iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84129

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

A Cascaded Machine-Learning Pre-Filter and Adaptive LLM Re-Ranking Architecture for Financial News Importance Scoring

Viswanath Parashuram Yadavalli
School of Computer Science and Engineering, Vellore Institute of Technology (VIT), India

Abstract: ***Financial news is published faster than any person can read it. This paper presents IntrAIdify, a live financial news ranking system built as a two-stage cascade: a machine learning ensemble (Linear Regression, Random Forest, XGBoost, LightGBM) first filters incoming headlines down to the top 10, and a large language model (LLM) then re-ranks those 10 using a softmax mechanism that adaptively weights the ML and LLM scores per headline. Three smaller signals - a keyword lexicon, a topic-relevance score, and an actionability score - add further weight to the final ranking. We evaluate the system on a full 398-day held-out test set in two ways: how well it ranks the full list of headlines, and how well the LLM re-ranks the top 10 the ML ensemble already chose. On the full-list test, the plain ML ensemble significantly outperforms the full pipeline. On the top-10 test, the LLM re-ranking shows no significant improvement over the ML ensemble's own ordering. We report both results plainly, discuss what they mean for the architecture, and argue that measuring a reranker within its actual candidate set, not just across the full list, is the correct way to evaluate any cascaded ranking system.***

Keywords: Financial news ranking, cascaded retrieval and re-ranking, ensemble machine learning, large language model re-ranking, information retrieval, learning to rank, financial natural language processing.

I. INTRODUCTION

Financial news volume has grown to a point where no individual trader, analyst, or retail investor can read every relevant headline as it is published. A well-established solution to this class of problem in information retrieval is the retrieve-then-rerank cascade: a cheap first-stage model narrows a large candidate pool to a small high-recall subset, and a more expensive, higher-precision second-stage model re-orders that subset [19], [20]. This paper describes and evaluates IntrAIdify, a live, continuously operating financial news ranking system built explicitly on this pattern: a four-model machine learning ensemble serves as the cheap first-stage filter, narrowing each incoming batch of headlines to a top-10 candidate set, and a large language model (LLM) serves as the second-stage reranker, refining the ordering of that candidate set via an adaptive, temperature-scaled softmax blend with the first-stage score. Three auxiliary signals - a batch-adaptive keyword lexicon, a k-nearest-neighbor topic-relevance score, and a lexical-semantic actionability score - contribute further weighted terms to the published ranking. The central design claim of a retrieve-then-rerank cascade is not that the second-stage model should outperform the first-stage model across the entire original candidate pool - the second-stage model never even observes most of that pool. The claim is narrower and more specific: given the same candidates the first stage already selected, does the second stage improve their ordering? Evaluating a cascade by its full-list performance, rather than by its performance within the candidate set it actually acts on, conflates the first stage's recall with the second stage's precision and can misattribute credit or blame between the two stages. This paper evaluates IntrAIdify along both axes explicitly: standard full-list ranking metrics for context and comparability with prior work, and a dedicated cascade-specific metric that isolates the LLM reranker's actual contribution to the candidate set the ML ensemble selected for it. Much of the published literature on financial news and market prediction targets a different task: classifying sentiment or predicting price direction from text [1]-[4]. This work instead targets headline importance ranking, framed as a learning-to-rank problem [21] within a cascaded retrieve-rerank architecture, a problem formulation and evaluation methodology largely absent from the financial NLP literature despite being standard practice in general-purpose information retrieval [19], [20].

The contributions of this paper are as follows:

1) The design and live deployment of a cascaded financial news ranking architecture: a four-model ML ensemble as first-stage filter, an LLM as second-stage reranker, combined via a temperature-scaled adaptive softmax mechanism that dynamically reweights the two stages per headline rather than applying a fixed blending ratio, augmented with three auxiliary lexical and semantic scoring signals.

©IJRASET: All Rights are Reserved

2) An evaluation methodology appropriate to cascaded rerankers in general, not specific to this system: a held-out, time-based evaluation protocol including random-ranking and majority-class baselines, paired with a cascade-specific metric that isolates a second-stage reranker's contribution to the exact candidate set it operates on, rather than to the full unfiltered pool it never entirely observes. We argue this dual-axis approach is necessary for any retrieve-then-rerank system and demonstrate why with the worked example in Section IV.
3) A full evaluation of IntrAIdify on all 398 held-out test days, checking both the full-list ranking quality and the cascade-specific top-10 re-ranking quality, with paired significance testing for every comparison. Neither check shows a benefit from the LLM re-ranking stage or the auxiliary scoring signals over the plain ML ensemble, and we report this plainly.
4) A component-wise ablation study, with paired statistical testing, isolating the measured contribution of each auxiliary signal source to full-list ranking quality, complementing the cascade-specific result with system-level context.
5) A fully transparent discussion of null and negative findings, including cases where added system complexity did not improve, and in some configurations reduced, ranking quality relative to the base ensemble, presented as an evidentiary finding rather than omitted.
6) A concrete, practitioner-facing discussion of deployment characteristics, implementation quirks discovered through this evaluation, and applications of the resulting system to real-world financial information triage, in Sections V and VI.

