Pranshu Rastogi @ FinMMEval 2026: Systems for Tasks 1 and 2 - Prompt-Engineered Gemini for Multilingual and Cross-Lingual Financial Reports QA*

FinMMEval Lab at CLEF 2026.

Pranshu Rastogi[1]

[1] *Independent Researcher, India*

Abstract

This paper describes our submission to FinMMEval 2026 at CLEF, covering Task 1 (multilingual financial multiple-choice question answering across English, Arabic, Chinese, and Hindi) and Task 2 (cross-lingual financial report and news question answering). The system runs entirely on Google Gemini 3.x without fine-tuning; all task knowledge comes from structured prompt engineering, language-specific few-shot examples, and per-question routing. For Task 1, we combine question-type detection with cost-gated self-consistency voting, invoking N=5 plurality voting only on the ~10–20% of items flagged as uncertain by a multilingual confidence gate. For Task 2, we adopt a tier-aware prompting strategy in which Easy questions are answered from the embedded financial statements with statement-level evidence, while Expert questions synthesis the five-language news bundle with verbatim, language-labelled quotes. Calculation-heavy items use a two-pass extraction-then-synthesis procedure to anchor answers on exact table figures. We discuss the engineering principles, multilingual normalization steps, and ROUGE-1-oriented output discipline that drove our design. Code: https://github.com/pranshurastogi29/finmmeval

Keywords

multilingual question answering, financial NLP, large language models, prompt engineering, self-consistency, cross-lingual reasoning, cot-reasoning

1. Introduction

The financial domain poses a tough challenge for large language models. It combines specialized terminology, jurisdictional variation, precise numerical reasoning, and—increasingly—multilingual context, since major capital markets, regulatory regimes, and corporate disclosure regimes operate in different languages. The FinMMEval 2026 shared task at CLEF [1] operationalizes this by pairing two complementary challenges in one benchmark: a multilingual multiple-choice examination in English, Arabic, Chinese, and Hindi [2], and a cross-lingual question-answering task in which English financial statements are paired with news articles in five languages [3].

The two tasks evaluate distinct capabilities. Task 1 probes whether a model holds calibrated financial knowledge across four linguistically and regulatory distinct ecosystems—the CFA/CPA corpus that underlies Anglophone finance, the Shari'ah-compliant frameworks captured by SAHM [4], the curriculum encoded in RealFin [5] for Chinese markets, and the Indic financial concepts surveyed by BhashaBench [6]. Task 2 stresses cross-lingual synthesis: the model must extract precise figures from English statements, integrate them with news evidence in five languages drawn from MultiFinBen [7], and produce a structured, evidence-bearing answer that closely matches expert reference text.

We approached both tasks with a strict constraint: **no fine-tuning, no retrieval, single frontier model**. Fine-tuning is brittle under distributional drift, expensive to maintain, and hard to

* *CLEF 2026 Working Notes, 21 — 24 September 2026, Jena, Germany*

[1] rastogipranshu29@gmail.com (P. Rastogi)

reproduce across model releases. Meanwhile, the embedded inputs in both tasks (questions for Task 1, statements plus news bundles for Task 2) already contain the information needed to answer correctly; the bottleneck is *eliciting* the right reasoning, not augmenting the context. We therefore invest engineering effort in prompt construction, question-type routing, multilingual normalization, and inference-time strategies (tiered self-consistency, two-pass extraction), all built on Google Gemini 3.1 Pro [10].

Three design principles guide our system. **Modular prompts**: each language and each tier receives a dedicated system prompt that encodes domain knowledge and few-shot exemplars, enabling implicit prefix caching to amortize input cost. **Cost-gated reasoning**: expensive inference strategies (self-consistency, two-pass extraction) are triggered only when cheap heuristics indicate they are needed. **Output discipline as a first-class objective**: for Task 2, we engineer the prompts to emit text in the format that ROUGE-1 rewards—exact dollar figures, standard period notation, verbatim multilingual quotations.

