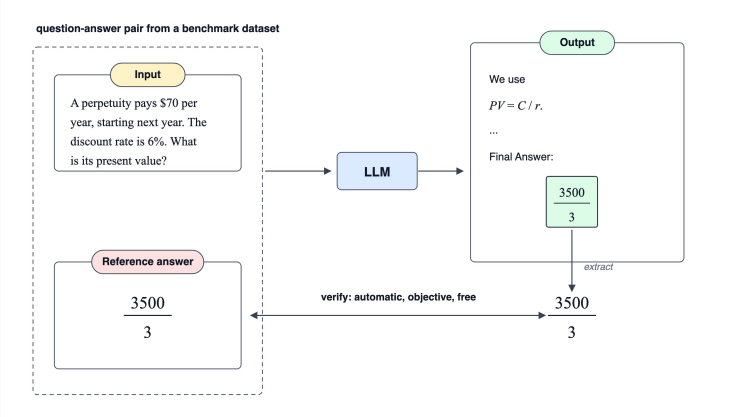


GRPO on one worked prompt (illustrative values)

The Secret Ingredient: Verifiable Rewards

DeepSeek-R1 recipe (2025):

1. Large base model (DeepSeek-V3, 671B)
2. Cold-start supervised fine-tuning (SFT): teach the *format*
3. **GRPO at scale, reward = a program**
4. Distill into 1.5B–70B students



The math checks itself

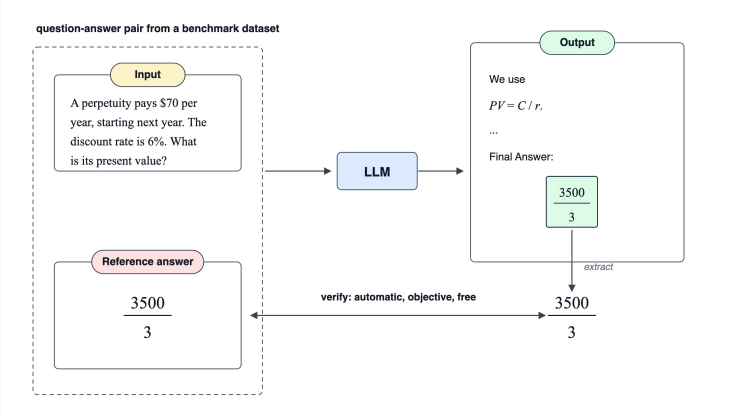
The Secret Ingredient: Verifiable Rewards

DeepSeek-R1 recipe (2025):

1. Large base model (DeepSeek-V3, 671B)
2. Cold-start supervised fine-tuning (SFT): teach the *format*
3. **GRPO at scale, reward = a program**
4. Distill into 1.5B–70B students

The reward, concretely:

- Math: extract answer, compare symbolically
- Code: run the test suite
- Automatic, objective, free

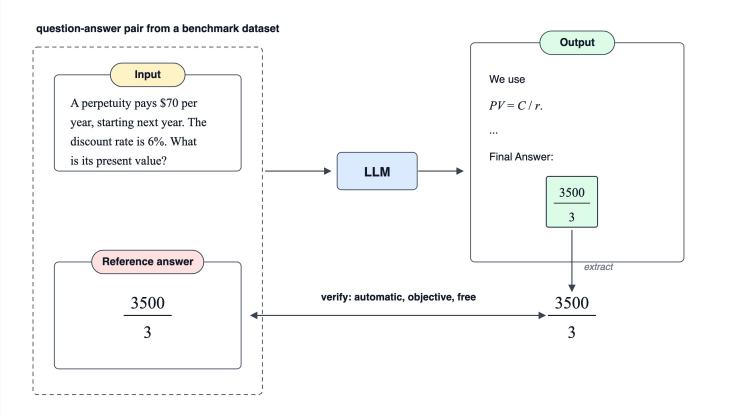


The math checks itself

The Secret Ingredient: Verifiable Rewards

DeepSeek-R1 recipe (2025):

1. Large base model (DeepSeek-V3, 671B)
2. Cold-start supervised fine-tuning (SFT): teach the *format*
3. **GRPO at scale, reward = a program**
4. Distill into 1.5B–70B students



The math checks itself

The reward, concretely:

- Math: extract answer, compare symbolically
- Code: run the test suite
- Automatic, objective, free

No human annotators in the RL stage. **The math checks itself.** So: what checks itself in *finance*?

Verifiers Are the Bottleneck (and They Can Be Wrong)

Mathematical

Run code, compare to ground truth.

`sympy.simplify(a)==b`

$\Rightarrow$ **exact, cheap** (R1's engine)

Claim

Check a statement against documents / a KB.

"Revenue grew 22%" $\rightarrow$ find the filing

$\Rightarrow$ noisy, coverage-bound

Policy

Check constraints: format, length, compliance.

`len(out)<=500`

$\Rightarrow$ cheap, partial coverage

Verifiers Are the Bottleneck (and They Can Be Wrong)

Mathematical

Run code, compare to ground truth.

`sympy.simplify(a)==b`

$\Rightarrow$ **exact, cheap** (R1's engine)

Claim

Check a statement against documents / a KB.

"Revenue grew 22%" $\rightarrow$ find the filing

$\Rightarrow$ noisy, coverage-bound

Policy

Check constraints: format, length, compliance.

`len(out)<=500`

$\Rightarrow$ cheap, partial coverage

And verifiers can be wrong. With flip rates ρ_+, ρ_-, the GRPO gradient shrinks by exactly $(1 - \rho_+ - \rho_-)$; estimating and correcting the noise recovers up to **+6.7pp** on GSM8K.

(Noise-corrected GRPO, El Mansouri, Izzati, Seddik & Lahlou, ICML 2026.)

Verifiers Are the Bottleneck (and They Can Be Wrong)

Mathematical

Run code, compare to ground truth.

`sympy.simplify(a)==b`

$\Rightarrow$ **exact, cheap** (R1's engine)

Claim

Check a statement against documents / a KB.

"Revenue grew 22%" $\rightarrow$ find the filing

$\Rightarrow$ noisy, coverage-bound

Policy

Check constraints: format, length, compliance.

`len(out)<=500`

$\Rightarrow$ cheap, partial coverage

And verifiers can be wrong. With flip rates ρ_+, ρ_-, the GRPO gradient shrinks by exactly $(1 - \rho_+ - \rho_-)$; estimating and correcting the noise recovers up to **+6.7pp** on GSM8K. *(Noise-corrected GRPO, El Mansouri, Izzati, Seddik & Lahlou, ICML 2026.)*

The middle column, **checking claims against documents**, is what finance needs. Building it is a discipline of its own: **retrieval** $\rightarrow$ Part 3.

Part 3: Grounding It, with Retrieval-Augmented Generation

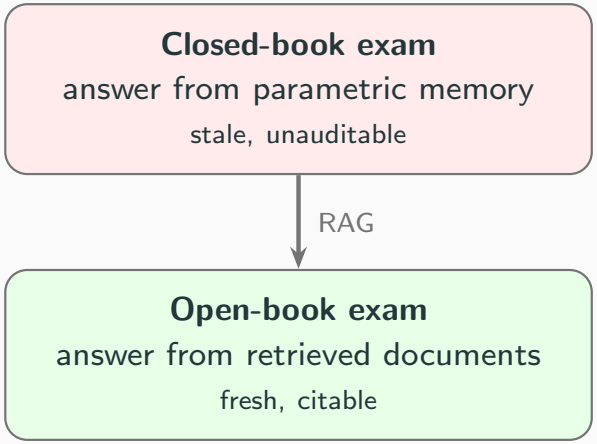
From BM25 to a confidence-scored signal on a real earnings call

Why Retrieval? Three Finance-Shaped Reasons

1. **Staleness.** Weights frozen months ago; yesterday's earnings call isn't in them. Facts here change quarterly, sometimes by the minute.
2. **Hallucination.** Quote a retrieved document, and the invented $8.4B (Part 1) gets caught.
3. **Auditability.** An answer with a citation is something compliance can check. "The model felt bullish" is not a defensible rationale.

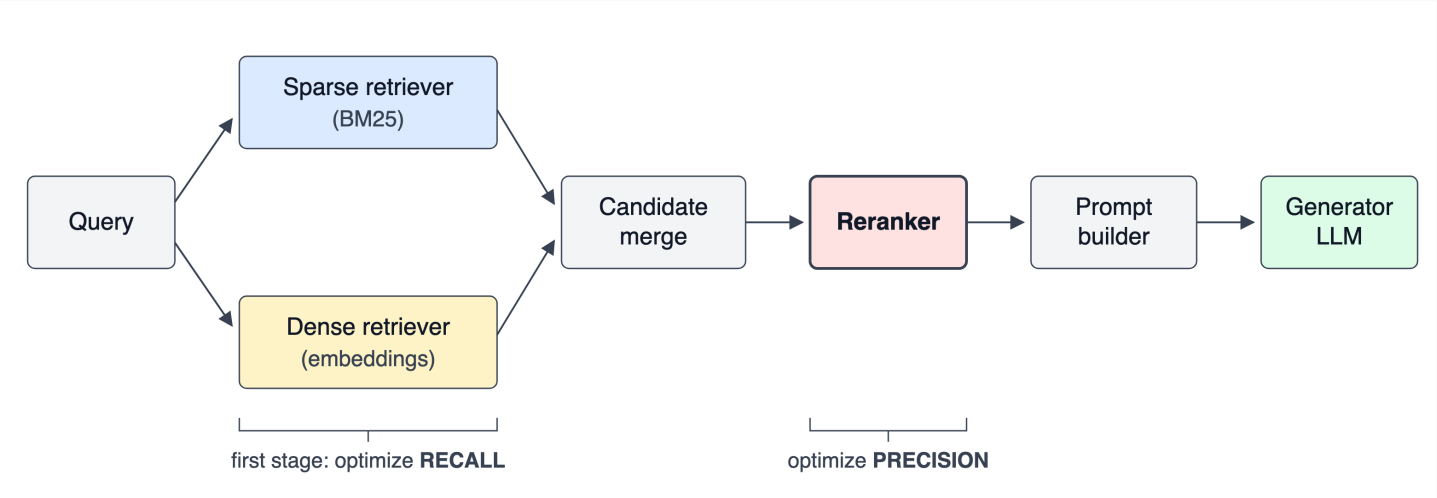
Why Retrieval? Three Finance-Shaped Reasons

1. **Staleness.** Weights frozen months ago; yesterday's earnings call isn't in them. Facts here change quarterly, sometimes by the minute.
2. **Hallucination.** Quote a retrieved document, and the invented $8.4B (Part 1) gets caught.
3. **Auditability.** An answer with a citation is something compliance can check. "The model felt bullish" is not a defensible rationale.

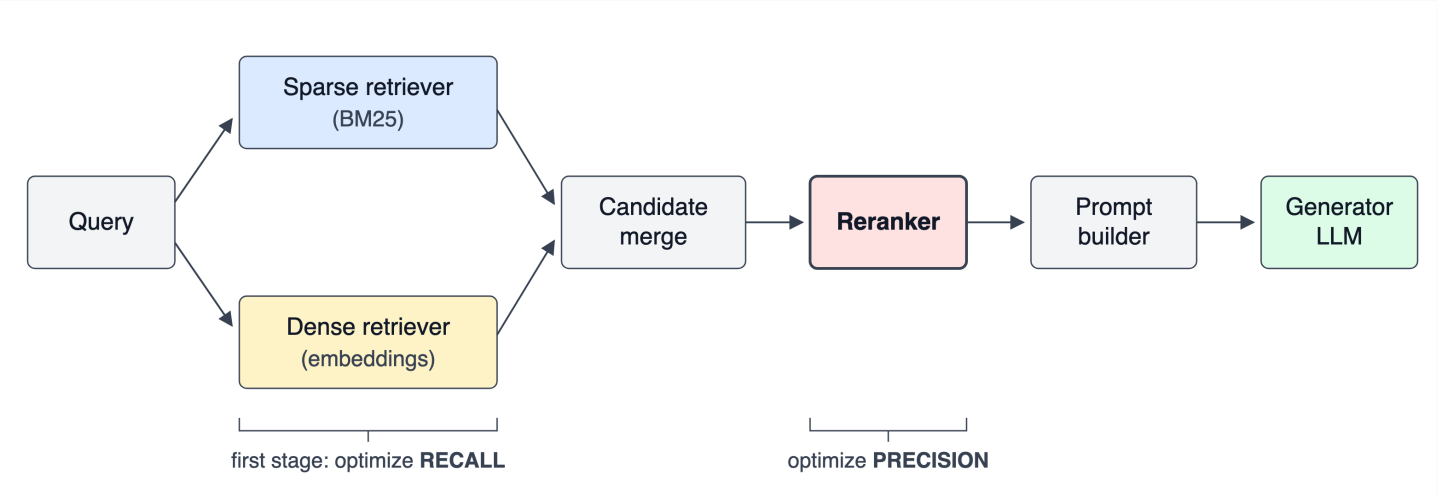


RAG = switch the model from a closed-book to an **open-book exam**. The hard part: finding the right page, fast.

The Pipeline, End to End

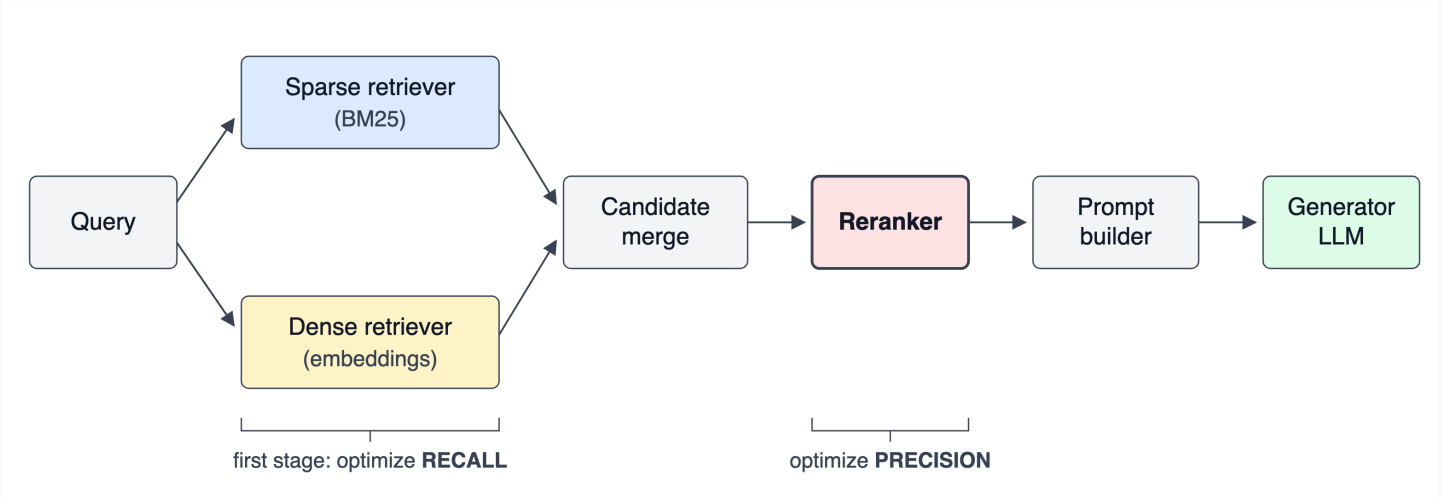


The Pipeline, End to End



First stage optimizes **recall@k** (did the right chunk make the top-k window the generator reads?); reranking optimizes **precision** at the top. Retrieval quality sets a *ceiling* on answer quality.