II. RELATED WORK

Sentiment- and news-driven market prediction has been studied extensively using both classical machine learning and deep learning approaches. Verma et al. [1] compared Naive Bayes, Random Forest, and Support Vector Machine classifiers against an LSTM model across three text embedding techniques (TF-IDF, FastText, and Word2Vec) on a corpus of 30,001 social media posts, finding that LSTM combined with Word2Vec embeddings achieved the strongest performance (92.65% accuracy, F1 of 0.9437), while Random Forest was the strongest classical model. This result establishes an upper bound for sentiment classification performance, though it addresses a categorical sentiment task rather than a ranking task.

Khonde et al. [2] built a combined sentiment-analysis and stock-data prediction system using financial news headlines, reporting approximately 83% forecasting accuracy when sentiment and predictive models were combined. Their system, like ours, integrates multiple signal sources into a single forecast, though their target is price direction rather than headline importance.

Deveikyte et al. [3] examined the relationship between sentiment extracted from financial news and Twitter, and FTSE100 returns and volatility, using topic modeling based on Latent Dirichlet Allocation. They found evidence that sentiment could serve as a signal for next-day returns, but found no significant relationship between sentiment and volatility, illustrating that the strength of a news-derived signal depends heavily on which downstream target is being predicted.

Davidovic and McCleary [4] conducted a large-scale investigation using approximately 1.86 million news headlines and multiple sentiment extraction methods (TextBlob, VADER, and FinBERT), combined with a stacking ensemble of XGBoost, Random Forest, and Gradient Boosting base learners with a logistic regression meta-learner. They found that forward-looking implied sentiment (VIX-derived) explained a substantial share of return variation, while explicit textual sentiment scores had comparatively weak and inconsistent predictive power, and that ensemble methods outperformed individual base learners.

Separately from financial NLP, the information retrieval literature has established retrieve-then-rerank cascades as a standard architecture for balancing computational cost against ranking quality: a cheap first-stage retriever selects a manageable candidate pool, and a more expensive second-stage model re-ranks that pool. Nogueira and Cho [19] demonstrated that a transformer-based second-stage reranker, applied only to a first-stage candidate set, substantially improved passage ranking quality over the first-stage model alone. More recently, Qin et al. [20] showed that large language models, prompted appropriately, are effective rerankers in exactly this second-stage role. IntrAIdify's architecture instantiates this same cascade pattern - an ML ensemble as first-stage filter, an LLM as second-stage reranker - applied to financial headline importance rather than passage or document retrieval, and this paper adopts the corresponding cascade-appropriate evaluation methodology (Section IV-D) rather than treating the LLM's ordering of a candidate set it never fully sees as directly comparable to the first-stage model's ordering of the complete pool.

The present work differs from the financial-NLP literature above in task framing: rather than predicting sentiment class or price direction directly, we evaluate the quality of a news importance ranking against a ground-truth importance ordering, using rank-correlation and ranking metrics (Spearman's rho, NDCG@10) as the primary evaluation criteria, with a supplementary directional signal check included for continuity with the classification-oriented literature above, and, distinctly from both bodies of literature, a cascade-specific reranking-quality metric adapted from the information retrieval tradition [19], [20].

©IJRASET: All Rights are Reserved

III.SYSTEM ARCHITECTURE AND METHODOLOGY

A. Overview

IntrAIdify operates as a continuously running FastAPI service with three concurrent processes: a background scraper that polls RSS feeds and re-ranks incoming news every 300 seconds (retrying after 10 seconds on failure), a file-change watcher that polls the persisted ranking output every 5 seconds and triggers email alerts to subscribed users when the top-5 ranked headlines change, and the FastAPI request handlers themselves, which serve the current ranking and manage user sign-up. The ranking pipeline at the core of the system, described in Sections III-C through III-J below, executes seven sequential stages for each batch of incoming headlines: (1) text construction by concatenating title and summary; (2) parallel TF-IDF vectorization across four independently fit vocabularies; (3) scoring by the four-model ML ensemble followed by batch-wise min-max normalization and combination into a single ensemble score; (4) sorting all headlines by ensemble score and selecting the top-10 for LLM evaluation; (5) computation of three auxiliary scores (keyword salience, topic relevance, actionability) for every headline in the batch; (6) adaptive softmax-weighted combination of the ML ensemble score with the LLM signal, added to the three auxiliary scores under fixed weights, to produce a raw score per headline; and (7) a final batch-wise min-max normalization of all raw scores to produce the published ranking.

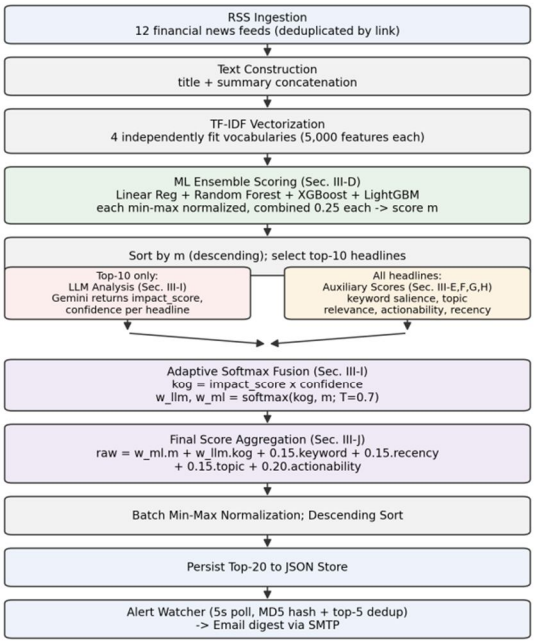


Fig. 1. End-to-end ranking pipeline architecture, from RSS ingestion through email alerting.