The remainder of this paper is organized as follows. Section 2 describes the FinMMEval datasets and the two task formulations. Section 3 details our methodology, structured around shared model configuration, the Task 1 pipeline, and the Task 2 pipeline. Section 4 covers the experimental setup, including infrastructure, resilience, and validation procedures. Section 5 reports and analyses our results. Section 6 concludes with reflections on what worked, what did not, and where future improvements lie.

2. Background and Task description

2.1 Task 1: Multilingual Financial Multiple-Choice QA

Task 1 [2] consists of 200 multiple-choice questions in each of four languages—English, Arabic, Chinese, and Hindi—yielding an 800-item test set. Most questions follow the standard four-option MCQ format; a minority of Arabic items are True/False, encoded with صح ("correct") and خطأ ("incorrect") as options A and B. Accuracy is the official metric.

Despite the uniform surface format, the questions vary a lot. During development we identified five recurring categories: **recall** of definitions and concepts, **negation-bearing** questions (containing *NOT*, *EXCEPT*, or their multilingual equivalents) that require choosing the option that does *not* apply, **numerical calculations** requiring multi-step arithmetic, **multi-statement** items in which several propositions are jointly evaluated, and **True/False** items in Arabic. Each category calls for a different prompting strategy: recall questions do best with minimal chain-of-thought to avoid hallucinated elaboration, while calculations benefit from structured scratchpads.

The questions further differ in their regulatory and conceptual context. English items frequently reference CFA and CPA-style scenarios; Arabic items invoke SOCPA standards, GAAP, and Shari'ah-compliant instruments [4]; Chinese items track the curriculum encoded in RealFin [5]; and Hindi items, drawn from BhashaBench [6], require familiarity with the Reserve Bank of India framework, Indian taxation, and domestic capital-markets terminology. A single shared prompt cannot handle this variation well, so we use a per-language prompt design (§3.2.1).

2.2 Task 2: Cross-Lingual Financial Report and News QA

Task 2 [3] comprises 256 items, evenly split between an **Easy** tier (128 items, answers grounded in financial statements) and an **Expert** tier (128 items, answers grounded in news synthesis). Each item bundles three components into a single query: (i) the income statement, balance sheet, and cash-flow statement of the target company, rendered as Markdown tables in English; (ii) five news articles about the company in English, Chinese, Japanese, Spanish, and Greek; and (iii) the question with a free-text answer template. The official metric is ROUGE-1 against expert reference answers, with a hard cap of 100 words on the answer body.

Looking at the development gold answers (parquet files, offline only), we found a clean separation between tiers. Easy answers consistently quote exact figures from the financial statements with precise column names (e.g., *Service and other*, *Additions to property and equipment*) and a structured Financial Statements Evidence: block. Expert answers, by contrast, draw conclusions from the news bundle—often stating *None* when the news is silent on the topic—and append a News Evidence: block containing verbatim, language-attributed quotations. This split between tiers is what drives our Task 2 pipeline design.

Within each tier, four question templates recur: for Easy, revenue trends, balance-sheet health, cash-flow irregularities, and R&D ratios; for Expert, top-3 revenue focuses, capital allocation, profit-margin strategy, and CapEx significance. The templates differ in whether they demand extraction, comparison, or higher-order synthesis, which is why we use a two-pass strategy for calculation-heavy items (§3.3.4).

---

3. Methodology

3.1 Shared Configuration

Model selection. All inference is performed with Google Gemini 3.1 Pro (preview), accessed through the google-genai Python SDK [10]. A secondary path uses Gemini 3.5 Flash, selectable via the GEMINI_MODEL environment variable; we use the Pro tier for the official submission given its stronger reasoning on Task 1's calculation items and Task 2's Expert synthesis.

Thinking levels. Gemini 3.x exposes discrete reasoning effort levels—minimal, low, medium, high—in place of an explicit token budget for hidden chain-of-thought. We allocate effort per pipeline phase: accuracy-critical stages (Task 1 main and self-consistency calls, Task 2 Expert synthesis) use high; Task 2 Easy uses medium; mechanical sub-passes (Task 2 number extraction, Task 1 Tier-2 fallback) use low. Since thinking tokens are billed as output tokens, we use the cheapest level that works for each stage.