The Pipeline, End to End

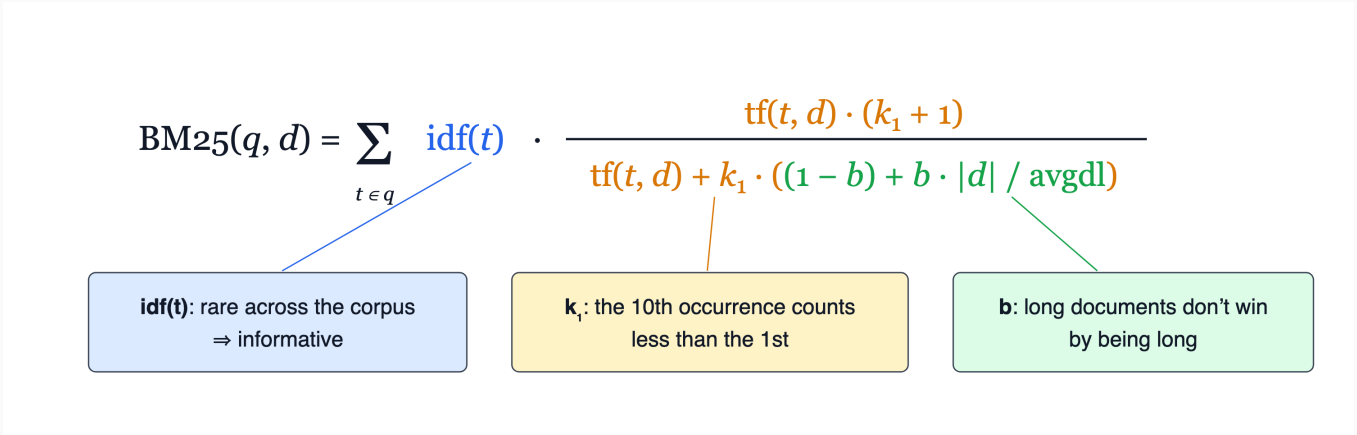


The original formulation (Lewis et al., 2020): marginalize the answer over retrieved documents,

$$p(y \mid x) \approx \sum_{z \in \text{top-}k} \underbrace{p_\eta(z \mid x)}_{\text{retriever}} \underbrace{p_\theta(y \mid x, z)}_{\text{generator}}$$

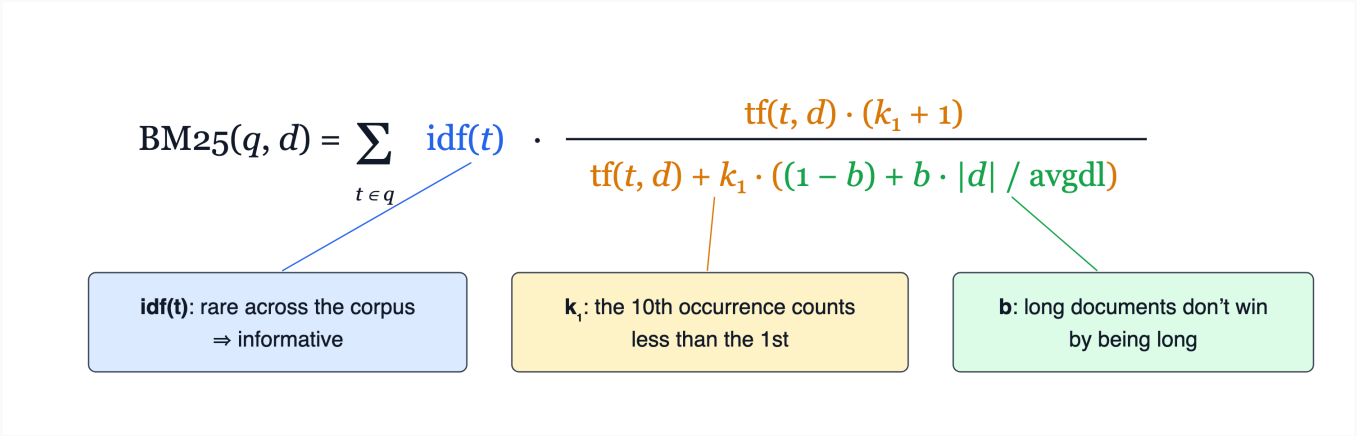
First stage optimizes **recall@k** (did the right chunk make the top-k window the generator reads?); reranking optimizes **precision** at the top. Retrieval quality sets a *ceiling* on answer quality.

Sparse Retrieval: a 50-Year-Old Idea, Still Standing



Two fixes over TF-IDF: saturation (k_1: the 10th "revenue" $\neq 10\times$ the 1st) and length normalization (b).

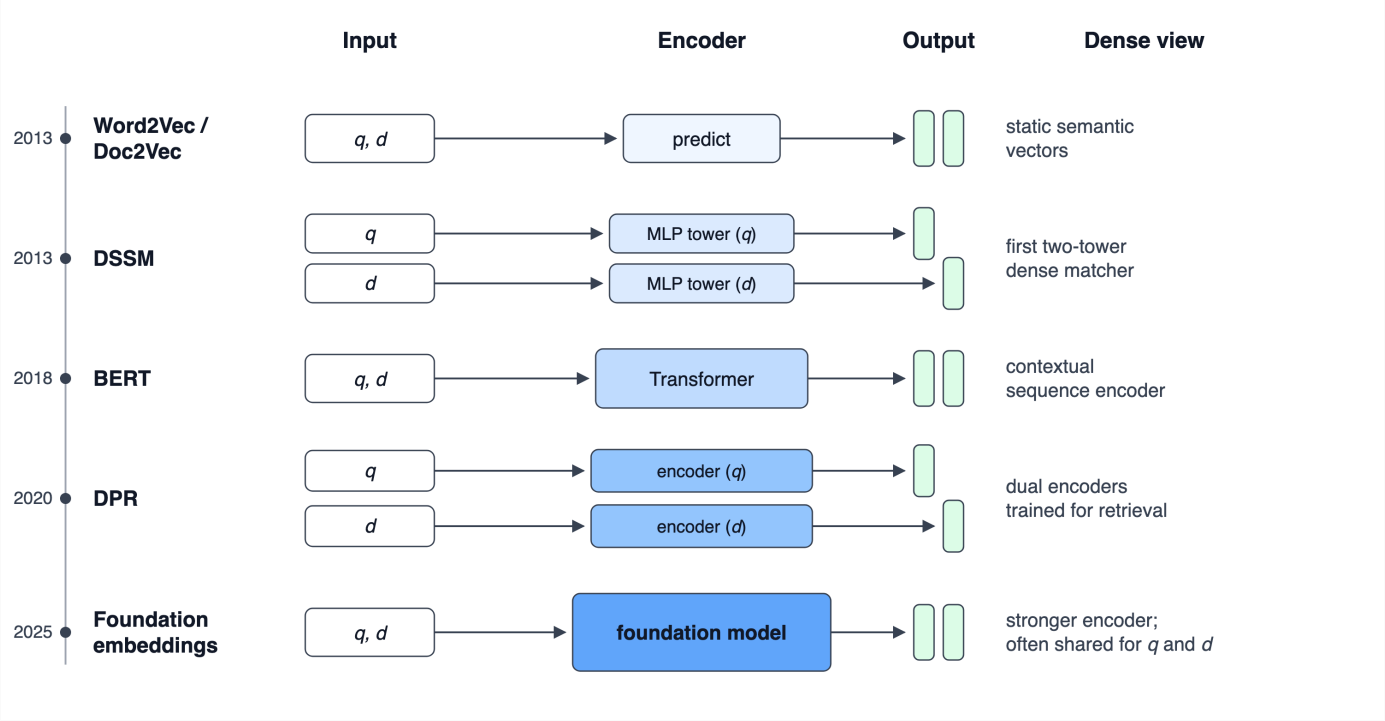
Sparse Retrieval: a 50-Year-Old Idea, Still Standing



Two fixes over TF-IDF: saturation (k_1: the 10th "revenue" $\neq 10\times$ the 1st) and length normalization (b).

Why it survives in finance: tickers, CUSIPs, "Note 14" want *exact* match. Weakness: "margin compression" vs "profitability headwinds" share no words.

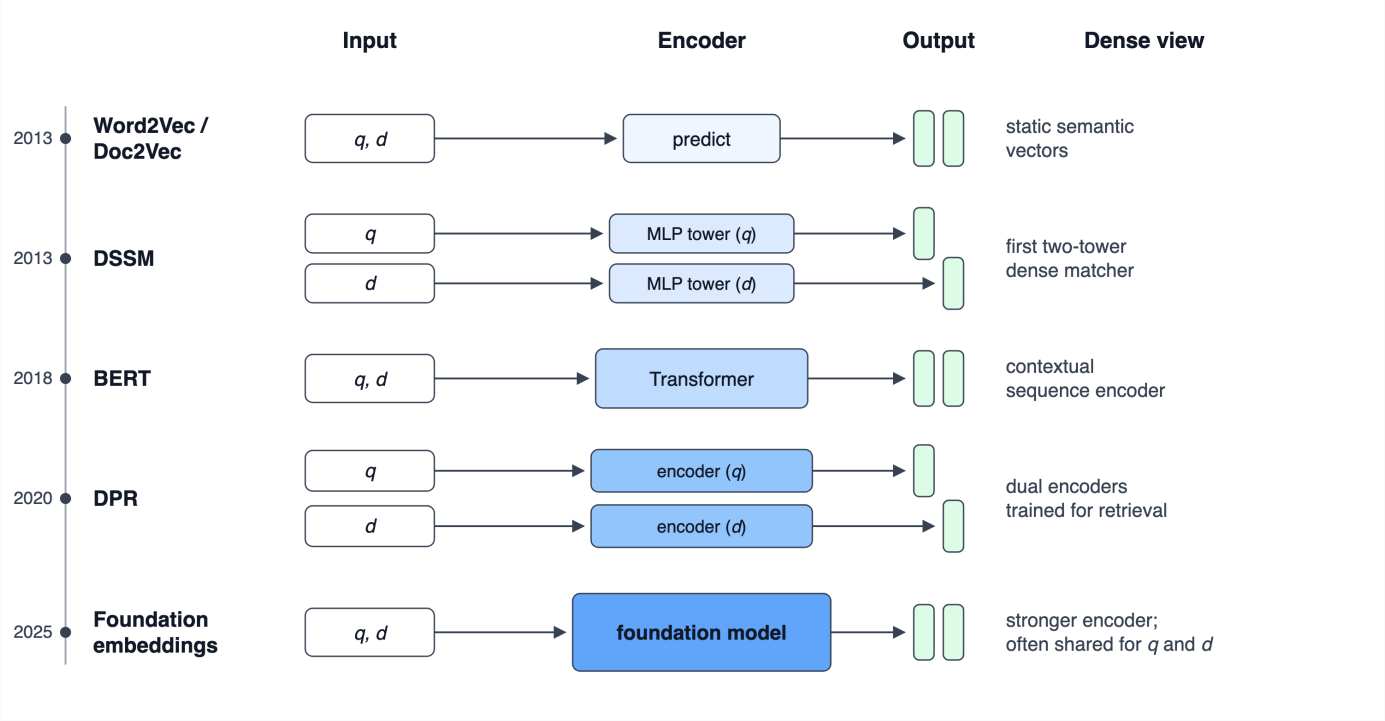
Dense Retrieval: From Words to Meaning



Texts $\rightarrow$ vectors; similar meaning $\rightarrow$ nearby points.

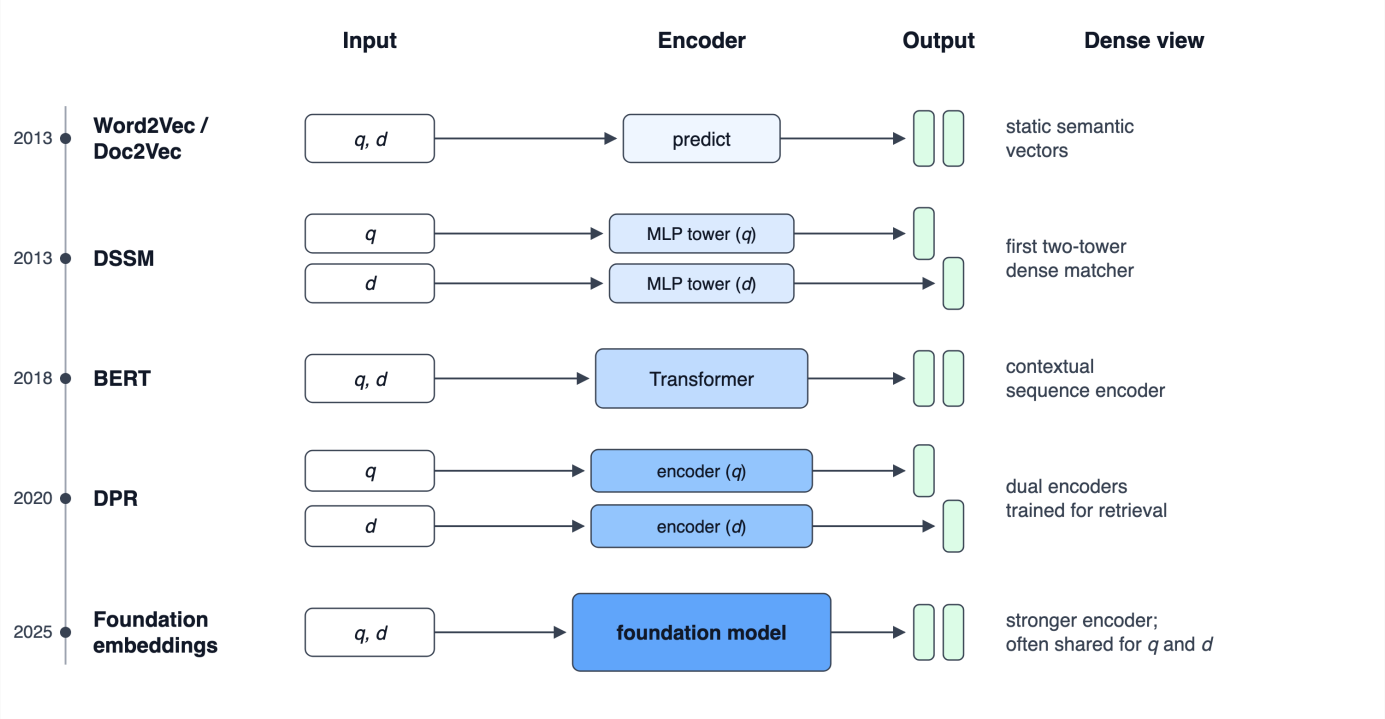
Retrieval = nearest-neighbor search.

Dense Retrieval: From Words to Meaning



Texts $\rightarrow$ vectors; similar meaning $\rightarrow$ nearby points. "Margin compression" and "profitability headwinds" become neighbors, no shared words required. Retrieval = nearest-neighbor search.