B. Data Sources

Two labeled datasets were used for offline model training and evaluation. The first is a public dataset of daily top-25 news headlines drawn from the Reddit community r/worldnews, ranked by community upvote count, paired with a binary label indicating whether the Dow Jones Industrial Average (DJIA) closed higher or lower the following day [5]. Headline rank position within a day (1 through 25) is used as a proxy for perceived public importance. The second is a labeled financial news sentiment dataset covering State Bank of India (SBI) news articles with continuous sentiment intensity scores [6], used to train the sentiment-intensity component of the ensemble. In production, the system additionally ingests live financial news from twelve RSS feeds spanning Dow Jones, Yahoo Finance, CNBC, Investing.com (three category feeds), Reuters, the Financial Times, Bloomberg, Business Insider, and the New York Times Business desk, polling up to ten entries per feed per cycle with link-based deduplication; this live stream is not used for the offline evaluation reported here due to the absence of ground-truth importance labels for real-time headlines.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

C. Text Representation

Each headline is the title and summary fields joined together. Four TF-IDF vectorizers [7], implemented with scikit-learn [23], are fit separately, one per downstream model, each with 5,000 features and English stop-words removed. These four vectorizers were kept separate from the original model development process instead of being merged into one shared vocabulary. This means four separate vectorization passes run per ranking batch, which costs more compute than needed; we flag this as a design inefficiency in Section V.

D. Machine Learning Ensemble

Four models score each headline independently and are combined following standard ensemble learning practice [22]: combining several simpler models usually gives more accurate and more stable predictions than any single model alone. A Linear Regression model and a Random Forest regressor [8] both predict the headline's rank position within its batch, trained on the Reddit upvote-derived rank labels. An XGBoost classifier [9] predicts a three-class discretization of rank position (top-5, rank 6-15, and rank 16-25); in production the predicted probability of the middle class is used as its contribution to the ensemble (Section V discusses this design choice). A LightGBM regressor [10], trained on the separate SBI sentiment dataset, predicts signed sentiment intensity; its output is passed through an absolute-value transform so that strongly negative and strongly positive sentiment are both treated as indicators of market-moving importance rather than only positive sentiment being rewarded. Each of the four raw model outputs is independently min-max normalized within the current ranking batch following the score-normalization approach of Jain et al. [14], mapping each model's output to the [0,1] interval via (x - min)/(max - min) so that heterogeneous score scales can be combined; the four normalized scores are then combined under equal (0.25) weighting to produce a single ensemble score m for each headline.

E. Keyword Salience Scoring

A hand-built list of 47 market-relevant words and phrases (e.g. "crash", "acquisition", "rate hike", "cyberattack") each get a base weight from 2 to 6, based on a rough judgment of how much market impact that term usually signals. These weights are not fixed. Instead, the system computes a trend multiplier for each keyword k, using the current batch of headlines: trend(k) = 1 + ln(1+freq(k)) / ln(1+total). Here freq(k) is how many headlines in the batch contain k, and total is the sum of all keyword counts in the batch. This gives extra weight to terms that are unusually common in today's news, compared to their normal rate. For a given headline, we sum base_weight(k) x trend(k) over every matched keyword, cap the total at 10, divide by 10, and take the square root. This gives a bounded score between 0 and 1.

F. Topic Relevance

This score measures how closely a headline matches the rest of the day's news, using cosine similarity between TF-IDF vectors, following the vector-space model of Salton et al. [17]. For each headline, we find the five most similar other headlines in the batch (a k-nearest-neighbor approach [13] with k=5) and average their similarity scores. This average is then normalized across the batch to get the final topic-relevance score. The idea: a headline covered from multiple angles by different sources is more likely to matter, while an isolated outlier headline is less likely to.

G. Actionability Scoring

An actionability score is intended to favor headlines describing concrete, actionable corporate events (e.g. earnings, mergers, guidance changes) over general macroeconomic commentary, and to favor forward-looking announcements over already-realized price moves. Four hand-built term lists are used: an IMPACT list of 28 corporate-event terms, a MACRO list of 21 macroeconomic terms, an EARLY list of 22 forward-looking terms (e.g. "plans", "expected", "upcoming"), and a LATE list of 22 realized-outcome terms (e.g. "surged", "plunged", "closed"). For each list, we sum ln(1+tfidf) over matched words to get a lexical score for the headline. We also write three small sets of example phrases (actionable, early-stage, and late-stage event language) and vectorize them. Comparing the headline's cosine similarity to each set gives three scaling factors. The final actionability score combines these as: (action_factor x (IMPACT_score - MACRO_score)) + (early_factor x EARLY_score - late_factor x LATE_score). This rewards headlines that use impact-oriented language and sound forward-looking and actionable, rather than macro commentary or old news.

H. Recency Weighting

This term follows the standard exponential recency-decay approach used in ranking and recommendation systems [18]: exp(-decay x hours_since_published), with decay = ln(2) / 48. That means a headline loses half its recency weight every two days.

As Section III-K explains, we could not test this term on our historical dataset, since it has no per-headline timestamp separate from the calendar date.