Temperature. We do not set temperature, intentionally accepting Gemini 3's default of 1.0. Lower temperatures reportedly hurt Gemini 3 reasoning, and the default stochasticity already gives us the sample diversity we need for self-consistency voting in Task 1.

Token budgets. Output limits are set generously to accommodate hidden thinking tokens: 8,000 tokens for Task 1 (high thinking plus a one-letter answer), 4,000 for Task 2 main answers (high thinking plus a 100-word answer), and 2,000 for Task 2 extraction sub-passes.

Implicit prefix caching. Gemini 2.5+ automatically caches stable input prefixes. We exploit this by constructing prompts in which the *system instruction* is invariant across all calls in a language (Task 1) or tier (Task 2), and only the user message varies per question. Context hints in Task 2 are appended to the user message, never to the system instruction, to preserve the cache prefix. We do not create explicit CachedContent objects, which would impose minimum-token thresholds that our shorter Task 2 system prompts do not meet.

3.2 Task 1 Pipeline

3.2.1 Language-Specific System Prompts with Embedded Few-Shot

Each of the four languages receives a dedicated system prompt with three components: (1) a domain primer—CFA/CPA concepts and U.S. regulatory framing for English, GAAP/SOCPA and Shari'ah-compliant instrument notes for Arabic, the RealFin curriculum for Chinese, and RBI/Indic-context content for Hindi; (2) four solved exemplars enclosed in <example> blocks, demonstrating

the full reasoning trajectory from question parsing to answer extraction; and (3) instructions to reason **in the question's language**, which improved our development-set accuracy on Arabic, Chinese, and Hindi over English-only chain-of-thought.

Placing examples in the system prompt rather than per-message is intentional. Across 200 questions per language, the system prefix is byte-identical, so implicit caching pays the input cost once and serves subsequent calls at the cache-read rate. The marginal value of additional exemplars therefore amortizes across the entire test partition.

3.2.2 Question-Type Detection and Routing

Each question is classified into one of five categories at runtime, and a category-specific instruction suffix is appended to the user message. Detection uses lightweight regular expressions that operate on the question text and option list:

Category	Detection rule	Tailored instruction
true_false	Options are صح/خطأ or surface equivalents	Emit the canonical letter only
except_not	Presence of *NOT*, *EXCEPT*, لا يُعد, 不是, नहीं	Reframe as "find the option that does *not* apply"
calculation	Digits, %, currency, or *calculate*/حساب/计算/गणना	Structured scratchpad: *Given* → *Formula* → *Substitution* → *Arithmetic* → *Answer*
multi_statement	Patterns such as "I ... II", *Statement 1/2*, كثن, 陈述	Evaluate each statement, then select
recall	Default	Minimal chain-of-thought

For all non-True/False categories, we additionally require an **elimination step**: the model must justify rejecting each incorrect option before producing the final answer on the last line. Elimination raises accuracy on adversarial distractors and makes downstream answer extraction more robust.

3.2.3 Arabic True/False Normalisation

Arabic True/False items are tricky: the options are encoded as A=صح and B=خطأ, but Gemini occasionally emits synonyms (صواب, خاطئ), unvocalized variants, or the English equivalents *True*/*False*. We curated a synonym map _AR_TF_SYNONYMS mapping each variant to the corresponding canonical option key. The map is applied as the first step of our answer-extraction cascade, ensuring that morphologically legitimate Arabic answers are not silently scored as incorrect.

3.2.4 Cost-Gated Tiered Self-Consistency

Self-consistency voting [11] can raise accuracy on reasoning tasks, but running it on every question is wasteful when the model is already confident. We use a three-tier approach that ramps up inference cost only when cheaper passes leave doubt:

- **Tier 0 (single high-thinking call).** The default path. If the returned answer is a valid label and contains no hedging language, it is accepted.
- **Tier 1 (N=5 self-consistency vote).** Triggered by an *uncertainty gate* that fires when (a) the Tier 0 answer is not a valid option label after extraction, or (b) the answer text contains multilingual hedging markers—*could be either*, يمكن أن يكون, 可能是, हो सकता है, and related phrases. We make five parallel calls at high thinking, then compute a ranked plurality vote with ties broken by first-seen order.
- **Tier 2 (stripped low-thinking fallback).** If the vote produces no valid winner, we issue a final call with a minimal stripped prompt (no exemplars, no routing suffix) at low thinking, accepting whatever single letter is returned.