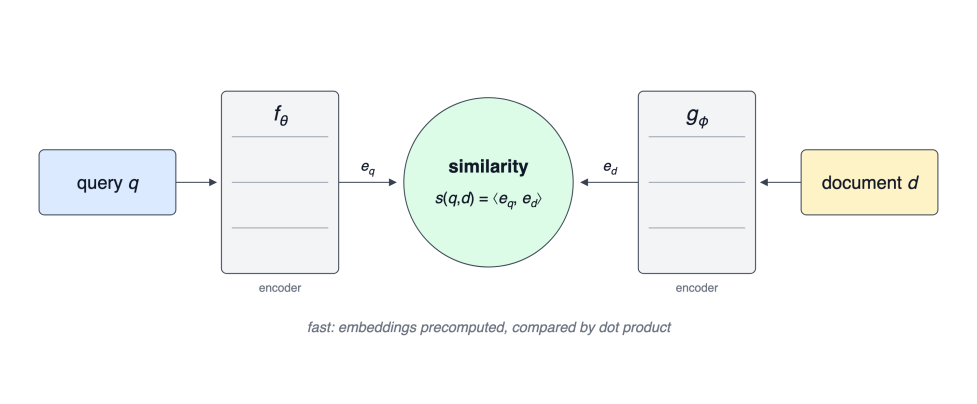
Dense Retrieval: From Words to Meaning



Texts $\rightarrow$ vectors; similar meaning $\rightarrow$ nearby points. "Margin compression" and "profitability headwinds" become neighbors, no shared words required. Retrieval = nearest-neighbor search.

Finance caveat: on FinMTEB (64 financial embedding tasks), general-benchmark rankings correlate weakly with financial performance; domain-adapted embedders consistently win (Tang & Yang, 2025).

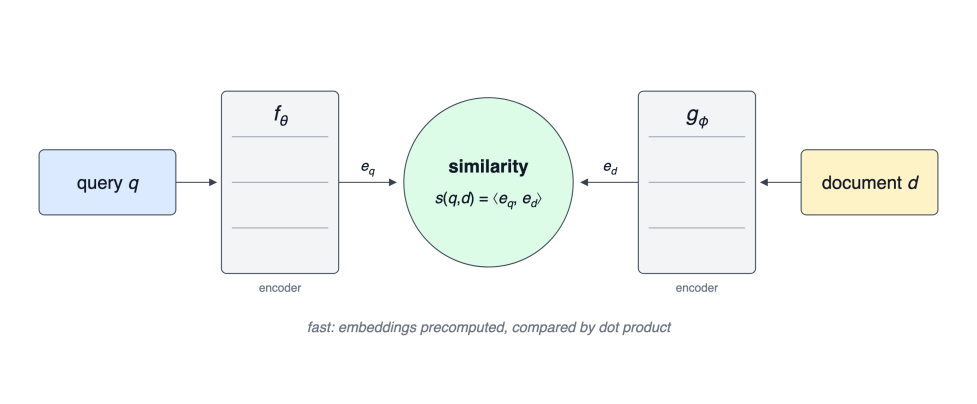
Training the Retriever: One Loss Function



$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp\big(s(q, d^+)/\tau\big)}{\exp\big(s(q, d^+)/\tau\big) + \sum_{d^- \in \mathcal{N}} \exp\big(s(q, d^-)/\tau\big)}$$

Pull the positive d^+ close, push negatives d^- away; τ is a temperature controlling how sharply the softmax concentrates (van den Oord et al., 2018).

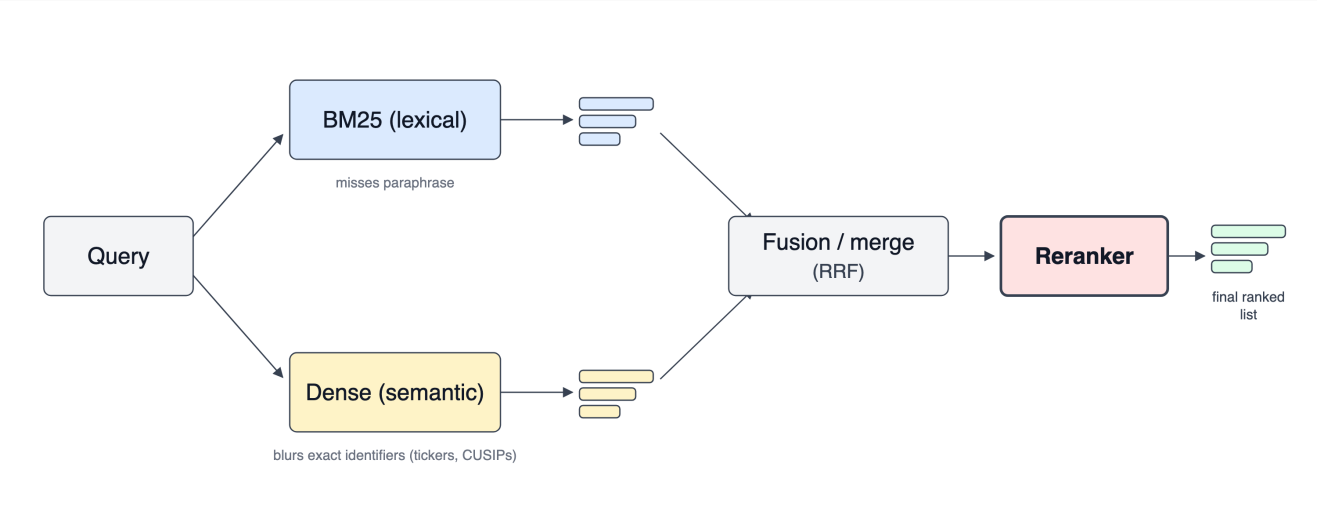
Training the Retriever: One Loss Function



$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp\big(s(q, d^+)/\tau\big)}{\exp\big(s(q, d^+)/\tau\big) + \sum_{d^- \in \mathcal{N}} \exp\big(s(q, d^-)/\tau\big)}$$

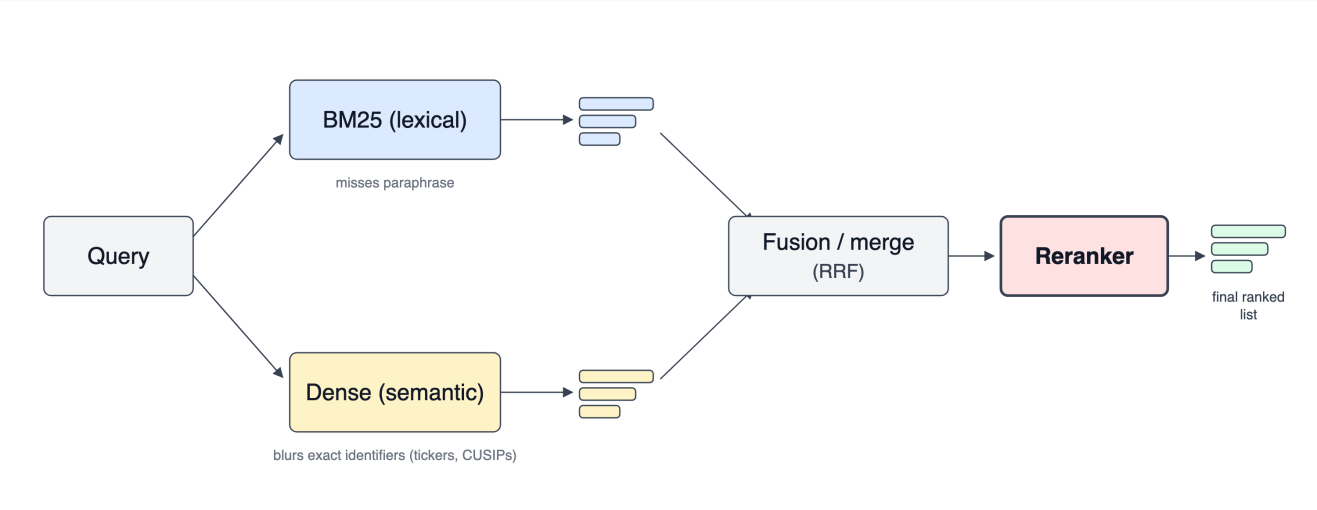
Pull the positive d^+ close, push negatives d^- away; τ is a temperature controlling how sharply the softmax concentrates (van den Oord et al., 2018). **Hard negatives** sharpen the boundary, and finance is full of them: same sentence, wrong fiscal year.

Hybrid Retrieval + Reranking



- **Hybrid:** complementary failure modes $\Rightarrow$ merge candidates (e.g. reciprocal rank fusion).

Hybrid Retrieval + Reranking



- **Hybrid:** complementary failure modes $\Rightarrow$ merge candidates (e.g. reciprocal rank fusion).
- **Rerank:** a slower model reads query + candidate *together*, reorders the top-k.
- Same shape as a trading stack: cheap screen over the universe, expensive diligence on the shortlist.

Where Financial Documents Make Retrieval Hard

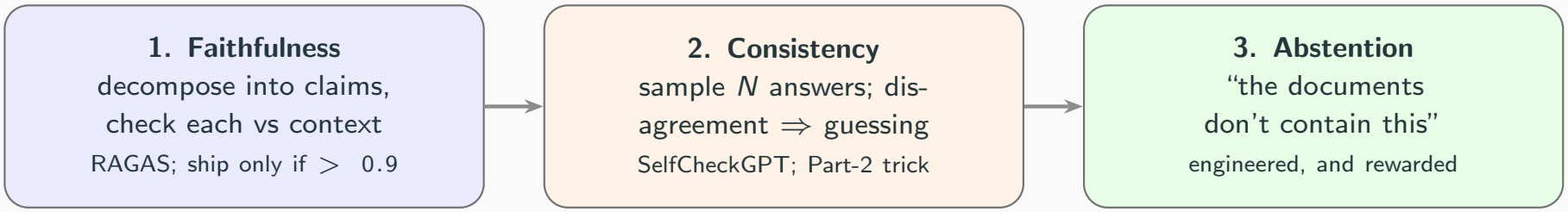
1. **Chunking is a modeling decision.**
   - 256–512 tokens: empirical sweet spot
   - Split on document structure (Items, Notes), keep the section path as metadata
2. **Tables hold the numbers.**
   - Serialize rows to sentences, or route to SQL
   - Raw table-row embeddings are noise
   - TAT-QA: QA over table+text from real reports; $\sim$42% of questions need arithmetic (Zhu et al., 2021)

Where Financial Documents Make Retrieval Hard

1. **Chunking is a modeling decision.**
   - 256–512 tokens: empirical sweet spot
   - Split on document structure (Items, Notes), keep the section path as metadata
2. **Tables hold the numbers.**
   - Serialize rows to sentences, or route to SQL
   - Raw table-row embeddings are noise
   - TAT-QA: QA over table+text from real reports; ∼42% of questions need arithmetic (Zhu et al., 2021)
3. **Charts hide guidance.**
   - FinMME: best multimodal models ∼50% on financial charts (GPT-4o 46.6%)
4. **"Just use long context"?**
   - Cost (Part 1) + lost-in-the-middle (Liu et al., 2023): models miss what's buried mid-context, and *footnotes live in the middle*

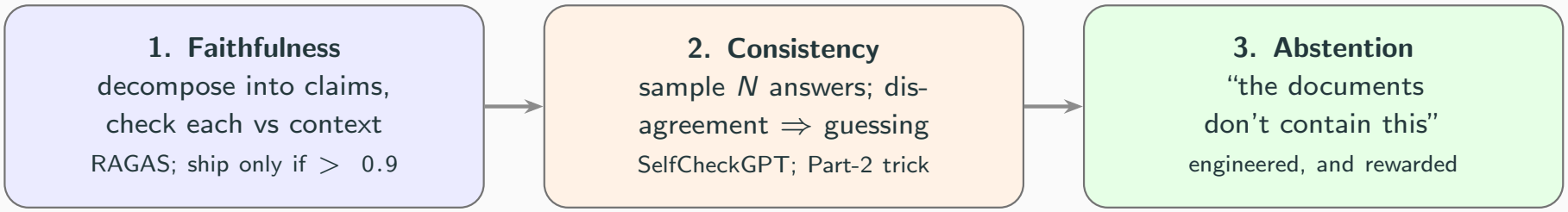
Retrieval isn't only about freshness: it's an **attention-focusing device** for documents built to bury things.

Is the Answer Actually Grounded?



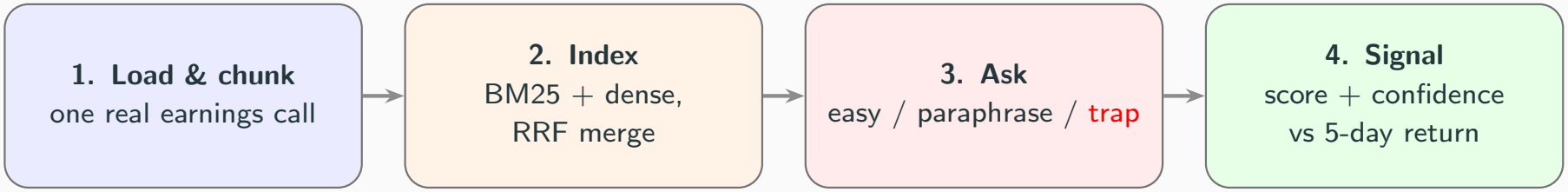
- A model can cite chunk 47 *and contradict it*: citation $\neq$ faithfulness.

Is the Answer Actually Grounded?



- A model can cite chunk 47 *and contradict it*: citation $\neq$ faithfulness.
- In finance, a confident wrong answer is strictly worse than a refusal. Build the refusal path.
- All three defenses run in the demo, next.

Demo: Earnings Call → Confidence-Scored Signal



Self-contained Colab notebook: paste an API key and run it yourself.

The link is on the tutorial website.

One call, one number: a *single example*, not an evaluation. Doing this properly → Hachem's backtesting part.