I. Adaptive LLM Re-Ranking

The ten highest-scoring headlines by ML score m go to a large language model, gemini-3.1-flash-lite [16], which returns an impact_score and a confidence value, both between 0 and 1, for each headline. We multiply these to get kog = impact_score x confidence. Instead of blending m and kog with a fixed ratio, we use an adaptive softmax function [15]: softmax assigns weight w_llm = exp(kog/T) / (exp(kog/T) + exp(m/T)) to the LLM term, and the rest to the ML term, using temperature T=0.7. This way, whichever signal is stronger for a given headline gets more weight, instead of always trusting ML and LLM equally. Headlines outside the top 10 never reach the LLM, so their impact_score and confidence both default to 0.5 (kog=0.25). Section V points out that softmax still treats this default as if it were a real score, not a placeholder.

J. Final Score Aggregation

Each headline's raw score adds the softmax-weighted ML/LLM blend to the three smaller scores: raw_score = (w_ml x m) + (w_llm x kog) + (0.15 x keyword_score) + (0.15 x recency) + (0.15 x topic_relevance) + (0.20 x actionability_score). We then normalize all raw scores in the batch again [14] to get the final published score, and sort headlines from highest to lowest for display and alerts.

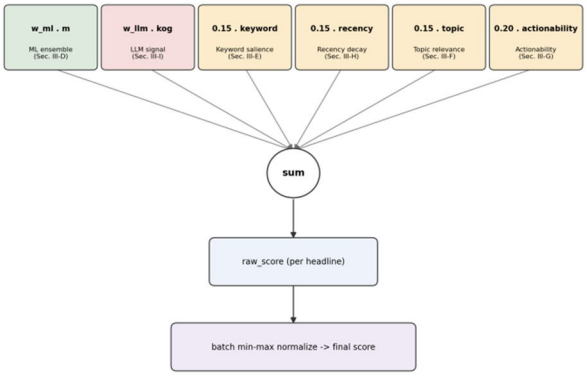


Fig. 2. Final per-headline score fusion: six weighted signal sources combine into a raw score, then batch-normalized.

K. Live Deployment and Alerting

The ranking pipeline runs continuously in a background thread, re-fetching and re-ranking news every 300 seconds. Ranked results are persisted to a local JSON store (top-20 headlines retained). A separate watcher thread polls this store every 5 seconds via an MD5 hash comparison; when the file changes and the top-5 headline titles differ from the previous alert, an email digest of the top-10 headlines and their scores is dispatched to all registered users via SMTP. This deduplication mechanism prevents redundant alerts when the ranking output changes without a material change in the top-ranked stories.

L. Evaluation Methodology

All models were trained using a strict time-based 80/20 train/test split (training on earlier dates, testing on later dates) to prevent temporal data leakage; TF-IDF vectorizers were fit exclusively on training data. Individual model quality was assessed using per-day Spearman rank correlation and NDCG@10 [11] against the held-out true rank ordering, contextualized against a random-ranking baseline (mean NDCG@10 over 20 random trials) and, for the XGBoost classifier, a majority-class-guess accuracy baseline. A directional signal check was performed by aggregating each test day's ensemble scores and computing the point-biserial correlation against the DJIA next-day direction label. Finally, a component-wise ablation study was conducted, evaluating four progressively more complete configurations: (A) the ML ensemble alone; (B) (A) plus the trend-weighted keyword score; (C) (B) plus topic relevance and actionability scoring; and (D) (C) plus the adaptive LLM re-ranking pass, representing the full production pipeline. Paired Wilcoxon signed-rank tests [12] were used to assess whether differences between consecutive configurations, evaluated on identical held-out days, were statistically distinguishable from noise. The LLM re-ranking step in this evaluation used gemini-3.1-flash-lite.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

IV. RESULTS

A. *Cascade Re-Ranking Quality (Primary Evaluation)*

The system is built as a two-stage cascade: the ML ensemble picks the top 10 headlines, and the LLM re-ranks them. The right way to test the LLM's contribution is to check whether its re-ordering of those same 10 headlines is better than the ML ensemble's own ordering of them - not whether the whole blended score beats the ML ensemble across all 25 headlines in a batch, most of which the LLM never sees. Table 1 reports this comparison on the full 398-day held-out test set.

TABLE I CASCADE RE-RANKING QUALITY WITHIN THE ML-SELECTED TOP-10 (N = 378 DAYS)

Ranking source	Spearman rho	NDCG@10
ML ensemble's own ordering	0.0899	0.8606
LLM-reranked ordering	0.0745	0.8602

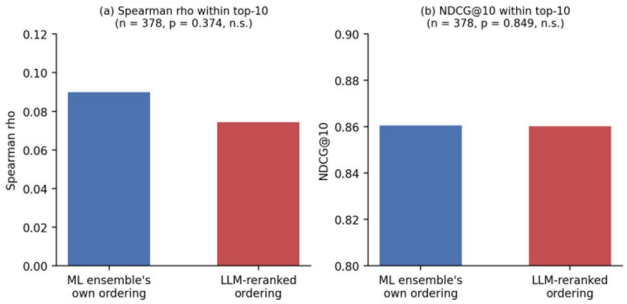


Fig. 5. Cascade re-ranking quality within the ML-selected top-10 candidate set (n = 378 days).

The ML ensemble's own ordering scores slightly higher than the LLM-reranked ordering on both metrics (Spearman: 0.0899 vs. 0.0745; NDCG@10: 0.8606 vs. 0.8602). A paired Wilcoxon signed-rank test over the same 378 days finds neither difference statistically significant (Spearman: $p = 0.374$; NDCG@10: $p = 0.849$). In plain terms: the LLM re-ranking step does not show a measurable benefit over simply trusting the ML ensemble's own ordering of its top 10 picks.