In our development analysis, the gate triggered Tier 1 on roughly 10–20% of items, concentrating compute on genuinely ambiguous questions while leaving the confident majority on the cheap Tier 0 path.

3.2.5 Answer Extraction Cascade

Even with explicit format instructions, model outputs sometimes wrap the answer in unexpected scaffolding. We apply six extraction strategies in decreasing order of specificity: (1) Arabic T/F synonym lookup, (2) exact match against the option label, (3) match on a Final answer: X template, (4) last-line letter extraction, (5) leading-letter scan, and (6) a full-text scan for any valid option letter as a final fallback. In practice, the first three strategies handle most well-formed outputs; the last three recover answers buried in verbose chain-of-thought tails.

3.3 Task 2 Pipeline

3.3.1 Self-Contained Inputs and the No-Retrieval Stance

Each Task 2 item is a self-contained payload: the question is preceded by the company's full income statement, balance sheet, and cash-flow statement as Markdown tables, followed by five news articles in different languages. We deliberately do *not* perform retrieval; the inputs already exceed what naive RAG would surface, and adding a retriever introduces a second source of error without addressing the dominant failure modes we observed (numeric rounding, evidence-label drift, missing companies in expert "None" cases).

3.3.2 Tier-Aware Prompting

The Easy and Expert tiers expect structurally different answers, and we maintain separate system prompts accordingly. Easy answers must quote *exact* figures from the financial statements and append a Financial Statements Evidence: block citing the relevant rows. Expert answers must synthesize the news bundle, attach a News Evidence: block containing verbatim quotations with language labels (e.g., *Japanese news: "..."*), and emit Answer: None. when the news provides no substantive content—a convention we observed consistently in the development gold.

3.3.3 Lightweight Context Hints

Before each Task 2 call, we prepend a compact [CONTEXT HINTS] block to the user message. The block contains three derived fields:

- **Company name**, canonicalized from the English news article (Microsoft, Honeywell, Johnson & Johnson, Universal Corporation, Alibaba; heuristic fallback for unseen companies). Gold answers in our development analysis almost universally open with *"[Company]'s ..."*, so seeding the company name materially raises ROUGE-1 recall on the opening tokens.

- **Filing type**, detected by searching for *Year Ended* (annual) versus quarter markers in the table headers. Annual filings are instructed to cite up to three fiscal years; quarterly filings cite current-quarter versus year-ago-quarter pairs.
- **Available fiscal years**, parsed from the table column headers. This guards against the model inventing periods not present in the data.

Because the hints vary per question, they live in the user message, leaving the system prompt cacheable.

3.3.4 Two-Pass Extraction for Arithmetic Items

R&D-ratio and balance-sheet-health questions are arithmetic and sensitive to rounding. Single-pass generation occasionally produced plausible-looking but slightly miscomputed numbers, hurting both correctness and ROUGE-1 against the gold answers (which quote precise figures). For these question templates, we use a two-pass procedure:

1. **Extraction pass (low thinking).** The model is given the tables and asked to emit only the raw numerator and denominator (e.g., the exact R&D expense and total revenue for each fiscal year in the table), without performing the calculation.
2. **Synthesis pass (tier-appropriate thinking).** The model receives the extracted numbers and is instructed to compute and format the final answer.

Decomposing the task this way prevents the model from amortizing its thinking budget across both extraction and synthesis simultaneously, which is what caused single-pass generation to stumble on these items.

3.3.5 ROUGE-1 Output Discipline

Because the official Task 2 metric is unigram overlap with expert references, prompt design that *predicts* expert phrasing pays a direct metric dividend. Our system prompts therefore enforce a battery of formatting conventions distilled from development-gold analysis: dollar figures as $X.XB for amounts ≥ 1 billion and $X,XXX million otherwise; period notation as Q3 FY2020 and FY2020; news quotations rendered verbatim with explicit language labels; column names quoted exactly as they appear in the statements; the Answer: None. convention for empty-news Expert items; and a hard 100-word cap on the answer body.