Part 4: Many Models

Debate, search, and learned orchestration

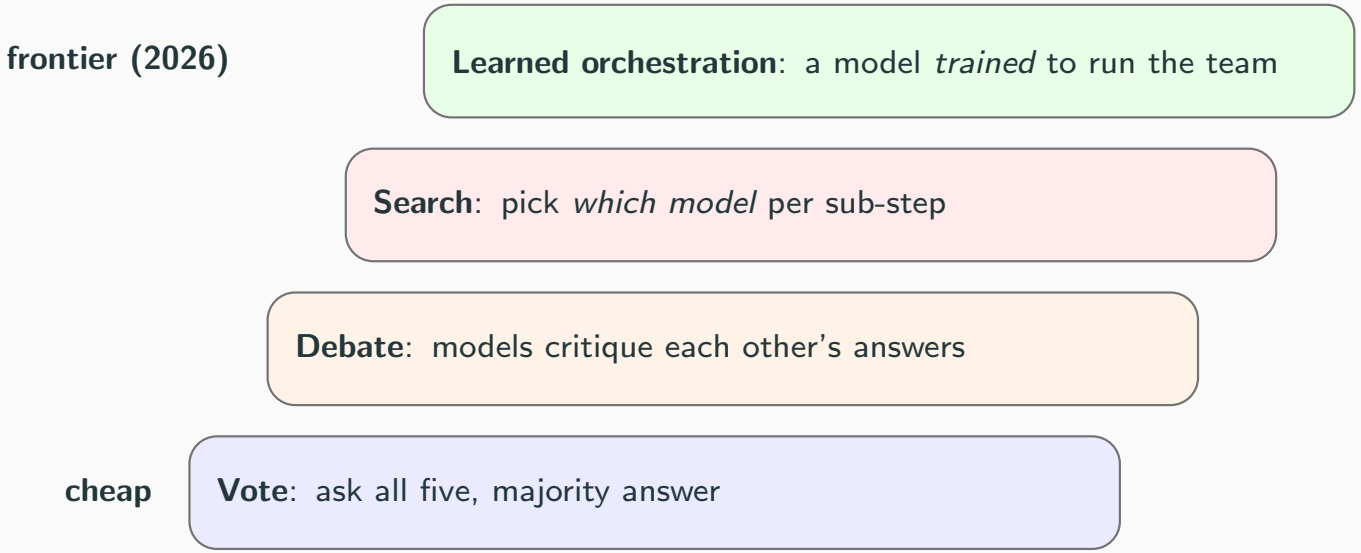
One Hard Query, Five Models. What Do You Do?

pick one and trust it / ask all five, majority vote / make them debate

search over them / train a model to orchestrate them

Which would you try first?

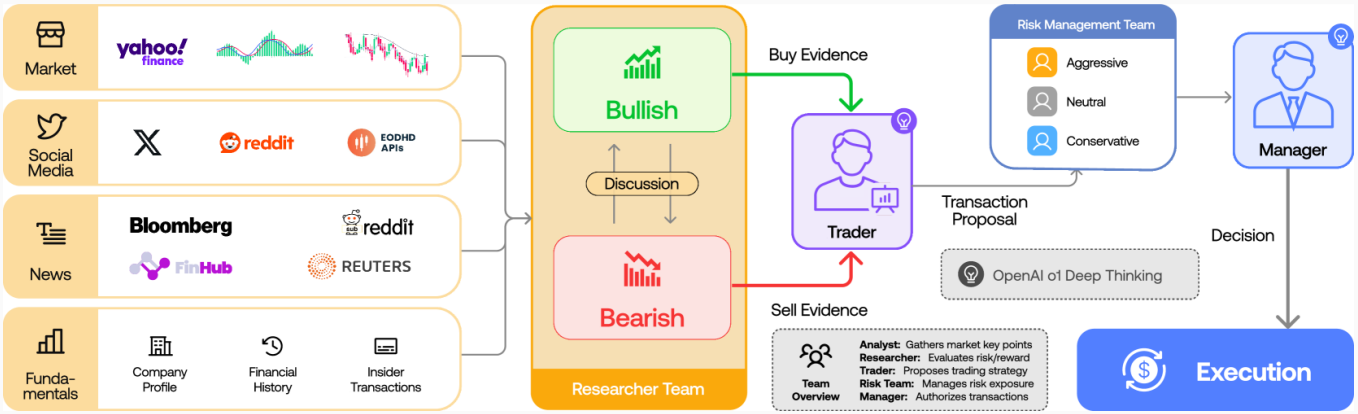
One Hard Query, Five Models. What Do You Do?



Which would you try first?

The recurring question for every option: when does coordination **beat the best single model**, and when is it a waste of compute?

Debate: Does Arguing Actually Help?

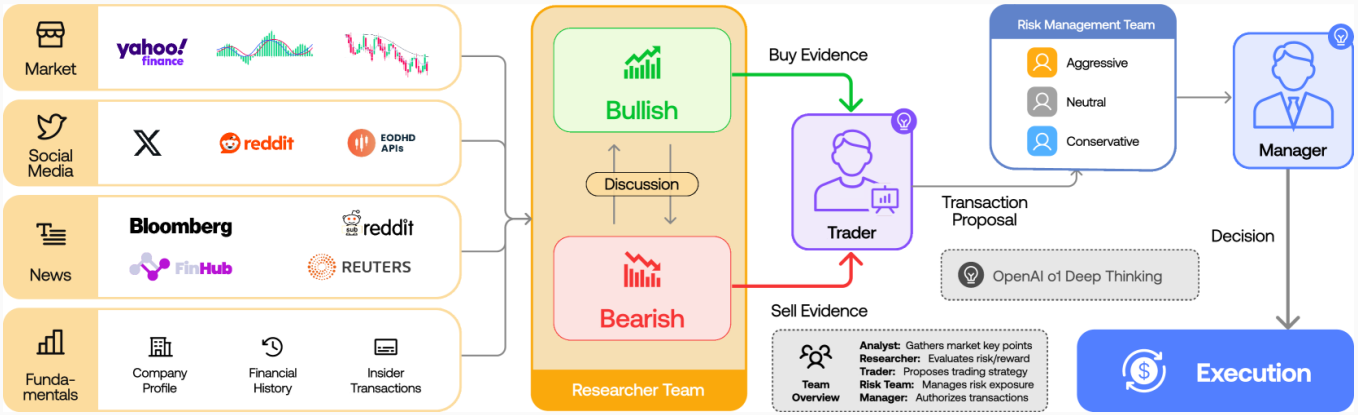


TradingAgents: analyst agents → bull/bear researcher debate → trader → risk team (Xiao et al., 2024, arXiv:2412.20138).

For: multi-agent debate lifts reasoning accuracy (+7–16 points by task, Du et al., 2023).

Against: weak debaters overturn a wrong consensus ∼**3.6%** of the time vs ∼**30–34%** for the strongest; structure barely matters (arXiv:2511.07784). *Much of "debate" is ensembling.*

Debate: Does Arguing Actually Help?



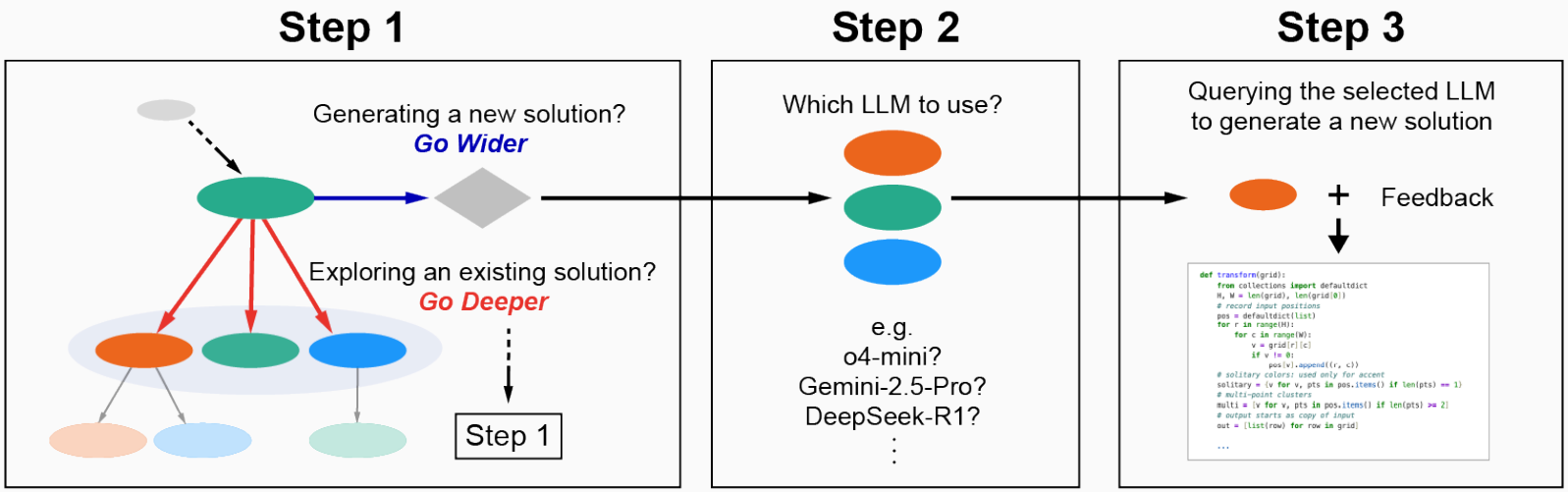
TradingAgents: analyst agents → bull/bear researcher debate → trader → risk team (Xiao et al., 2024, arXiv:2412.20138).

For: multi-agent debate lifts reasoning accuracy (+7–16 points by task, Du et al., 2023).

Against: weak debaters overturn a wrong consensus ∼**3.6%** of the time vs ∼**30–34%** for the strongest; structure barely matters (arXiv:2511.07784). *Much of "debate" is ensembling.*

In finance the debate transcript is the product: an **audit trail** of the case for and against the trade.

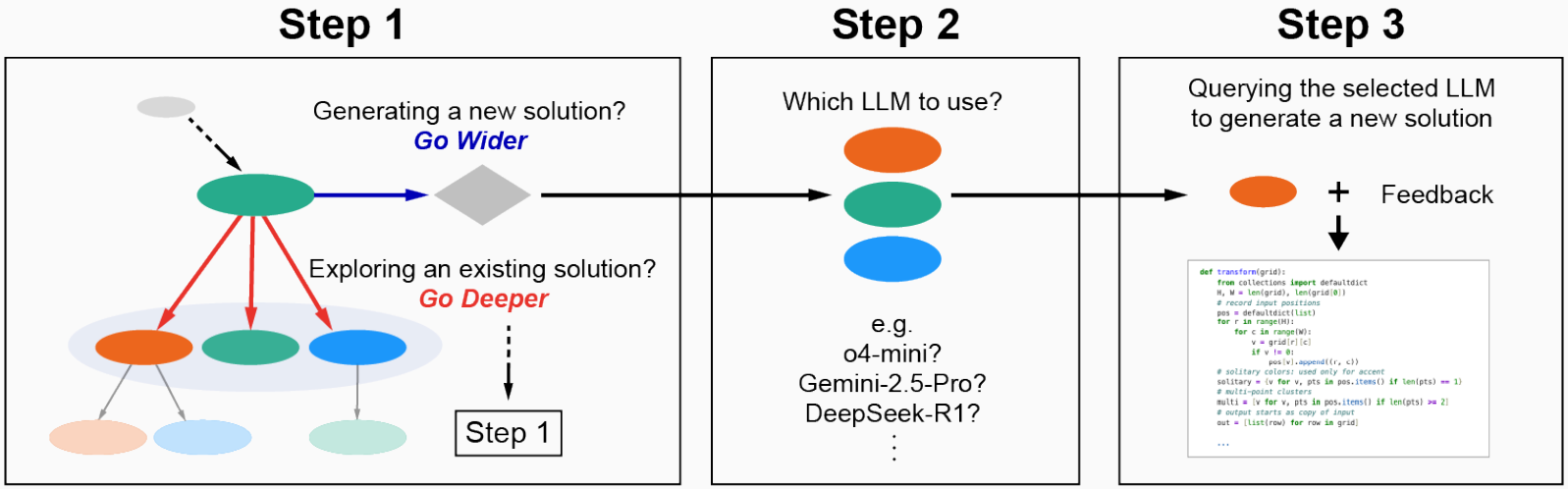
Search Over Models: AB-MCTS



Multi-LLM AB-MCTS: wider vs deeper, *and which model*, at every node (Sakana AI, 2025; open source as TreeQuest).

- Three levers at every node: **propose** a new solution, **refine** an existing one, and **route** (which model does it).

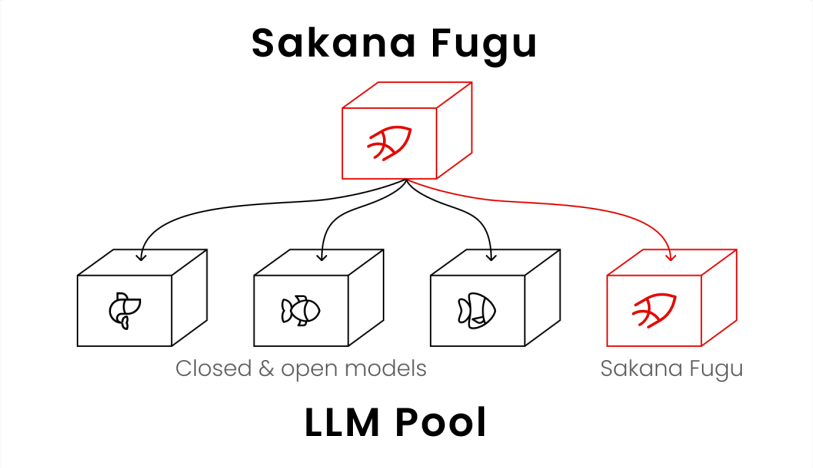
Search Over Models: AB-MCTS



Multi-LLM AB-MCTS: wider vs deeper, *and which model*, at every node (Sakana AI, 2025; open source as TreeQuest).

- Three levers at every node: **propose** a new solution, **refine** an existing one, and **route** (which model does it).
- ARC-AGI-2: a 3-model pool $>$**30%** (Pass@250) vs $\sim$23% for the best single model; some problems solved that *no* single model solves.
- One model's failed attempt becomes another model's hint: measured, not hoped-for, complementarity.

Learned Orchestration: Sakana Fugu



Fugu: an orchestrator model over a swappable LLM pool, including itself (Sakana AI, June 2026; arXiv:2606.21228).

- A model *trained* (fine-tuning + evolution + RL) to decompose, delegate, verify, synthesize, one API.
- Fugu Ultra reports parity with frontier single models on engineering / science / reasoning benchmarks.
- Pitch = resilience: a swappable pool, so no single-vendor dependency.

The pattern: coordination logic moved **prompt** → **search loop** → **trained weights**, the same migration CoT made in Part 2.

Part 5: Measuring LLMs in Finance

Benchmarks, numbers, and FinChain

How Good Are They, Really?

Benchmark	Task	Best LLM
FinQA	numerical QA on filings	GPT-4: ∼68%
FinanceBench	expert questions on 10-Ks/10-Qs	GPT-4-Turbo: 19%→85% (by retrieval setup!)
XFinBench	graduate-level, multimodal	o1: 67.3%
FinMME	financial charts	best models: ∼50%

Where human baselines exist: **10–25 points below them**. And FinanceBench: the *retrieval setup* moves GPT-4-Turbo from 19% to 85%: Part 3 in one table row.

Two Problems with Every Benchmark You Just Saw

Problem 1: contamination

Static, public for years $\Rightarrow$ in the training data.

Score $=$ capability $+$ memorization, in unknown proportions.

Problem 2: final-answer grading

Right number via wrong reasoning (canceling sign errors, memorized patterns) still counts.

Unauditable reasoning is undeployable here.

Two Problems with Every Benchmark You Just Saw

Problem 1: contamination

Static, public for years ⇒ in the training data.

Score = capability + memorization, in unknown proportions.

Problem 2: final-answer grading

Right number via wrong reasoning (canceling sign errors, memorized patterns) still counts.

Unauditable reasoning is undeployable here.

The wish list

(a) Questions **generated fresh** ⇒ cannot have leaked. (b) Every **intermediate step machine-checkable** (Part 2's lesson). ⇒ **FinChain**.