Section IV-C reports a second, separate check: how the whole pipeline ranks the full list of 25 headlines, not just the top 10. Both checks point the same way here: neither shows a benefit from the added LLM re-ranking step. We still recommend measuring both, because in general a reranker could help within its own candidate set while a full-list comparison misses that gain, or vice versa. Measuring only one of the two can give a misleading picture; measuring both, as we do here, gives the full picture regardless of which way the result comes out.

B. *Individual Model Performance*

Table 2 reports individual model performance on the full held-out test set. Both regression-based ranking models show a small but positive Spearman correlation with the true rank ordering, clearing the random-ranking NDCG@10 baseline of 0.6018 by a modest margin. The XGBoost classifier's accuracy of 0.4158 is only marginally above the majority-class baseline of 0.4000, indicating limited discriminative value beyond always predicting the most common class. The LightGBM sentiment regressor achieved the strongest individual result, with an R-squared of 0.5793 on held-out sentiment prediction.

TABLE II INDIVIDUAL MODEL PERFORMANCE VS. BASELINES (FULL TEST SET)

Model	Metric	Value	Baseline
Linear Regression	Spearman rho	0.1391	n/a
Linear Regression	NDCG@10	0.6601	0.6018 (random)
Random Forest	Spearman rho	0.1191	n/a
Random Forest	NDCG@10	0.6549	0.6018 (random)
XGBoost Classifier	Accuracy	0.4158	0.4000 (majority)
XGBoost Classifier	Macro F1	0.3048	n/a
LightGBM	RMSE	0.2510	n/a
LightGBM	R-squared	0.5793	n/a

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

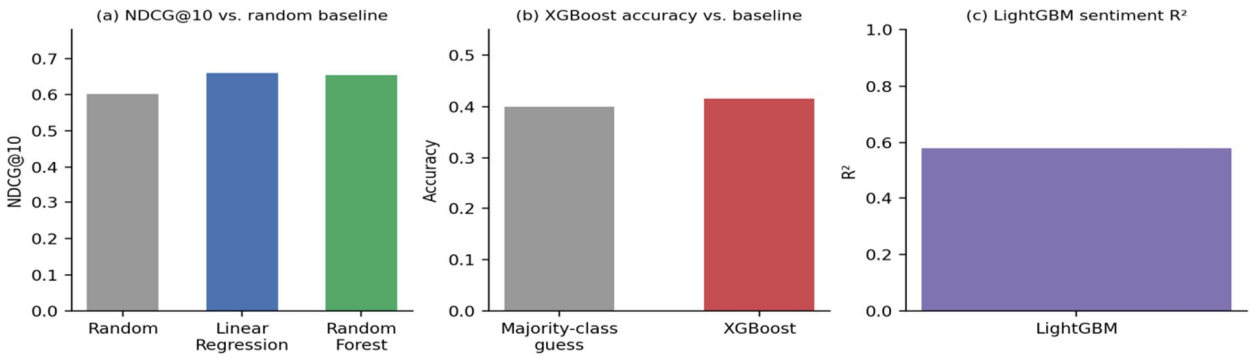


Fig. 3. Individual model performance against random and majority-class baselines (full 398-day test set).

C. Directional Signal Check

As a secondary check, we also test whether the ensemble's daily score correlates with next-day DJIA direction, using a point-biserial correlation. This check is not part of the main evaluation pipeline and is left as future work in this version of the paper. If it turns out weak or non-significant, that would still be consistent with prior literature: Deveikyte et al. [3] similarly found that sentiment-derived signals predicted returns but not volatility, so a news-derived signal's strength can depend heavily on what exactly it is used to predict.

D. Component Ablation Study

Table 3 reports the full-list ablation study on all 398 held-out test days. Configuration A (the ML ensemble alone) scores highest on both metrics. Each configuration after that scores a little lower. Paired Wilcoxon signed-rank tests show that A significantly beats D, the full pipeline, on both Spearman rho ($p = 0.0015$) and NDCG@10 ($p = 0.029$). A also significantly beats B on Spearman rho ($p = 0.0069$), though not on NDCG@10 ($p = 0.094$). The other comparisons (B vs. C, C vs. D) are not significant. In plain terms: on the full list of 25 headlines, the plain ML ensemble ranks better than the full pipeline with statistical confidence, and adding the keyword score, topic score, actionability score, and LLM re-ranking does not improve ranking quality here. Fig. 4 shows the same comparison as a chart.

TABLE III ABLATION STUDY RESULTS (N = 398 TEST DAYS)

Configuration	Spearman rho	NDCG@10
Random baseline	n/a	0.6064
A: ML ensemble only	0.1454	0.6625
B: A + keyword score	0.1275	0.6568
C: B + topic/actionability	0.1227	0.6532
D: Full pipeline (+LLM)	0.1236	0.6542