A small post-processing step (_ensure_answer_prefix) enforces that every stored output begins with Answer:. Gemini does not support assistant-side pre-filling, so we enforce this in code rather than through prompt scaffolding.

4. Experimental Setup

Datasets. Task 1 uses the four official test files ({english,arabic,chinese,hindi} _task1_final_test_public.jsonl, 200 items each). Task 2 uses task2_test_questions.jsonl (256 items). Development data—the official Task 1 dev sets and the Task 2 dev parquet files containing gold answers—was used *only* offline to design prompts, analyze gold-answer conventions, and select few-shot exemplars; the gold parquet files are never read at inference time.

Inference scripts. Two lightweight orchestration scripts (infer_task1.py and infer_task2.py) implement the pipelines above. Both write predictions to JSONL files in submissions/, flushing after each item so an interrupted run loses no work. Both also **resume** from a partial output file by skipping IDs already present, which lets us re-run after model swaps or rate-limit interruptions without restarting from scratch.

Throughput control. We insert a fixed delay between calls (0.5 s for Task 1, 0.8 s for Task 2) to stay within API rate limits. Per-item exceptions are caught, logged, and recorded as ERROR (Task 1) without aborting the batch, so a transient failure on a single item does not contaminate the submission.

Validation. A custom script validate_submissions.py runs after each batch and checks for the failure modes that the official scorer flags as invalid: missing or duplicate IDs, unknown IDs, empty or ERROR-sentinel answers, Task 1 labels outside the allowed option set, and Task 2 answers exceeding the 100-word cap or violating the required format. A clean exit code is a precondition for submission.

Estimated cost. With Gemini 3.1 Pro pricing of $2/$12 per million input/output tokens (≤200K prompts) and implicit caching active, our full submission consumes approximately 800–1,400 calls on Task 1 (including ~10–20% Tier-1 expansion) and approximately 320 calls on Task 2 (192 single-pass plus 64 two-pass), for a total estimated cost on the order of $13 USD. Gemini 3.5 Flash is priced approximately 5× lower per token than 3.1 Pro on both input and output, so a full Flash submission would fall to the order of $2–3; we did not run a controlled Flash-vs-Pro comparison on the official test set, so we cannot report a quantified accuracy delta and treat this as an open question (§5.4).

---

5. Results

5.1 Task 1: Accuracy by Language

Table 1 reports our official test-set accuracy on Task 1. We list per-language scores alongside the overall average across the 800 items.

Table 1: Task 1 — Accuracy by language

Language	# Items	Accuracy
English	200	**91.00%**
Arabic	200	**95.00%**
Chinese	200	**92.00%**
Hindi	200	**91.50%**
Overall	**800**	**92.38%**

Several observations from our development analysis carried over to the test set. First, the cost-gated self-consistency triggered Tier 1 on a small fraction of items (in line with our 10–20% target); among those, plurality voting flipped the predicted answer in a fair number of cases, most of which appeared to be corrections rather than regressions based on our offline checks. Second, Arabic True/False questions were highly sensitive to the synonym-normalization step (§3.2.3): without it, surface variants would have been scored as wrong despite being semantically correct. Third, calculation questions stayed the hardest category across all four languages; the structured scratchpad helped but did not fully close the gap with recall questions.

5.2 Task 2: ROUGE-1 by Tier

Table 2 reports our official test-set ROUGE-1 on Task 2, separating Easy and Expert tiers.

Table 2: Task 2 — ROUGE-1 by tier

Metric	Samples	Score
Precision	256	0.3208
Recall	256	0.3537
ROUGE-1	**256**	**30.96%**

The Easy tier benefited most visibly from the context-hint pipeline (§3.3.3): pre-seeding the company name and available fiscal years pushed unigram overlap upward on the opening tokens of the answer, where gold references most reliably converge on a fixed phrasing. The two-pass extraction strategy (§3.3.4) was applied only to the arithmetic templates; on the remaining Easy templates, single-pass generation with high-quality formatting instructions was sufficient.