FinChain: Symbolic, Executable Financial Reasoning

FinChain template (Compound Interest)

#Question: `investor` invested `principal` in `project`, growing at `rate` %/yr compounded annually over `time` years. Compute the `compound interest`.

#Variables: `principal` = range(1000, 5000); `rate` = uniform(2, 10); `time` = range(1, 5)

#Executable CoT: Step 1: `amount` $=$ `principal` $\times \left(1 + \frac{\texttt{rate}}{100}\right)^{\texttt{time}}$ Step 2: `CI` $=$ `amount` $-$ `principal`

FinChain, Fig. 1 (Xie et al., 2026; https://mbzuai-nlp.github.io/finchain/).

- Sample the slots $\rightarrow$ a **fresh instance** with a **fully verified step-by-step solution**, on demand. Contamination-free and step-checkable by construction.

FinChain: Symbolic, Executable Financial Reasoning

FinChain template (Compound Interest)

#Question: `investor` invested `principal` in `project`, growing at `rate` %/yr compounded annually over `time` years. Compute the `compound interest`.

#Variables: `principal` = range(1000, 5000); `rate` = uniform(2, 10); `time` = range(1, 5)

#Executable CoT: Step 1: `amount` $= \texttt{principal} \times \left(1 + \frac{\texttt{rate}}{100}\right)^{\texttt{time}}$ Step 2: `CI` = `amount` − `principal`

FinChain, Fig. 1 (Xie et al., 2026; `https://mbzuai-nlp.github.io/finchain/`).

- Sample the slots → a **fresh instance** with a **fully verified step-by-step solution**, on demand. Contamination-free and step-checkable by construction.
- **12** domains, **58** topics, **290** templates, **2,900** regenerable cases. 10 financial experts reviewed all templates; 29/290 flagged and fixed.
- **ChainEval** grades the *steps*, not the answer: a step counts only if its number is right (within **5%**). Tracks expert judgement at $\rho = 0.655$ vs 0.43 for text overlap.

What FinChain Finds

All 25 models, step-level

Model	ChainEval	Final ans.
GPT-5-mini	**67.2**	80.3
GPT-5	66.6	82.0
Gemini 2.5 Pro	66.0	84.3
GPT-4.1	65.3	**84.7**
DeepSeek-R1	51.2	29.0

- **The mini beats the flagship** on step quality; $\sim$85% right answers vs $\sim$67% verifiable reasoning: the auditability gap, quantified.

What FinChain Finds

All 25 models, step-level

Model	ChainEval	Final ans.
GPT-5-mini	**67.2**	80.3
GPT-5	66.6	82.0
Gemini 2.5 Pro	66.0	84.3
GPT-4.1	65.3	**84.7**
DeepSeek-R1	51.2	29.0

Open 7–8B: tuned vs general

Model (7–8B)	ChainEval
Qwen-2.5-Instruct (general)	**60.4**
Mathstral (math-tuned)	59.9
Fin-R1 (best finance-tuned)	58.1
Finance-Qwen (finance-tuned)	34.6
MetaMath (math-tuned)	7.9

- **The mini beats the flagship** on step quality; ∼85% right answers vs ∼67% verifiable reasoning: the auditability gap, quantified.
- Two extra reasoning steps separate a pass from a failure. The vendor-demo question: "what happens at step four?"

The L&M lesson, refined

Domain matters for **vocabulary and data** (FinBERT, RAG). Reasoning **transfers from math**, and narrow tuning can erode it.

Takeaways

1. An LLM is a **next-word predictor**: no database inside; invention and overconfidence are design consequences, not bugs.
2. Reasoning $\neq$ one thing. The **deductive core is verifiable**, and cheap **verification** powers every reasoning model (o1, R1, GRPO).

Thank you. Salem Lahlou salem.lahlou@mbzuai.ac.ae `https://lahlou.org`

Materials: this deck, the RAG notebook, and FinChain, all on the tutorial page.

Takeaways

1. An LLM is a **next-word predictor**: no database inside; invention and overconfidence are design consequences, not bugs.
2. Reasoning $\neq$ one thing. The **deductive core is verifiable**, and cheap **verification** powers every reasoning model (o1, R1, GRPO).
3. **RAG turns fluent into auditable**: ground every number, measure *recall@k* first, engineer the abstention path.
4. Multi-model coordination works when members are **strong and diverse**; the logic is migrating *prompt* $\rightarrow$ *search* $\rightarrow$ *weights*.

Thank you. Salem Lahlou salem.lahlou@mbzuai.ac.ae `https://lahlou.org`

Materials: this deck, the RAG notebook, and FinChain, all on the tutorial page.

Takeaways

1. An LLM is a **next-word predictor**: no database inside; invention and overconfidence are design consequences, not bugs.
2. Reasoning $\neq$ one thing. The **deductive core is verifiable**, and cheap **verification** powers every reasoning model (o1, R1, GRPO).
3. **RAG turns fluent into auditable**: ground every number, measure *recall@k* first, engineer the abstention path.
4. Multi-model coordination works when members are **strong and diverse**; the logic is migrating *prompt* $\rightarrow$ *search* $\rightarrow$ *weights*.
5. On verifiable financial reasoning, **general beats finance-tuned** (FinChain). Domain lives in the *data*, not the reasoning engine.
6. Best models: $\sim$85% answers, $\sim$67% verifiable steps, cliffs at 4 steps $\Rightarrow$ **assist and verify, don't automate**.

Thank you. Salem Lahlou salem.lahlou@mbzuai.ac.ae `https://lahlou.org`

Materials: this deck, the RAG notebook, and FinChain, all on the tutorial page.

Agentic AI for Financial Markets

Trading, Research, and Deployment

Part 3 of the KDD 2026 Tutorial “Generative AI and Large Language Models (LLMs) for Financial Markets”

Zangir Iklassov · MBZUAI Abu Dhabi

zangir.iklassov@mbzuai.ac.ae August 2026

Official MBZUAI faculty photograph

Assistant Teaching Professor of Machine Learning

Mohamed bin Zayed University of Artificial Intelligence

Reinforcement learning (RL) + combinatorial optimization

Learning decisions that respect hard operational constraints.

Large language model (LLM) reasoning + search

Generating, testing, and refining multiple solution paths.

Agentic systems for real decisions

Connecting models to tools, evidence, feedback, and oversight.

Current title and biography: official MBZUAI faculty profile, accessed 7 Aug 2026.

From Learned Policies to Agentic Research

On the Study of Curriculum Learning for Inferring Dispatching Policies on the Job Shop Scheduling

Curriculum RL learns dispatching policies from easier to harder instances.

IJCAI 2023

Reinforcement Learning for Solving Stochastic Vehicle Routing Problem

A policy builds routes under uncertain demand and travel cost.

ACML 2023 / PMLR 2024

Self-Guiding Exploration for Combinatorial Problems

An LLM explores several solution paths, then executes and refines them.

NeurIPS 2024

Bridgewater's AIA: How to Read the Public Numbers

AIA (Artificial Investment Associate) is Bridgewater's machine-powered macro strategy; **AUM** is assets under management.

$1.6B
exact Bridgewater disclosure
1 July 2024

"almost $2B"
Bloomberg report
unnamed sources

~$4.5B
Reuters estimate of assets under management, 1 July 2026

Was +8.1% good? The January–June 2026 raw return was competitive: AIA **8.1%**; PivotalPath hedge-fund composite **7.3%**; its 29-year first-half average **5.15%**; S&P 500 (broad US large-cap equities) **9.67%**. Missing risk, exposure, and fee data prevent a risk-adjusted verdict. Reuters separately reports **11.3% annualized** over a longer record starting in late 2023; Bloomberg dates the external vehicle to July 2024. The gap is unexplained.

Sources: Bridgewater, 1 July 2024; Reuters, 1 July 2026; PivotalPath figures reported by Institutional Investor, 8 July 2026. Reuters and Bloomberg fund figures rely on unnamed sources.

Outline

1 · Agent foundations
planning, tools, memory, interaction, and two feedback methods

2 · Trading and analysis agents
FinRL-Meta → FinAgent → FINCON → TradingAgents

3 · Quantitative research systems
AlphaGen → AlphaAgent → R&D-Agent-Quant → AlphaEvo

4 · Industry use
adoption estimates and research agents, measured benefits and future deployment

5 · Closing
shared synthesis, resources, and questions

PART 01

What Is an Agent?

A policy–environment loop, with optional tools and memory

Financial reasoning → **Agents** Trade Research Industry Close

An Agent Is a Model That Acts

$$\underbrace{\textbf{Agent}}_{\text{decision process}} = \underbrace{\pi_\theta \leftrightarrow \mathrm{Env}}_{\text{policy–environment loop}} + \underbrace{[\mathcal{T} + M]}_{\text{optional tools + memory}}$$

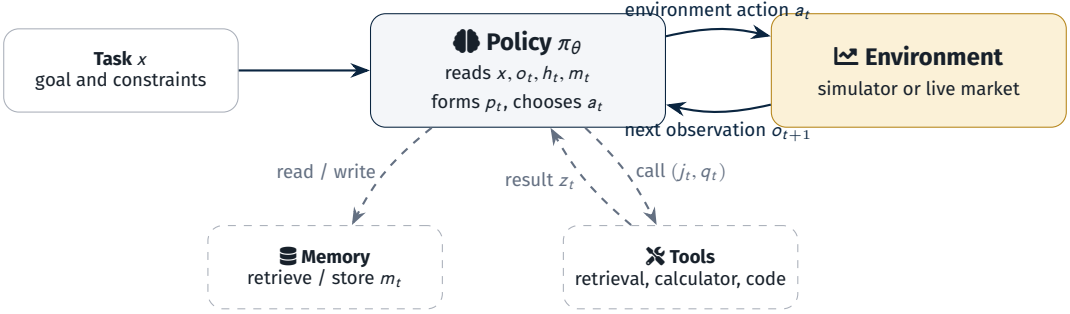
The fixed input x states the task. At step t, policy π_θ reads the observation o_t, history h_t, and retrieved memory m_t. It may form plan p_t, call a tool for evidence z_t, and then choose environment action a_t.

A numeric RL policy may omit explicit planning, tools, and memory; then $p_t = z_t = m_t = \varnothing$.

$$\begin{aligned} p_t &= \pi_\theta^{\mathrm{plan}}(x, o_t, h_t, m_t), \\ a_t &\sim \pi_\theta^{\mathrm{act}}(\cdot \mid x, o_t, h_t, \\ &\qquad\qquad m_t, p_t, z_t), \\ o_{t+1} &= \mathrm{Env}(o_t, a_t). \end{aligned}$$

The superscripts are two modes or prompts of one policy, not necessarily two trained models. $\sim$ means select or sample; the environment applies a_t and returns o_{t+1}.

The Agent Loop



A tool returns evidence z_t; it does not change the market. Only action a_t changes environment state. Then Env returns o_{t+1}: the next simulator state, or live fills and observations.

One Unified Agent Step

Common context: $s_t = (x, o_t, h_t, m_t)$ = fixed task, current observation, within-run history, and retrieved memory.

1. Plan

$$p_t = \pi_\theta^{\text{plan}}(s_t)$$

p_t says what to do next. If the policy plans implicitly, set $p_t = \varnothing$.

2. Call a tool

$$(j_t, q_t) = \pi_\theta^{\text{call}}(s_t, p_t),$$
$$z_t = \mathcal{T}_{j_t}(q_t).$$

j_t: tool; q_t: arguments; z_t: returned evidence. The call is optional and may repeat.

3. Act

$$a_t \sim \pi_\theta^{\text{act}}(\cdot \mid s_t, p_t, z_t),$$
$$o_{t+1} = \text{Env}(o_t, a_t).$$

a_t changes external state—for example, it submits an order. A tool call does not.

4. Record

$$h_{t+1} = h_t \oplus (p_t, j_t, q_t, z_t, a_t, o_{t+1}),$$
$$m_{t+1} = U(m_t, h_{t+1}, \text{score}_{t+1}).$$

$\oplus$: append. The optional score may be reward, error, test feedback, or human judgment.

Two Ways Agents Use Feedback

Let $s_t = (x, o_t, h_t, m_t)$. The same LLM policy π_θ can be reused with different prompts; its weights stay fixed.

ReAct

Yao et al. · ICLR 2023

Plan → call a tool → read its result → continue the same attempt

$$
\begin{aligned}
p_t &= \pi_\theta^{\text{plan}}(s_t), \\
(j_t, q_t) &= \pi_\theta^{\text{call}}(s_t, p_t), \\
z_t &= \mathcal{T}_{j_t}(q_t), \\
h_{t+1} &= h_t \oplus (p_t, j_t, q_t, z_t).
\end{aligned}
$$

The tool result z_t becomes evidence for the next plan. A separate action a_t is needed to change the environment.

Self-Refine

Madaan et al. · NeurIPS 2023

Draft → critique the draft → revise that same draft

$$
\begin{aligned}
y_0 &= \pi_\theta^{\text{draft}}(x), \\
c_i &= \pi_\theta^{\text{critique}}(x, y_i), \\
y_{i+1} &= \pi_\theta^{\text{refine}}(x, y_i, c_i).
\end{aligned}
$$

The model produces its own critique c_i and rewrites output y_i. It does not need an external tool or environment action.

x: fixed task; t: interaction step; i: draft revision; j_t: tool; q_t: tool arguments; z_t: returned evidence; $\oplus$: append.

Why Markets Are a Hard Test for Agents

mixed delays
fills: seconds
thesis: months

costly
mistakes are
real money

adversarial
others adapt
to exploit you

shifting
behavior changes
across regimes

Market microstructure explains why: prices respond to informed trading, liquidity is state-dependent, and other participants adapt to *your* actions. An agent in a market is not solving a puzzle — it is **playing against other players**. A **market regime** is a recurring condition with different behavior, such as a calm market versus a volatile crisis.

Three Financial-Agent Workflows

Forecast-driven

Predict returns, then allocate positions.

$$\widehat{r}_{t+1:t+h} = \pi_\theta^{\text{forecast}}(x, I_{\le t}),$$
$$a_t = \Gamma(\widehat{r}, \widehat{\Sigma}_t, C_t)$$

$I_{\le t}$: evidence known by date t; h: forecast horizon; $\widehat{r}$: predicted returns; $\widehat{\Sigma}_t$: estimated risk; C_t: constraints; Γ: allocator; a_t: positions sent to the environment.

Event-reactive

Detect new information, then update the trade.

$$e_t = \phi(I_t, I_{t-1}),$$
$$a_t \sim \pi_\theta^{\text{act}}(\cdot \mid x, o_t, e_t)$$

I_t, I_{t-1}: current and prior evidence; ϕ: change detector; e_t: detected event; o_t: market and portfolio observation; a_t: resulting trade or position. Evidence may be prices, news, or filings.

Research loop

Propose a rule, test it, and retain the result.

$$\text{hyp}_k \sim \pi_\theta^{\text{idea}}(\cdot \mid x, \mathcal{K}, \mathcal{H}_{<k}),$$
$$\text{code}_k = \mathcal{T}_{\text{code}}(\text{hyp}_k),$$
$$\text{score}_k = \text{Eval}(\text{code}_k; \mathcal{D}),$$
$$\mathcal{H}_k = \mathcal{H}_{<k} \cup \left\{ \begin{matrix} (\text{hyp}_k, \text{code}_k, \\ \text{score}_k) \end{matrix} \right\}.$$

$\mathcal{H}_{<k}$ is the set of earlier $(\text{hyp}_i, \text{code}_i, \text{score}_i)$ records. $\mathcal{K}$: domain knowledge; $\mathcal{D}$: evaluation data; $\cup$: add the new tested record.