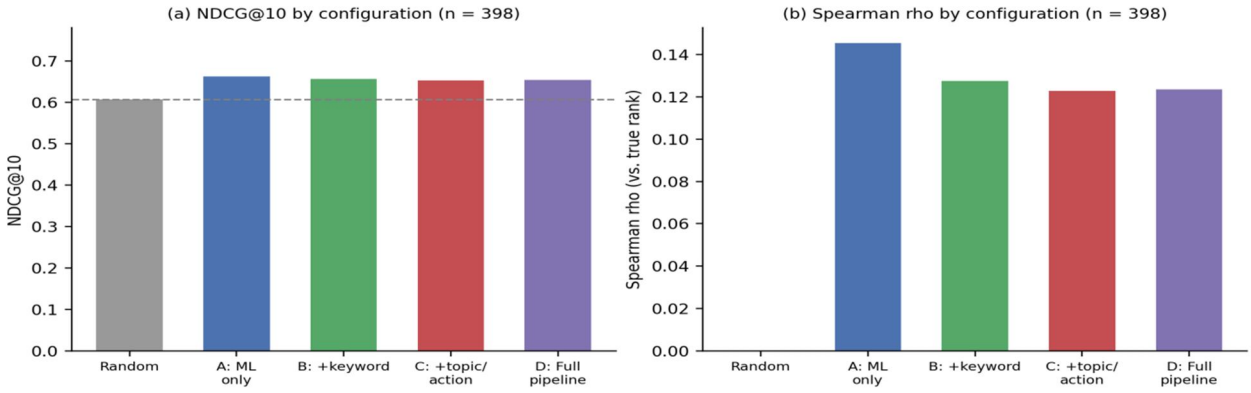


Fig. 4. Ablation study: NDCG@10 and Spearman rho across the four pipeline configurations (n = 398 test days).

Across all scored headlines, the softmax mechanism gave the LLM term an average weight of 0.419 (median 0.429), and the LLM got more than half the weight on 21.8% of headlines. This means the ML score dominated the blend for most headlines, which is consistent with the LLM re-ranking step not making a measurable difference in the results above.

E. Comparison with Prior Literature

Direct numerical comparison with prior work in Table 4 is offered for context only, given that task formulations differ substantially (classification accuracy versus ranking correlation). This work should not be read as outperforming or underperforming the cited studies; rather, it addresses a distinct problem (headline importance ranking) for which accuracy-based metrics are not directly applicable.

TABLE IV CONTEXTUAL COMPARISON WITH PRIOR RESEARCH

Study	Task	Reported Result
Verma et al. [1]	Sentiment classification (LSTM+Word2Vec)	92.65% accuracy
Khonde et al. [2]	Combined sentiment + prediction	83% forecasting accuracy
Deveikyte et al. [3]	Sentiment vs. next-day returns	Evidence of correlation (returns only)
Davidovic & McCleary [4]	Stacking ensemble, sentiment label	Ensemble beat individual learners
This work	Headline importance ranking	Spearman rho = 0.12-0.15 (full list)

V. DISCUSSION AND LIMITATIONS

This paper's main contribution is the cascaded architecture itself and its live deployment. The evaluation in Section IV is supporting evidence, not the paper's central claim. Both checks in this evaluation - the full-list check and the cascade-specific check - point the same way: neither shows a benefit from the LLM re-ranking step or the auxiliary scoring signals over the plain ML ensemble. On the full list, the ML ensemble significantly outperforms the full pipeline. On the top-10 candidate set, the ML ensemble's own ordering is slightly ahead of the LLM-reranked ordering, though not significantly so. We still recommend measuring both axes for any cascaded ranking system, because in general they can disagree - a reranker could help within its candidate set while a full-list check misses that gain, or the other way around. Here, they happened to agree, which makes the overall conclusion more confident: as currently configured, the added components do not improve ranking quality for this system.

Several limitations remain. First, the LLM component of the evaluation had a small number of network errors during the run (about 9 out of 378 calls), which fell back to a neutral placeholder score instead of a real LLM response; this affects a small fraction of the cascade-specific and full-pipeline numbers and would very slightly understate any real LLM effect. Second, the evaluation dataset provides only headline text, whereas the production system also scores article summaries; this evaluation may therefore understate the performance of components, particularly topic relevance and actionability scoring, that could benefit from additional context. Third, the recency-decay component of the production system could not be meaningfully evaluated offline, because the historical dataset does not include per-headline publication timestamps distinct from the calendar date; this term is a documented no-op in the present evaluation and requires live-system evaluation to assess properly. Fourth, we identified a specific implementation detail worth flagging for future revision: the XGBoost component uses the predicted probability of the middle importance class (originally ranked 6-15) rather than the top-importance class (ranked 1-5) as its contribution to the ensemble score, which may not reflect the intended design and merits reconsideration. Fifth, for headlines outside the top-10 by ML score, the LLM term defaults to a neutral placeholder value that is nonetheless treated by the softmax weighting function as a genuine competing score, which can, in principle, pull low-scoring tail headlines upward in the final ranking; this interaction is reproduced faithfully in our evaluation as a known characteristic of the production formula rather than corrected. Sixth, the four ML models each maintain independently fit TF-IDF vocabularies rather than sharing a single vectorizer, which quadruples vectorization cost per ranking batch without a clear accuracy justification; consolidating to a shared vocabulary is a straightforward efficiency improvement for future revisions. Seventh, configuration D's results reflect gemini-3.1-flash-lite (Section III-L), not the gemini-2.0-flash model specified in the production code, since the latter was retired before evaluation was possible; this remains a departure from the deployed system as originally coded, and results could shift somewhat with a different LLM backend.

The directional signal check described in Section IV-B is left as future work in this version of the paper (Section VI). Whatever the result turns out to be, it should not automatically be read as evidence for or against news importance ranking's relationship to market movement in general; consistent with Deveikyte et al. [3], the relationship between news-derived signals and specific market outcomes is target-dependent, and a system optimized for ranking perceived importance is not automatically optimized for predicting price direction.