The Expert tier proved more challenging, as expected. The main failure modes were (a) substantive content drift when the news bundle covered the question's topic only obliquely, and (b) Answer: None. calibration—both false positives (the model under-asserts when relevant content exists) and false negatives (the model hallucinates content when the gold answer is *None*). The tier-specific evidence-label conventions held up well, however, and verbatim language-attributed quotes (§3.3.5) reliably appeared in the News Evidence: blocks.

5.3 Cost-Accuracy Discussion

Two choices drove most of our cost savings. Implicit prefix caching dropped the effective input cost of Task 1 calls after the first request per language, since the per-language system prompts run to several thousand tokens of domain primer plus exemplars. Cost-gated self-consistency, in turn, kept Tier 1 expansion confined to the small fraction of genuinely uncertain items, rather than scaling N×5 across the full 800-item set. Together, these keep the submission cost roughly 10× below a naïve run with universal high-thinking and N=5 voting everywhere.

5.4 Limitations

Our submission is intended as a pragmatic system report, not a controlled scientific study, and it inherits two limitations that are important to state directly. The claims that per-language prompts, cost-gated self-consistency, context hints, Arabic True/False synonym normalization, and two-pass extraction each contribute to final accuracy or ROUGE-1 are supported by qualitative development-set observations rather than quantitative on-off comparisons at test time. Second, several Task 2 prompt-engineering choices (dollar-formatting conventions, period notation, verbatim quotation labelling, the Answer: None. idiom, and the 100-word cap) were reverse-engineered from the development gold answers to align with ROUGE-1 conventions. This improves the reported metric but is a form of test-adjacent optimization: a fraction of the ROUGE-1 gain reflects surface phrasing match rather than better underlying reasoning.

6. Conclusion

We described a prompt-engineering pipeline for FinMMEval Task 1 and Task 2, running on Google Gemini 3.1 Pro with no fine-tuning or retrieval. The Task 1 pipeline combines language-specific few-shot system prompts, regular-expression question-type routing, Arabic True/False normalization, and cost-gated tiered self-consistency. The Task 2 pipeline relies on tier-aware prompting that mirrors the evidence-labeling conventions of the development gold, lightweight context hints (company, filing type, fiscal years) that seed key unigrams for ROUGE-1, and a two-pass extraction-then-synthesis procedure for arithmetic items.

Three things stand out. **Per-language prompts beat shared prompts** for Task 1: the regulatory and conceptual gaps across the four languages make a one-size-fits-all prompt clearly worse than a modular design, especially once implicit caching eliminates the input-cost penalty of longer prompts. **Output discipline is a first-class objective for ROUGE-1**: prompts engineered to anticipate expert phrasing yield measurable metric gains even without improving the model's actual reasoning. **Cost gating compounds with implicit caching**: triggering expensive strategies only when uncertainty heuristics fire, on top of a cached prompt prefix, keeps total inference cost about 10× below the brute-force alternative.

We see three directions worth exploring next. First, lightweight retrieval over the embedded news bundle might improve Expert-tier ROUGE-1 for questions whose answers hinge on a single language's article; the current pipeline trusts the model to attend to the relevant article unaided. Second, language-specific calibration of the self-consistency confidence gate—currently a shared multilingual pattern set—could fire Tier 1 more precisely on the languages where Gemini is least confident. Third, expanding the few-shot pool with development-set distillation, while preserving cache locality, could help close the remaining gap on Task 1 calculation items.

Acknowledgements

I thank the FinMMEval 2026 organizers and the broader CLEF community for designing and hosting the shared task. I am grateful to the contributors of the SAHM [4], RealFin [5], BhashaBench [6], and MultiFinBen [7] benchmarks, whose work underpins the test material engaged with in this submission. Inference was carried out via the Google Gemini API.

Declaration on Generative AI

During the preparation of this work, the author(s) used chatGPT in order to: Grammar and spelling check. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication’s content.

References

[1] Z. Xie, Y. Dai, R. Elbadry, V. Jani, X. Peng, L. Qian, G. Georgiev, D. Dimitrov, F. Zhang, J. Huang, J. Geng, Y. Chen, Y. Yuan, H. Wu, Y. Wang, I. Koychev, V. Stoyanov, M. Song, Y. Chen, X. Liu, P. Nakov, Overview of FinMMEval 2026: Multilingual and Multimodal Financial Evaluation, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction, Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Springer Lecture Notes in Computer Science, Jena, Germany, 2026.