These are workflows, not mutually exclusive system types: one agent may use all three at different stages.

How I Built the Literature Map

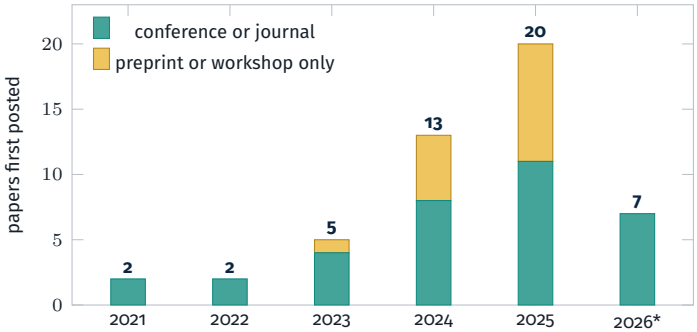
Included

- A paper presenting or directly testing a named financial-agent architecture.
- The system repeatedly observes evidence and uses memory, reflection, tools, code, role coordination, or RL feedback.
- Its output is a trade, portfolio, alpha factor, research report, or dedicated agent evaluation.

Excluded

- One-pass prediction, sentiment classification, or question answering.
- General finance LLMs and surveys without a financial-agent method.
- Blogs or repositories without a paper.

The Field Accelerated After 2023



49
included papers

40
first appeared from 2024 onward

34
now in a conference or journal

Our review of 49 papers; each appears once in the year first posted. Publication status checked 7 August 2026; the 2026 count is partial.

Four Kinds of Output in the 49-Paper Review

Alpha factor: a formula intended to rank assets by later return. **R&D**: research and development. Each paper is assigned once by its main emphasis.

Trading or portfolio — 24 papers

The system's main output is a market action—buy, sell, or hold—or a vector of portfolio weights. Examples retained in this tutorial: FinAgent, FINCON, and TradingAgents.

Evaluation or infrastructure — 9 papers

The main output is a reusable market environment, benchmark, toolchain, arena, or audit. It measures or enables agents rather than directly deciding a trade. FinRL-Meta is the infrastructure example retained here.

Alpha, factor, or R&D — 8 papers

The main output is a formula, factor set, predictive model, or next research experiment. Examples retained here: AlphaGen, AlphaAgent, R&D-Agent-Quant, and AlphaEvo.

Financial analysis or research — 8 papers

The main output is an evidence-backed report, valuation, forecast, or analytical answer. It can support a later decision without allocating capital itself.

Source: our review of 49 papers. Categories are mutually exclusive and assigned by each paper's primary output.

PART 02

Agents That Trade

Policies, environments, memory, tools, teams, analysis, and evaluation

Trading and Evaluation Systems in This Section

System	Primary output	What this paper adds to the sequence
FinRL-Meta	benchmark environments	reuse data converters, simulators, costs, and rolling tests
FinAgent	trades	text, price charts, reflection, and executable technical tools
FINCON	trades / portfolio	specialists + manager; language lessons from stronger versus weaker runs
TradingAgents	trades	bull–bear research debate followed by a risk debate

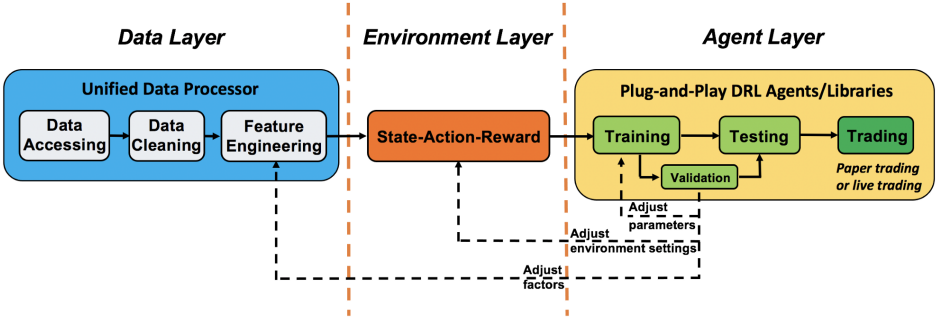
Sources: the four primary papers discussed in the following slides. Rows are ordered by the presentation, not by reported performance.

FinRL-Meta · Paper metadata and Core idea

FinRL-Meta: Market Environments and Benchmarks for Data-Driven Financial Reinforcement Learning, Xiao-Yang Liu et al., NeurIPS 2022 Datasets and Benchmarks Track.

FinRL-Meta builds the laboratory around financial reinforcement-learning agents: common data adapters, market environments, costs, constraints, rolling splits, benchmarks, and paper-trading interfaces.

FinRL-Meta · Framework



FinRL-Meta, Figure 2, archived paper p. 4.

FinRL-Meta · What goes in

A user specifies:

- a data source and asset universe;
- start and end dates and data frequency;
- features, such as OHLCV, technical indicators, fundamentals, news sentiment, or alternative data;
- an action and reward definition;
- transaction costs, margin and shorting permissions, and risk controls;
- one or more reinforcement-learning policies to train and compare.

The library described in the paper supports more than 30 market-data sources and three plug-in reinforcement-learning libraries: ElegantRL, Stable-Baselines3, and RLlib.

FinRL-Meta · How it works · 1–6

1. Access data. A common interface downloads the requested vendor data for the selected market, dates, and frequency.

2. Clean and align it. The data layer standardizes vendor formats, handles missing or erroneous records, joins sources by ticker and time, and computes selected features.

3. Create one environment. A Gym-style market simulator defines `reset`, `step`, and `reward`, plus the market clock, actions, costs, account limits, shorting rules, and risk control.

4. Plug in candidate policies. Several reinforcement-learning algorithms can now see the same state and interact under the same experiment specification.

5. Train, validate, and test. Policies are fit on the training environment, tuned or selected on validation data, and backtested on later data.

6. Return comparable records. The output is trades, account paths, and metrics generated under the chosen common rules, plus reusable data and environment code.

In the displayed DJIA benchmark, the framework runs the same quarterly A2C/PPO/DDPG selector described in the 2020 ensemble paper.

FinRL-Meta · Key formula or decision rule

FinRL-Meta does not introduce a new trading policy objective. A compact way to express its contribution is

$$(s_{t+1}, r_t) = \mathrm{Env}_\lambda(s_t, a_t),$$

where s_t is the current state, a_t is a policy's action, and λ is the complete experiment specification: data, clock, features, costs, constraints, reward, and risk control. Policies are meaningfully comparable only when they use the same λ.

For its reproduced ensemble benchmark,

$$j_k^* = \arg \max_{j \in \{\mathrm{A2C,PPO,DDPG}\}} \mathrm{Sharpe}(j; \mathcal{V}_k),$$

so the highest-validation-Sharpe policy trades the following quarterly window.

FinRL-Meta · Why it is different and Experiment

Earlier framework work asks, “How can I formulate and train this trader?” FinRL-Meta asks, “How can I put different traders into the same reproducible experiment?” Its main artifact is infrastructure, not a smarter policy.

For the compact benchmark used in the presentation:

- Source and universe: Yahoo Finance daily data for DJIA-30.
- Policies: A2C, PPO, and DDPG with quarterly validation-based selection.
- Training stated by the source: 1 April 2009 through 30 June 2019.
- Validation and testing stated by the source: 1 July 2020 through 31 March 2022 in quarterly rolling windows.
- Comparator: each individual policy and the DJIA index.

The paper does not explain how the interval from 1 July 2019 through 30 June 2020 is assigned. It should not be silently labelled as validation, embargo, or training.

FinRL-Meta · Reported results

Method	Annual return	Annual volatility	Sharpe	Calmar	MDD
Selector	25.9%	15.9%	1.53	2.27	-11.4%
A2C	23.3%	16.2%	1.37	1.97	-11.8%
PPO	13.1%	13.4%	0.99	0.88	-14.9%
DDPG	12.7%	15.0%	0.88	0.85	-14.9%
DJIA	19.7%	14.4%	1.32	1.74	-11.3%

FinAgent · Paper metadata and Core idea

A Multimodal Foundation Agent for Financial Trading: Tool-Augmented, Diversified, and Generalist, Wentao Zhang et al., ACM KDD 2024.

FinAgent asks fixed GPT-4 and vision-enabled GPT-4V to combine dated text, price-chart images, retrieved history, reflections on earlier price moves and trades, and deterministic technical tools before returning buy, sell, or hold.

FinAgent · What goes in

- Daily prices and news from Financial Modeling Prep for five US stocks and ETHUSD.
- A Kline, or candlestick, chart with indicators such as moving averages and Bollinger Bands.
- A trading chart showing earlier actions and the resulting portfolio path; both chart types are generated with `pyecharts`.
- Seeking Alpha expert guidance for the stock experiments.
- Current account state and an investor-preference prompt.
- Retrieved market-intelligence, price-reflection, and trade-reflection memories.

- Three rule-based signals: MACD crossover, KDJ with an RSI filter, and Z-score mean reversion.

Filings are not a Financial Modeling Prep input in the displayed experiment.

FinAgent · How it works · 1–4

1. Summarize today's evidence. GPT-4 extracts relevant news, sentiment, likely duration, and a compact market summary. It also writes separate retrieval queries rather than reusing the trading summary as a search query.

2. Retrieve similar history. The memory system uses those diversified queries to find older market intelligence from several perspectives, such as time horizon, direction, or source type.

3. Explain completed price moves. For an older event whose later prices are now known, GPT-4V reads the news summary and Kline chart and writes a low-level reflection connecting the evidence with the subsequent short-, medium-, and long-horizon move.

4. Review completed trades. GPT-4V examines roughly the previous 14 trading days, including past actions, reasons, and the trading chart, and writes a high-level lesson about successful and mistaken behavior.

5. Call auxiliary tools. The technical rules return trend, momentum, and mean-reversion signals. Seeking Alpha guidance supplies another stock-specific opinion. These are evidence for the final prompt, not independent policies whose predictions are averaged.

6. Make the current decision. GPT-4 combines current and retrieved market intelligence, both levels of reflection, account state, trader preference, and tool outputs. It returns structured BUY, HOLD, or SELL plus reasoning.

7. Store the new record. The current evidence, reflections, decision, and later outcome become retrievable history for future dates. GPT-4 and GPT-4V weights do not change.

FinAgent · Key formula or decision rule

The paper writes the final decision compactly as

$$a_t = D_\lambda\Big(\mathrm{LLM}\Big(\phi^D_\lambda(s_t, \mu_t)\Big)\Big), \qquad \mu_t = \mu(s_t, \mathrm{Mem}^\lambda_t, \mathrm{Tool}^\lambda_t).$$

- λ identifies the trading task.
- s_t is today's multimodal market and account state.
- Mem^λ_t is retrieved market and reflection history.

FinAgent · Annual-rate formula and definitions

- $\mathrm{Tool}_t^{\lambda}$ contains the auxiliary guidance and rule signals.
- μ_t is the combined reasoning context constructed by FinAgent's modules.
- ϕ_{λ}^{D} converts that context into the final decision prompt.
- D_{λ} parses the LLM response into action $a_t \in \{\text{buy, hold, sell}\}$.

The paper's annual rate of return is linearly annualized:

$$ARR = \frac{V_T - V_0}{V_0} \frac{252}{T},$$

where V_0 and V_T are initial and final portfolio values and T is the observed number of trading days. It is not geometric compounding.

FinAgent · Why it is different and Experiment

FinAgent goes beyond text-only memory. It gives a vision model two different jobs: explain how older evidence related to a completed price move, and inspect whether recent trading decisions worked. It then adds executable, deterministic tools to the final prompt.

- Full dataset: 1 June 2022 through 1 January 2024, 398 trading days.
- History/setup period: 1 June 2022 through 1 June 2023.
- Test: 1 June 2023 through 1 January 2024.
- Assets: AAPL, AMZN, GOOGL, MSFT, TSLA, and ETHUSD.
- Models: GPT-4 for market-intelligence and final-decision text; GPT-4V for chart-based price and trade reflections. Their weights remain fixed.
- Comparators: buy-and-hold, technical rules, machine/deep-learning models, RL policies, and two earlier language-model systems.

- The technical-strategy hyperparameters are tuned with Optuna on the designated training data.

FinAgent · Reported results

Asset	FinAgent ARR	Buy-and-hold ARR	Sharpe	MDD
AAPL	31.90%	13.00%	1.43	10.40%
AMZN	65.10%	42.33%	1.61	13.20%
GOOGL	56.15%	22.47%	1.78	8.45%
MSFT	44.74%	22.49%	1.79	5.57%
TSLA	92.27%	37.40%	2.01	12.14%
ETHUSD	43.08%	29.26%	1.18	12.72%

FinAgent · Sequential ablation

The sequential ablation is particularly informative:

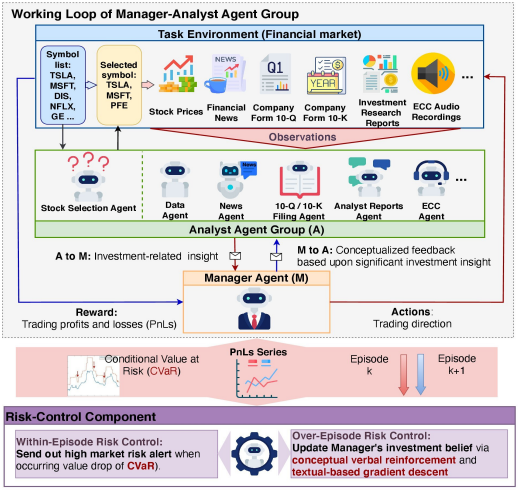
Context added in order	TSLA ARR / Sharpe	ETHUSD ARR / Sharpe
Market evidence only	39.27% / 0.77	25.97% / 0.77
+ price-history reflection	57.16% / 1.02	52.33% / 1.34
+ trade-history reflection	89.25% / 1.46	54.80% / 1.40
+ tools	92.27% / 2.01	43.08% / 1.18

FINCON · Paper metadata and Core idea

FinCon: A Synthesized LLM Multi-Agent System with Conceptual Verbal Reinforcement for Enhanced Financial Decision Making, Yangyang Yu et al., NeurIPS 2024.

FINCON uses fixed GPT-4-Turbo specialists to distill different evidence sources, one manager to choose trading directions, a numerical optimizer to convert directions into weights, and two feedback loops to warn about current tail risk and save natural-language investment beliefs across simulated runs.

FINCON · Manager, analysts, and two feedback loops



FINCON, Figure 2, archived paper p. 4.

FINCON · What goes in

- Yahoo Finance prices and tabular market indicators.
- Reuters/Refinitiv news.
- SEC filings, including 10-Q and 10-K reports.
- Zacks research and stock-selection evidence.
- Earnings-call audio interpreted using Whisper.
- Current portfolio state, procedural memories, previous manager reflections, and saved investment beliefs.

The main setup uses GPT-4-Turbo at temperature 0.3. The source describes seven specialist analyst types, although some filing work is visually grouped in its architecture figure.