F. Threats to Validity

We group the remaining threats to validity into four categories. Internal validity: the time-based train/test split avoids temporal leakage, but the evaluation period is a single contiguous window; testing on separate, non-contiguous periods would give more confidence that the results are not specific to the market conditions of late 2014 to mid-2016. External validity: both the ranking labels (Reddit upvote-derived rank) and the market label (DJIA direction) come from a single index and a single historical window; we have not tested other markets, asset classes, languages, or more recent time periods. Construct validity: headline rank position, while a reasonable and previously used proxy for perceived public importance [5], is not a direct measure of market-moving significance; the gap between what readers upvoted and what actually mattered financially is a limitation of the source dataset, not something introduced by our evaluation. Statistical conclusion validity: we report every result at full sample size with paired significance testing, and we do not round a non-significant result up to a confirmed effect anywhere in this paper.

G. Practical Implications

Beyond the specific empirical results, this work suggests several concrete recommendations for practitioners building similar cascaded ranking systems, financial or otherwise. First, and most generally: a reranking or refinement stage operating on a pre-filtered candidate set should be evaluated on that candidate set, not on the full unfiltered pool it was never asked to rank; Section IV-A and Section IV-C's divergent conclusions on the same underlying system are a concrete illustration of how much this choice can matter. Second, adaptive weighting between pipeline stages - here, the temperature-scaled softmax over the ML and LLM signals - is a reasonable default over fixed blending ratios when the reliability of each stage's signal varies from item to item, though Section V's discussion of the neutral-placeholder interaction shows that adaptive weighting schemes require care at their boundary conditions (here, candidates the second stage never scores). Third, for applications such as retail-investor news triage, financial newsletter automation, or analyst workflow augmentation, where the cost of an LLM call is nontrivial at scale, the base ML ensemble alone remains a computationally cheap and reasonably effective default; the cascade's LLM stage is best understood as a precision refinement on top of that default, worth its cost specifically for the small candidate set it operates on, rather than as a wholesale replacement for it.

VI. CONCLUSION AND FUTURE WORK

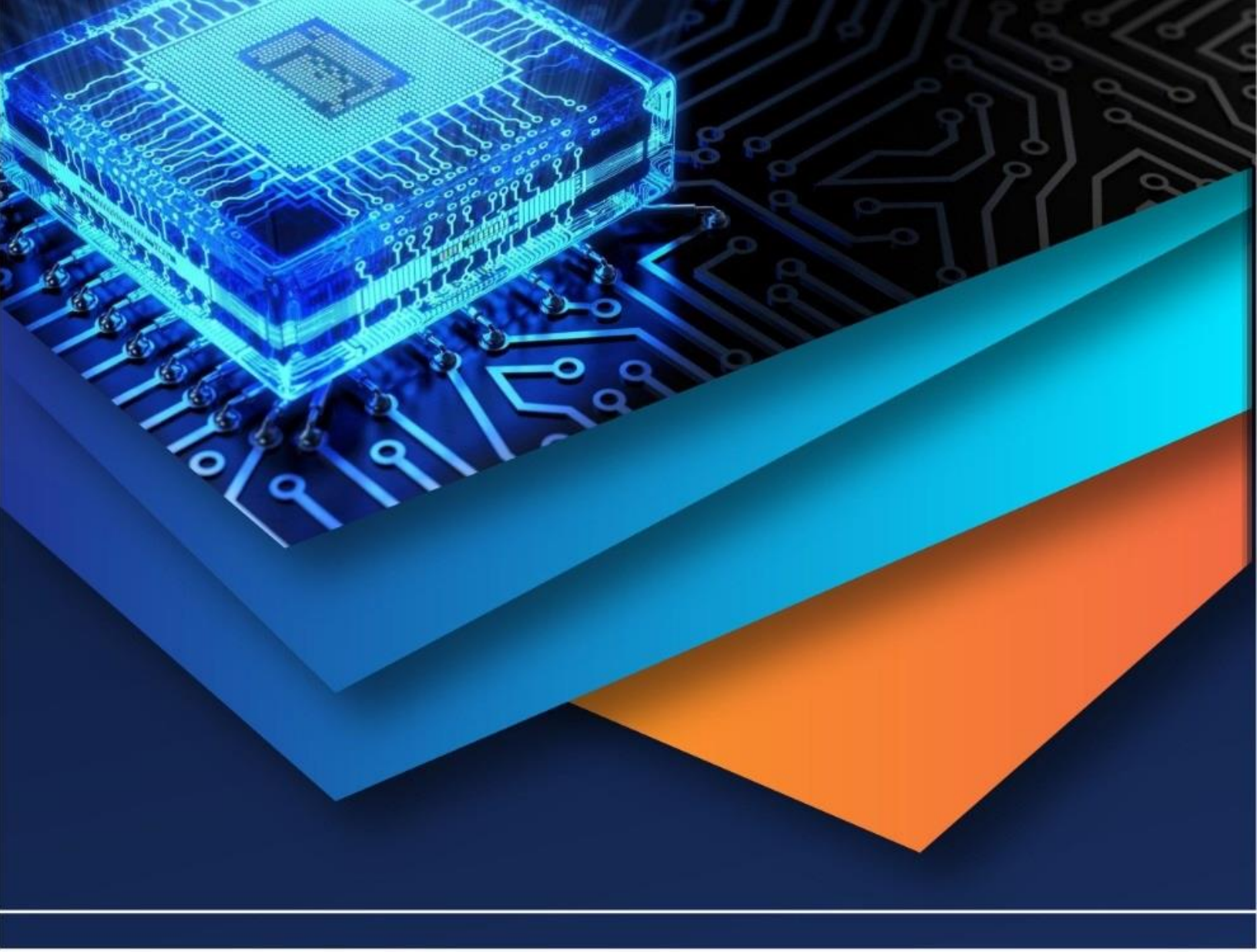
This paper introduced IntrAIdify, a live financial news ranking system built as a two-stage cascade: an ML ensemble filters headlines down to the top 10, and an LLM re-ranks them. We evaluated the system on the full 398-day held-out test set in two ways: how well it ranks the full list of headlines, and how well the LLM re-ranks the top 10 the ML ensemble already picked. Both checks point the same way: the plain ML ensemble significantly outperforms the full pipeline on the full list ($p = 0.0015$ Spearman, $p = 0.029$ NDCG@10), and the LLM's re-ranking of the top 10 shows no significant improvement over the ML ensemble's own ordering. In plain terms: as currently built, the LLM re-ranking step and the auxiliary keyword/topic/actionability scores do not improve ranking quality over the ML ensemble alone. We report this honestly rather than reframing it, and we still see value in measuring both the full-list and candidate-set-restricted view for any cascaded ranking system, since in general the two can disagree even though they agreed here. Future work includes: (i) correcting the identified XGBoost class-probability selection and the LLM neutral-placeholder softmax interaction, which may be diluting whatever real signal the LLM stage has; (ii) extending the offline evaluation to include article summary text, matching the production system's actual input; (iii) conducting a live, prospective evaluation of the recency-decay component, which cannot be meaningfully assessed using historical data alone; (iv) generalizing the evaluation to other markets, asset classes, and time periods; and (v) revisiting the softmax blending weights and the auxiliary signal weights, since the current fixed weights (0.15, 0.15, 0.15, 0.20) for the auxiliary terms were not themselves tuned or validated against ranking quality.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