[2] Z. Xie, Y. Dai, R. Elbadry, V. Jani, G. Georgiev, D. Dimitrov, F. Zhang, X. Peng, L. Qian, J. Huang, J. Geng, Y. Chen, Y. Yuan, H. Wu, Y. Wang, I. Koychev, V. Stoyanov, M. Song, Y. Chen, X. Liu, P. Nakov, Overview of the FinMMEval 2026 Task 1: Multilingual Financial Multiple-Choice Question Answering, in: CLEF 2026 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, Jena, Germany, 2026.

[3] Z. Xie, X. Peng, G. Georgiev, D. Dimitrov, R. Elbadry, F. Zhang, L. Qian, J. Huang, V. Jani, Y. Dai, J. Geng, Y. Chen, Y. Yuan, H. Wu, Y. Wang, I. Koychev, V. Stoyanov, M. Song, Y. Chen, X. Liu, P. Nakov, Overview of the FinMMEval 2026 Task 2: Financial Question Answering and Summarization, in: CLEF 2026 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, Jena, Germany, 2026.

[4] R. Elbadry, S. Ahmad, A. Heakl, D. Bouch, M. Ahsan, M. AlMahri, M. Elsaid khalil, Y. Wang, S. Lahlou, S. Ananiadou, V. Stoyanov, J. Huang, X. Peng, P. Nakov, Z. Xie, SAHM: A Benchmark for Arabic Financial and Shari'ah-Compliant Reasoning, arXiv preprint arXiv:2604.19098 (2026). doi:10.48550/arXiv.2604.19098.

[5] Y. Dai, Y. Lin, Z. Xie, Y. Wang, RealFin: How Well Do LLMs Reason About Finance When Users Leave Things Unsaid?, arXiv preprint arXiv:2602.07096 (2026). doi:10.48550/arXiv.2602.07096.

[6] V. Devane, M. Nauman, B. Patel, A. M. Wakchoure, Y. Sant, S. Pawar, V. Thakur, A. Godse, S. Patra, N. Maurya, S. Racha, N. K. Singh, A. Nagpal, P. Sawarkar, K. V. Pundalik, R. Saluja, G. Ramakrishnan, BhashaBench V1: A Comprehensive Benchmark for the Quadrant of Indic Domains, arXiv preprint arXiv:2510.25409 (2025).

[7] X. Peng, L. Qian, Y. Wang, R. Xiang, Y. He, Y. Ren, M. Jiang, V. J. Zhang, Y. Guo, J. Zhao, H. He, Y. Han, Y. Feng, Y. Jiang, Y. Cao, H. Li, Y. Yu, X. Wang, P. Gao, S. Lin, K. Wang, S. Yang, Y. Zhao, Z. Liu, P. Lu, J. Huang, S. Wang, T. Papadopoulos, P. Giannouris, E. Soufleri, N. Chen, Z. Deng, H. Fu, Y. Zhao, M. Lin, M. Qiu, K. E. Smith, A. Cohan, X.-Y. Liu, J. Huang, G. Xiong, A. Lopez-Lira, X. Chen, J. Tsujii, J.-Y. Nie, S. Ananiadou, Q. Xie, MultiFinBen: Benchmarking Large Language Models for Multilingual and Multimodal Financial Application, arXiv preprint arXiv:2506.14028 (2025). doi:10.48550/arXiv.2506.14028.

[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, et al., Language models are few-shot learners, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 1877–1901.

[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Advances in Neural Information Processing Systems, volume 35, 2022, pp. 24824–24837.

[10] Google DeepMind, Gemini: A family of highly capable multimodal models, Technical report, Google DeepMind, 2024. arXiv:2312.11805.

[11] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-consistency improves chain of thought reasoning in language models, in: International Conference on Learning Representations (ICLR), 2023.

[12] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, Association for Computational Linguistics, 2004, pp. 74–81.