FINCON · How it works · 1–6

1. Split the evidence by role. Each specialist processes one modality or task and writes a concise report rather than a trade.

2. Retrieve relevant procedural memory. Each specialist can retrieve role-specific prior events; the reported configuration uses the top five procedural memories per agent.

3. Choose directions. The manager reads the specialist reports, current state, saved beliefs, and any within-run risk warning, then returns buy, short, or hold for each asset. "Sell" in the portfolio task means a negative, short position is allowed.

4. Convert language to weights. An external mean-variance-style optimizer takes the manager's directions, expected returns, and return covariance and calculates numeric portfolio weights subject to direction signs.

5. Observe daily profit and loss. The simulator executes the resulting positions and records daily P&L and portfolio history.

6. Apply within-run risk feedback. FINCON computes the worst-1% P&L tail. If CVaR becomes worse than the previous day or daily P&L is negative, the next manager prompt receives a cautious self-reflection instruction. This loop also remains active in testing.

7. Compare complete build runs. After a simulated build-period trajectory, FINCON compares it with the preceding trajectory, identifies profitable and losing action patterns, and writes conceptual investment lessons. The overlap between action sequences controls the prompt-update scale.

8. Freeze saved beliefs for testing. The between-run verbal belief update stops during the test period; the already learned beliefs and the within-run warning mechanism remain available.

The LLM's model weights stay fixed. "Verbal reinforcement" changes prompts, memories, and saved beliefs rather than neural parameters.

FINCON · Optimizer and direction constraints

The external optimizer solves

$$\max_{\mathbf{w}} \langle \mathbf{w}, \boldsymbol{\mu} \rangle - \langle \mathbf{w}, \boldsymbol{\Sigma} \mathbf{w} \rangle ,$$

with direction constraints for each asset n:

$$w_n \in \begin{cases} [0, 1], & \text{manager says buy,} \\ [-1, 0], & \text{manager says sell/short,} \\ \{0\}, & \text{manager says hold.} \end{cases}$$

- $\mathbf{w}$ is the portfolio-weight vector.
- $\boldsymbol{\mu}$ is the estimated-return vector.
- $\boldsymbol{\Sigma}$ is the estimated return-covariance matrix.

FINCON · Tail-risk decision rule and Why it is different

The source's displayed formulation does not state a separate sum-to-one or cash constraint, so one should not be added when explaining it.

Within a run,

$$\mathrm{CVaR}_{\alpha}(PnL) = \mathbb{E}[PnL \mid PnL \le \mathrm{VaR}_{\alpha}(PnL)], \qquad \alpha = 1\%.$$

VaR is the 1% lower-tail threshold; CVaR is the average P&L among observations at or below that threshold. A more negative CVaR means worse tail outcomes.

FINCON does not let the LLM invent exact portfolio weights. Language roles summarize evidence and choose directions; a numerical solver handles covariance and weights. It also separates immediate risk feedback inside a run from slower natural-language belief updates across complete runs.

FINCON · Experiment

- Build period, called training by the paper: 3 January through 4 October 2022.
- Test: 5 October 2022 through 10 June 2023.
- Single-stock tests: eight stocks; the presentation focuses on TSLA, GOOG, and NIO.
- Portfolio tests: P1 = TSLA/MSFT/PFE and P2 = AMZN/GM/LLY, selected by a stock-selection agent from a 42-stock pool with substantial news coverage.
- Comparators: buy-and-hold, several LLM and RL agents for single stocks; Markowitz mean-variance, an A2C reinforcement-learning baseline, and equal weight for portfolios.

- Runs: five stochastic trajectories. The reported trajectory is the median-cumulative-return run; if median CR and median Sharpe occur in different runs, the CR run is used.

FINCON · Single-stock results

Selected single-stock rows:

Asset	FINCON CR / Sharpe / MDD	Buy-and-hold CR
TSLA	82.87% / 1.97 / 29.73%	6.43%
GOOG	25.08% / 1.05 / 17.53%	22.42%
NIO	17.46% / 0.34 / 40.65%	-77.21%

FINCON · Portfolio results

Portfolio rows:

Portfolio and method	CR	Sharpe	MDD
P1 FINCON	113.84%	3.27	16.16%
P1 Markowitz MV	12.64%	0.61	17.84%
P1 A2C RL baseline	19.46%	0.83	26.92%
P1 equal weight	9.34%	0.49	21.22%
P2 FINCON	32.92%	1.37	21.50%
P2 Markowitz MV	10.29%	0.54	25.10%
P2 A2C RL baseline	11.59%	0.65	15.79%
P2 equal weight	15.06%	0.87	14.66%

FINCON · Risk-ablation results

Risk-component ablations:

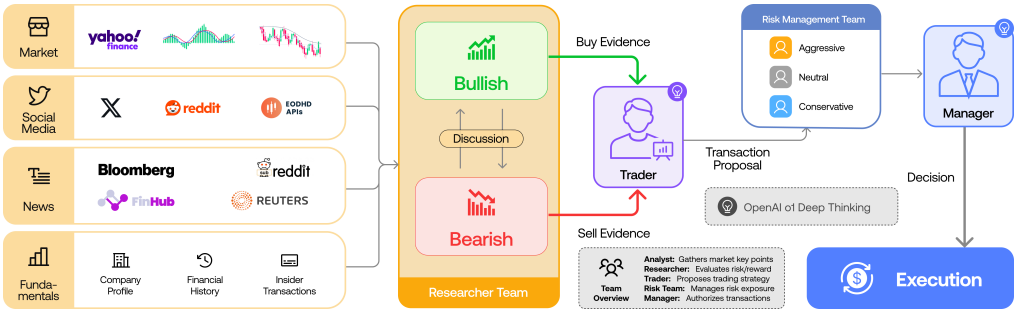
Task	Full FINCON CR	Without within-run CVaR CR	Without saved belief-update CR
GOOG	25.08%	-1.46%	-11.94%
NIO	17.46%	-52.89%	8.20%
P1	113.84%	14.70%	28.43%

TradingAgents · Paper metadata and Core idea

TradingAgents: Multi-Agents LLM Financial Trading Framework, Yijia Xiao, Edward Sun, Di Luo, and Wei Wang, 2024 preprint; presented at a non-archival AAAI 2025 workshop.

TradingAgents turns the decision into a virtual investment-firm workflow: four analysts gather evidence, two researchers debate the investment case, a trader drafts a signal, three risk roles debate that proposal, and a fund manager returns the final buy, sell, or hold action.

TradingAgents · Virtual investment-firm workflow



TradingAgents, Figure 1, archived paper p. 3.

TradingAgents · What goes in

- OHLCV prices.
- Company news from sources named by the paper, including Bloomberg, Yahoo, EODHD, and Finnhub.
- Reddit and X/Twitter posts and auxiliary sentiment scores.
- Insider information, including SEDI records, and company filings.
- Financial statements, earnings reports, company profiles, and financial history.
- A tool library containing 60 technical indicators, including MACD, RSI, and Bollinger Bands.

The paper describes role-specific prompts containing each agent's name, role, goal, constraints, context, and tools. It names GPT-4o-mini and GPT-4o as quick models for retrieval and shallow tasks, and o1-preview as a deeper model for reasoning-heavy work. It does not provide one complete per-node table of exact model, temperature, and canonical prompt settings.

TradingAgents · How it works · 1–3

1. Gather four reports. Fundamental, social-sentiment, news, and technical analysts use ReAct-style reasoning and tools, then save structured reports into shared state.

2. Debate the investment case. A bullish researcher argues for the asset and a bearish researcher argues against it using those reports. They exchange natural-language arguments for n rounds.

3. Choose the research case. A facilitator reviews the debate and saves the prevailing perspective as a structured record. The paper denotes the number of rounds by n but does not state one fixed experimental value.

TradingAgents · How it works · 4–7

4. Draft a trade. The trader reads the reports and research record and proposes a buy, sell, or hold signal with supporting reasoning.

5. Debate risk. Risk-seeking, neutral, and risk-conservative agents review the trader's proposal for n rounds and suggest adjustments.

6. Approve or revise. The fund manager reviews the risk discussion and writes the final action and report back into shared state.

7. Execute in the backtest. The chosen action is applied in the historical simulator, and the next date begins with updated portfolio state.

The workflow is sequential. The roles do not independently place trades at the same time.

TradingAgents · Why it is different and Experiment

TradingAgents makes two explicit disagreement stages part of one decision. Reports preserve evidence between roles, while brief debates force opposing investment and risk perspectives to be written down before the final action.

- Test: 1 January through 29 March 2024.
- Main reported assets: AAPL, GOOGL, and AMZN. The paper describes a broader technology-stock dataset, but its main numeric table contains these three rows.
- Output: daily buy, sell, or hold actions in a historical simulation.
- Baselines: buy-and-hold plus four technical-rule strategies—MACD, KDJ+RSI, zero-mean reversion, and simple moving average.
- Cost: approximately 11 LLM calls and more than 20 tool calls per prediction; the paper reports this computational count but not an API-dollar cost.

- No LLM-weight training or fine-tuning phase is reported.

TradingAgents · Reported results

Asset	TradingAgents cumulative return	Sharpe	MDD	Buy-and-hold cumulative return
AAPL	26.62%	8.21	0.91%	-5.23%
GOOGL	24.36%	6.39	1.69%	7.78%
AMZN	23.21%	5.60	2.11%	17.10%

MDD is a positive loss magnitude. The table also reports annualized-return values of 30.50%, 27.58%, and 24.90%, respectively, but those values do not reconcile with the formula stated in the paper and the three-month cumulative returns.

PART 03

Agentic Quantitative Research

From formula search to complete quantitative research and development

Quantitative Research Systems in This Section

System	Primary output	What this paper adds to the sequence
AlphaGen	weighted factor library	an RL policy writes formulas; the library's IC supplies feedback
AlphaAgent	hypothesis + approved factor library	turn an economic idea into a formula constrained for simplicity, novelty, and consistency
R&D-Agent-Quant	factors + predictive model	choose the next research direction, write code, test it, and retain results
AlphaEvo	small low-correlation factor subset	combine language-model proposals, evolutionary search, and temporal validation

Reading rule. A **factor** is a formula that turns recent market data into one score per stock. **IC** is the correlation between that score and later return. Some papers print IC as a decimal, others as percentage points; markets, horizons, and tests also differ, so this is not a cross-paper leaderboard. AlphaGen is a non-LLM search precursor; the later systems add language-model research roles.

Sources: the four primary papers discussed in this section. The following slide introduces FunSearch and EoH as program-search precedents.

Two Precedents for LLM-Guided Program Search

Same loop, different proposal rule. x: fixed problem; c: executable code; $s = E(c)$: evaluator score; $\mathcal{A}$: verified archive.

FunSearch

Romera-Paredes et al. · Nature 2024

1. Sample strong, diverse code $C_k \subset \mathcal{A}_k$.
2. Propose $c_{k+1} \sim \pi_\theta^{\text{code}}(x, C_k)$.
3. Execute $s_{k+1} = E(c_{k+1})$.
4. Keep verified code in $\mathcal{A}_{k+1}$.

Difference: the LLM rewrites code; exact execution decides what survives.

Reported example: a 512-point cap set in eight dimensions.

Evolution of Heuristics (EoH)

Liu et al. · ICML 2024

1. Read tested archive $\mathcal{A}_k$.
2. $\pi_\theta^{\text{heuristic}}(x, \mathcal{A}_k)$ proposes idea h_{k+1} + code c_{k+1}.
3. Execute $s_{k+1} = E(c_{k+1})$.
4. Keep verified (h, c, s) in $\mathcal{A}_{k+1}$.

Difference: the LLM writes both a verbal heuristic and its code.

Reported example: 2.13% bin-packing gap versus 6.75% for FunSearch; lower gap is better.

Sources: FunSearch (Nature 2024) and EoH (ICML 2024). They are related search precedents, not a direct chronological lineage.

AlphaGen · Paper overview and inputs

Paper: *Generating Synergistic Formulaic Alpha Collections via Reinforcement Learning* — Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He; ACM KDD 2023.

Core idea

AlphaGen is a reinforcement-learning agent that writes stock-prediction formulas one token at a time and rewards a new formula according to how much it improves the complete weighted factor library—not according to how strong it is alone.

- Daily open, high, low, and close prices, volume, and volume-weighted average price (VWAP).
- A legal vocabulary of features, constants, lookback periods, and mathematical operators.
- The formulas and weights already retained in the current factor library.

The prediction target is each stock's return over the following 20 trading days. AlphaGen does not receive news or a text prompt, and it does not use an LLM.

1. Write a formula. A Proximal Policy Optimization (PPO) policy chooses one formula token at a time. Its actions are formula tokens, not stock trades. It uses reverse Polish notation, which puts inputs before their operator. For example,

$$[\text{close}, 5, \text{TS-Mean}]$$

means "calculate the five-day trailing mean of the closing price." Invalid next tokens are masked, and a formula is limited to 20 tokens.

2. Execute the formula. The completed expression is run for every stock and date. It produces one score for stock i on date t, written $f_j(X_{i,t})$, where $X_{i,t}$ is that stock's available price-volume history and f_j is factor j.

3. Add it to the factor library. AlphaGen temporarily adds the new factor and refits all linear combination weights:

$$c_{i,t} = \sum_{j=1}^{K} w_j f_j(X_{i,t}).$$

Here K is the number of retained factors, w_j is factor j's fitted weight, and $c_{i,t}$ is the library's combined score for stock i. If the library exceeds its capacity, the factor with the smallest absolute weight is removed.

On each date, the evaluator correlates the combined scores across stocks with their later 20-day returns, then averages across dates:

$$IC = \frac{1}{T}\sum_{t=1}^{T} \mathrm{Corr}_i(c_{i,t}, r_{i,t\to t+20})\,.$$

T is the number of evaluation dates and $r_{i,t\to t+20}$ is stock i's subsequent 20-day return. RankIC performs the same calculation after converting scores and returns to ranks.

AlphaGen · Feedback and decision rule

5. Use the library score as reinforcement-learning feedback. A formula receives a high reward when the updated library predicts later returns well. PPO consequently becomes more likely to generate token sequences that add useful information to the existing factors.

Key formula or decision rule

The stock score is the weighted factor library $c_{i,t} = \sum_j w_j f_j(X_{i,t})$. The PPO reward is the rebuilt library's IC with subsequent 20-day return. Therefore, the accept-and-learn signal concerns the new factor's contribution to the whole library, not its standalone IC.

AlphaGen · Experiment

- **Market:** Chinese A-shares from BaoStock.
- **Universes:** CSI 300 large-cap stocks and CSI 500 smaller stocks outside the CSI 300.
- **Training:** 2009–2018.
- **Validation:** 2019.
- **Test:** 2020–2021.
- **Target:** subsequent 20-day stock return.
- **Repeated runs:** stochastic configurations are reported over ten random seeds.
- **Separate trading simulation:** rank stocks by the combined score, hold the top 50, and replace at most five positions per day.