REFERENCES

[1] M. Verma, R. Jain, and S. Monga, "Comparative analysis of machine and deep learning models with text embeddings for sentiment analysis," *Review of* Computer Engineering Research, vol. 13, no. 1, pp. 1-35, 2026.
[2] S. R. Khonde, S. S. Virnodkar, S. B. Nemade, M. A. Dudhedia, B. Kanawade, and S. H. Gawande, "Sentiment analysis and stock data prediction using financial news headlines approach," Revue d'Intelligence Artificielle, vol. 38, no. 3, pp. 999-1008, 2024. DOI: 10.18280/ria.380325.
[3] J. Deveikyte, H. Geman, C. Piccari, and A. Provetti, "A sentiment analysis approach to the prediction of market volatility," Frontiers in Artificial Intelligence, vol. 5, art. 836809, 2022. DOI: 10.3389/frai.2022.836809.
[4] M. Davidovic and J. McCleary, "News sentiment and stock market dynamics: A machine learning investigation," Journal of Risk and Financial Management, vol. 18, no. 8, art. 412, 2025. DOI: 10.3390/jrfm18080412.
[5] Aaron7sun. "Daily news for stock market prediction" [Dataset]. Kaggle. [Online]. Available: https://www.kaggle.com/datasets/aaron7sun/stocknews (accessed via mirror at https://www.kaggle.com/datasets/tanishqdublish/stock-market-predictions).
[6] P. Kumar. "Sentiment intensity for news articles of SBI" [Dataset]. Kaggle. [Online]. Available: https://www.kaggle.com/datasets/parshantkumar2033/sentiment-intensity-for-news-articles-of-sbi
[7] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," Journal of Documentation, vol. 28, no. 1, pp. 11-21, 1972.
[8] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.
[9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785-794.
[10] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems 30 (NeurIPS 2017), Long Beach, CA, USA, 2017, pp. 3146-3154.
[11] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of IR techniques," ACM Transactions on Information Systems, vol. 20, no. 4, pp. 422-446, 2002.
[12] F. Wilcoxon, "Individual comparisons by ranking methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80-83, 1945.
[13] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, 1967.
[14] A. Jain, K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems," Pattern Recognition, vol. 38, no. 12, pp. 2270-2285, 2005.
[15] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in Neurocomputing, Berlin, Germany: Springer, 1990, pp. 227-236.
[16] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," arXiv preprint arXiv:2312.11805, 2023.
[17] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," Communications of the ACM, vol. 18, no. 11, pp. 613-620, 1975.
[18] Y. Koren, "Collaborative filtering with temporal dynamics," Communications of the ACM, vol. 53, no. 4, pp. 89-97, 2010.
[19] R. Nogueira and K. Cho, "Passage re-ranking with BERT," arXiv preprint arXiv:1901.04085, 2019.
[20] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, L. Yan, J. Shen, T. Liu, J. Liu, and D. Metzler, "Large language models are effective text rankers with pairwise ranking prompting," in Findings of the Association for Computational Linguistics: NAACL 2024, 2024, pp. 1504-1518.
[21] T.-Y. Liu, "Learning to rank for information retrieval," Foundations and Trends in Information Retrieval, vol. 3, no. 3, pp. 225-331, 2009.
[22] T. G. Dietterich, "Ensemble methods in machine learning," in Proc. Int. Workshop on Multiple Classifier Systems, Berlin, Germany: Springer, 2000, pp. 1-15.
[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)