AlphaGen · Reported results

Method	CSI 300 IC	CSI 300 RankIC	CSI 500 IC	CSI 500 RankIC
XGBoost	0.0404	0.0576	0.0353	0.0639
Genetic-programming top pool	0.0078	0.0157	0.0200	0.0504
Individually filtered PPO pool	−0.0044	0.0101	0.0042	0.0506
AlphaGen	**0.0725**	**0.0806**	**0.0438**	**0.0727**

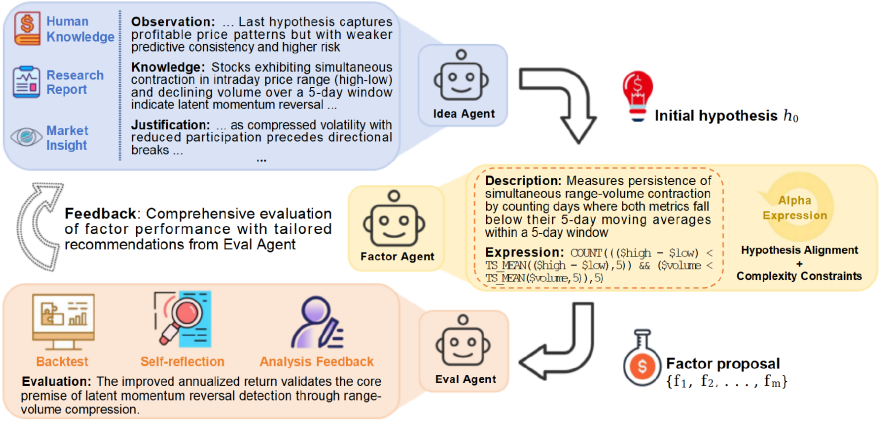
AlphaAgent · Paper overview

Paper: *AlphaAgent: LLM-Driven Alpha Mining with Regularized Exploration to Counteract Alpha Decay* — Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin; ACM KDD 2025.

Core idea

AlphaAgent uses three LLM roles to turn a written market idea into a simple, novel, executable factor, tests it, remembers the result, and then feeds approved factors into LightGBM for a cost-adjusted portfolio backtest.

AlphaAgent · Paper workflow



AlphaAgent, Figure 1.

AlphaAgent · What goes in

- A user-specified research direction plus financial knowledge, market observations, and earlier research reports.
- Previous successful and failed factor attempts retained in the agent's knowledge base.
- An existing factor library, including widely known factors against which novelty can be checked.
- Daily open, high, low, close, and volume data and a fixed library of legal factor operators.

The main agents use GPT-3.5-Turbo. The LLM proposes research objects; it does not directly send buy or sell orders.

AlphaAgent · Experiment

- **Markets and sources:** BaoStock CSI 500 and Yahoo Finance S&P 500 OHLCV data.
- **Training:** 2015–2019.
- **Validation:** 2020.
- **Paper-labelled test:** 2021–2024.
- **Adaptive search:** 20 independent trials, each containing five sequential factor-evolution rounds; each round refits LightGBM.
- **Portfolio:** 50 stocks, with five replaced per rebalance.
- **Costs:** CSI 500 assumes 5 basis points to buy and 15 to sell; S&P 500 assumes 5 basis points to sell. One basis point is 0.01%.

AlphaAgent · Reported results

Market	IC	ICIR	Annualized excess return	Information ratio	Maximum drawdown
CSI 500	**0.0212**	**0.1938**	**11.00%**	**1.488**	−9.36%
S&P 500	**0.0056**	**0.0552**	**8.74%**	**1.055**	−9.10%

In the paper's constraint ablation, a "hit" is a generated factor whose annual-return result exceeds 4.0% on CSI 500 or 1.8% on S&P 500. The reported hit rate falls from 0.29 to 0.16 when the factor-modeling constraints are removed.

AlphaEvo · Paper overview

Paper: *AlphaEvo: LLM-Seeded Evolutionary Discovery of Robust Quantitative Trading Signals* — Temiko Machavariani, Kyriakos Tsourekas, Matvei Rotte, Luka Tsulaia, Iskhak Tazhibaev, Munachiso Samuel Nwadike, Salem Lahlou, Prashant Kumar Jamwal, Kentaro Inui, Martin Takáč, and Zangir Iklassov; local 2026 preprint manuscript.

Core idea

AlphaEvo lets an LLM propose market-regime-aware starting formulas, lets genetic programming search many executable variants, and then gives an LLM the formulas' early/middle/late scores so it can make targeted structural revisions before removing near-duplicates.

AlphaEvo · What goes in

- Daily Yahoo Finance OHLCV data for one stock-quarter discovery task.
- A regime summary derived from RSI, the MA20/MA50 gap, realized volatility, a volume ratio, and the dominant candlestick pattern.
- An optional written trading idea.
- A constrained formula language with 27 terminals and 46 operators. A terminal is an input such as close or volume; an operator is a calculation such as rolling mean, subtraction, or normalization.
- Retrieved factor-family examples and, optionally, useful formulas from earlier sessions on the same stock.

The reported 120-task experiment starts every task with empty session memory to keep tasks statistically separate. The default LLM in the manuscript is Groq-hosted Llama 3.1 8B.

AlphaEvo · How it works · 1–3

1. Summarize the market regime. The system converts recent data into a short description of trend, volatility, volume, and candlestick behavior.

2. Narrow the formula search space. A three-level retrieval process chooses one of four broad alpha categories, one of 11 subcategories, and six to eight relevant terminals from the full set of 27. The prompt also receives two or three example formulas and the legal operator definitions.

3. Ask the LLM for seeds. The seed prompt returns up to ten executable formulas, each with a one-sentence financial rationale. Invalid, constant, infinite, and all-missing formulas are rejected; deterministic classical anchor formulas fill any empty places.

4. Run ordinary genetic programming. A population of 24 formulas evolves for 20 generations, with the eight best retained as elites. New candidates are made by crossover (60%), mutation (25%), or wrapping an existing expression in a smoothing/normalization operator (15%). A global archive keeps earlier strong formulas from being lost.

5. Score each executable factor. The basic predictive metric is time-series IC:

$$IC(f) = \mathrm{Corr}_t(f(H_t), r_{t+1}),$$

where H_t is one stock's available history, $f(H_t)$ is its signal on date t, and r_{t+1} is its next-day return. The search uses a composite score:

Experiment 1: 120 stock-quarter discovery tasks

- **Stocks:** AAPL, MSFT, NVDA, META, GOOGL, AMZN, TSLA, JPM, JNJ, XOM, WMT, and CAT.
- **Period:** ten quarters from 2022 Q1 through 2024 Q2.
- **Tasks:** 12 stocks $\times$ 10 quarters $=$ 120 separately run search tasks.
- **Within-task split:** first 70% precedes the repeatedly inspected final 30%.
- **Primary metric:** TSIC, which follows one stock across dates.

Experiment 2: one formula across 503 stocks

The manuscript separately takes each method's **best single formula**—not AlphaEvo's final eight-factor bundle—and applies it to 503 stocks that were S&P 500 constituents in 2024 over January 2010–December 2024. Each day it ranks the stocks, buys the top 20%, and shorts the bottom 20%.

AlphaEvo · Reported results · Experiment 1

Method	Short-window Sharpe	MDD	Total return	Turnover
Random genetic programming	3.68	4.2%	21.5%	0.164
LLM-only	1.63	8.1%	9.7%	0.240
GP + anchors	4.17	3.9%	24.9%	0.238
AlphaEvo	**4.61**	**3.7%**	**27.9%**	**0.240**

PART 04

The Industry Today

Measured adoption, usefulness, and investment-firm workflows

AI Adoption Is Not Trading Authority

One Mercer survey asked 131 asset managers two different questions.

55%

integrated **AI of any kind** into at least one investment process

This can mean research, forecasting, data processing, risk analysis, or decision support. It does not imply an autonomous agent.

broad use ⇏ authority

~5%

gave AI some **autonomous or semi-autonomous authority** for an investment recommendation or trade

Mercer combines recommendations with trades and combines the two authority levels, so this is not a count of fully autonomous portfolios.

The 5% statistic asks a much narrower question than the 55% statistic. AI is widely present in the investment process; authority over a capital-related decision is still rare.

Source: Mercer AI in Asset Management Survey, 131 global asset managers, February–March 2026. Voluntary, self-reported responses. Another 27% were piloting AI and 18% reported no integration.

What Asset Managers Report AI Has Improved

Share of Mercer respondents reporting a measurable benefit; respondents could select more than one

69%	55%	8%	8%
greater operational efficiency	faster or higher-quality investment insight	improved investment returns	lower risk or portfolio volatility

What the numbers support

The strongest reported benefits are faster work and faster analysis. Only 8% of managers said AI produced a measurable return improvement, and Mercer did not report the size of that improvement.

What they do not prove

These percentages count managers reporting each benefit. They are not an independent experiment, do not isolate agentic systems, and do not establish that AI caused the outcome.

Source: Mercer, 131 asset managers globally, surveyed February–March 2026. Participation was voluntary and results are self-reported.

Investment Firms: Research Agents, Human Decisions

Balyasny Asset Management

Global multi-strategy investment firm.

- About **95%** of investment teams use its AI research platform.
- A macroeconomic scenario-analysis workflow reportedly fell from about **2 days to 30 minutes**.
- A merger-arbitrage agent estimates whether an announced acquisition will close; those probabilities inform human decisions.

Company/vendor-reported workflow result; no separate agent-managed portfolio return is disclosed.

Man Group — AlphaGPT

London-listed manager: $227.6B AUM at year-end 2025; $140B on its quantitative/rule-based systematic platform. AlphaGPT is an internal research workflow, not a fund.

1. an idea agent proposes a testable concept;
2. an implementer writes production-style Python;
3. an evaluator applies statistical, risk, and economic checks.

Man reports dozens of viable concepts in minutes and coding compressed from hours/days to minutes. "Viable" means ready for testing, not accepted or profitable; no standalone AlphaGPT return is published.

Sources: OpenAI/Balyasny, 6 March 2026; Man Group article and 2025 Annual Report. Workflow gains are company/vendor reported.

PART 05

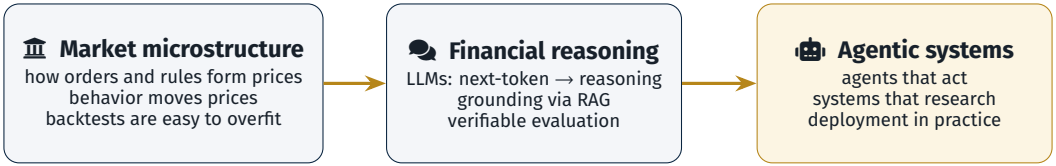
Closing the Tutorial

Three layers, one thesis

Financial reasoning → Agents Trade Research Industry **Close**

The Arc We Traveled Today

Across the full tutorial: market behavior → model reasoning → agent action.



Markets generate noisy data and text; models interpret them; agents turn analyses into tested actions. Before deployment, every layer must pass out-of-sample, cost, risk, and governance checks. Here RAG means **retrieval-augmented generation**: retrieve evidence before answering.

Resources & Thanks

Take it home

- AlphaEvo — code, prompts, alpha library: github.com/Zangir/AlphaEvo
- **Coming soon** (not yet published): Colab notebooks for financial-sentiment model fine-tuning · agent conflict resolution · repair a flawed backtest
- Tutorial site: zangir.github.io/genai-finance-kdd2026

Reading list (start here)

- FunSearch — evaluator-guided program discovery
- TradingAgents — multi-role investment and risk debate
- AlphaAgent — LLM factor mining
- Deflated Sharpe — backtest selection correction (Bailey & López de Prado)

With thanks to Martin Takáč, Jean-Jacques Duhot, and the AlphaEvo team.

Questions — then the joint ethics discussion.

APPENDIX

Backup Slides

Backup: Jargon Decoder

Term	Plain meaning	Technical cue
Agent	policy in an observation–action–feedback loop; tools and memory are optional	$a_t \sim \pi(a \mid o_t, c_t)$
Alpha / signal	rule with predictive information about return	$f_t \mapsto r_{t+1}$
IC / Rank IC	Pearson / Spearman correlation between signal and later return	$\rho_P(f_t, r_{t+1})$ / $\rho_S(f_t, r_{t+1})$
Sharpe ratio	mean daily excess return per unit of volatility	$r =$ daily excess return; $\sqrt{252}\,\bar{r}/\sigma_r$
Genetic prog.	evolve symbolic formulas by variation and selection	population → mutate/cross → select
Regime	a market state with different dynamics	latent state z_t
Alpha decay	predictive strength weakens over time	$IC_{t+h} < IC_t$
Reflexivity	in live markets, actions can alter the process being predicted	a_t can change s_{t+1}
RAG	retrieval-augmented generation: retrieve evidence before answering	query → evidence → cited output

Backup: Industry Sources

- **Adoption and benefits**: Mercer, “How Artificial Intelligence Is Shaping Asset Management,” May 2026 (131 managers surveyed in Feb–Mar 2026).
- **Agentic-AI adoption**: Cambridge CCAF, “The 2026 Global AI in Financial Services Report: Adoption, Impact and Risks” (352 firms surveyed in Oct 2025–Jan 2026).
- **Regulatory market view**: Bank of England, “Financial Stability Report,” July 2026.
- **Bridgewater AIA**: Bridgewater launch note, 1 Jul 2024; Bridgewater AIA Labs; Bloomberg, 1 Jul 2024 and 2 Jan 2026; Reuters, 1 Jul 2026.
- **Balyasny**: OpenAI/Balyasny case study, 6 Mar 2026.
- **Man Group AlphaGPT**: Man Group, 13 Nov 2025; Bloomberg, 10 Jul 2025.

Company adoption, time-saving, and productivity figures are self-reported; private-fund size and return figures are company/press reported.

Backup: Primary Sources for Trading and Analysis Systems

- **FinRL-Meta**: Liu et al., NeurIPS 2022 Datasets & Benchmarks, official conference proceedings.
- **FinAgent**: Zhang et al., KDD 2024, DOI:10.1145/3637528.3671801, arXiv:2402.18485.
- **FINCON**: Yu et al., NeurIPS 2024, arXiv:2407.06567v3; official repository The-FinAI/FinCon.
- **TradingAgents**: Xiao et al., arXiv:2412.20138; AAAI 2025 Multi-Agent AI in the Real World workshop.

Result labels in the main deck reproduce the authors' tables; audit notes distinguish paper claims, released-code details, and our methodological inferences.

Backup: Quant Research Sources

- **AlphaGen**: Yu et al., KDD 2023, DOI:10.1145/3580305.3599831, arXiv:2306.12964.
- **AlphaAgent**: Tang et al., KDD 2025, DOI:10.1145/3711896.3736838, arXiv:2502.16789.
- **R&D-Agent-Quant**: Li et al., NeurIPS 2025 Datasets & Benchmarks Track; arXiv:2505.15155; official Microsoft RD-Agent repository.
- **AlphaEvo**: Machavariani et al., 2026 preprint manuscript.

Screened but outside the evaluated-trading-agent set:

- FinGPT is a financial language-model/data pipeline; the original paper contains no trading-agent backtest.
- TradingGPT proposes personas, memory, and debate, but states that experiments and ablations remain future work.
- StockAgent evaluates emergent behavior in a synthetic two-stock market; its trading volume is not evidence of investment alpha.
- QuantAgent is a useful writer–judge precursor, but has no untouched final test and reports mostly figure-only results.