CheckMate at CheckThat! 2026: Fusion-Based Retrieval, Reasoning-Trace Ranking, and Citation-Controlled Article Generation

CheckThat! at CLEF 2026

Taha Munawar[1,*], Abdullah Amir[1], Faisal Alvi[1] and Abdul Samad[1]

[1]Dhanani School of Science and Engineering, Habib University, Karachi, Pakistan

Abstract

This paper describes our participation as team *CheckMate* in the CheckThat! 2026 Lab. We address three tasks covering different parts of the automated fact-checking pipeline: source retrieval for scientific web claims, fact-checking numerical claims through reasoning-trace ranking, and full fact-checking article generation. For Task 1, we develop a multilingual retrieval and reranking pipeline based on Qwen3 dense retrieval, listwise reranking, and calibrated score fusion. For Task 2, we explore pointwise, pairwise, and preference-based ranking methods for selecting useful reasoning traces for numerical claim verification. For Task 3, we build a staged article-generation pipeline that filters noisy evidence, extracts claim-aware evidence snippets, constructs verdict-aware evidence packets, and generates citation-controlled fact-checking articles. On the official evaluation phase, our Task 1 system achieved 0.7194 average MRR@5, our Task 2 Arabic and Spanish submissions achieved average scores of 0.5599 and 0.4093 respectively, and our Task 3 system achieved a mean score of 0.328.

Keywords

Fact-Checking, Information Retrieval, Numerical Claim Verification, Reasoning-Trace Ranking, Fact-Checking Article Generation, CEUR-WS

1. Introduction

The rapid proliferation of misinformation across multilingual social media platforms necessitates the development of robust, automated systems capable of verifying complex claims at scale. While early fact-checking research focused primarily on simple, "flat" claim-to-document matching, the current consumer landscape requires the generation of transparent, long-form explanations. To address these challenges, the CLEF 2026 CheckThat! Lab introduces three interconnected tasks, all of which we attempt in this work [1].

Task 1 involves grounding informal discourse of scientific claims made by multilingual social media posts (in English, German, and French) that implicitly cite a "recent study" or a "university report" without providing any direct links. It requires formal evidence by verifying these claims from a candidate pool of scientific papers.

Task 2 shifts the focus from simple classification to test-time scaling and reasoning quality. Given a claim in English, Spanish, or Arabic, participants must rank multiple LLM-generated reasoning traces based on their utility and logical soundness. By introducing a re-ranking framework for reasoning paths, this task aims to mitigate "reasoning drift" in Large Language Models (LLMs), ensuring that the final verdict is not just correct but also derived from a valid rationalization of quantitative evidence.

Task 3 introduces a generative challenge that requires synthesizing a claim, its veracity, and a set of evidence documents into a cohesive, professional fact-checking article, moving beyond short-form justification toward full-scale automation of the journalistic process.

To facilitate reproducibility, our code, exact prompts, and hyperparameter configurations are publicly available at https://github.com/tahamunawar/LLM-CheckThat2026.

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

*Corresponding author.

tm08122@st.habib.edu.pk (T. Munawar); aa08453@st.habib.edu.pk (A. Amir); faisal.alvi@sse.habib.edu.pk (F. Alvi); abdul.samad@sse.habib.edu.pk (A. Samad)

2. Related Work

2.1. Task 1: Source Retrieval for Scientific Web Claims

Scientific claim-source retrieval requires matching informal social media claims to formal academic publications, creating both a stylistic and semantic gap between queries and evidence documents. Prior CheckThat! work on related claim-to-source tasks showed that multi-stage pipelines are effective for this setting, typically using an initial retrieval stage followed by neural reranking [2]. Earlier systems also demonstrated the value of combining lexical or dense retrieval with rerankers, using hard negatives to improve robustness against superficial keyword overlap [3, 4, 5, 6]. These findings directly motivate our Task 1 design: a multilingual dense retriever for candidate generation, listwise reranking over retrieved candidates, and calibrated fusion between complementary ranking signals. The multilingual setting also requires cross-lingual alignment between informal claims and scientific article representations.

2.2. Task 2: Fact-Checking Numerical Claims

Numerical claim verification is challenging because models must reason over quantities, dates, temporal scopes, and sometimes conflicting evidence. Prior work on numerical fact-checking benchmarks such as QuanTemp and QuanTemp++ highlights that quantitative claims require more than surface-level textual matching [7, 8]. In LLM-based verification, a further challenge is reasoning drift, where a model begins with a plausible reasoning path but reaches an unsupported verdict after misinterpreting numerical or temporal evidence [9]. Test-time scaling methods address this by generating multiple candidate reasoning traces, but this shifts the problem toward selecting the most reliable trace. This motivates verifier and reward-model approaches that evaluate reasoning quality rather than only the final answer [10, 11, 12]. Typically, these approaches utilize either Outcome-Reward Models (ORMs), which evaluate terminal correctness holistically [13], or Process-Reward Models (PRMs), which offer dense supervision by scoring intermediate steps to guide state-space search [14]. While these verifiers are traditionally trained on labor-intensive step-level human annotations [10], recent work explores automated advantage-modeling and synthetic training paradigms to improve scaling efficiency [11, 15]. Our Task 2 methods follow this direction by comparing pointwise scoring, pairwise ranking, order-debiased pairwise prompting, PREPAIR-style structured comparison, and DPO with Bradley-Terry reward modeling for ranking reasoning traces.

2.3. Task 3: Generating Full Fact-Checking Articles

Automated fact-checking has increasingly moved beyond verdict prediction toward generating explanations and full fact-checking articles. Earlier explanation-generation work argued that fact-checking systems should justify their predictions, not merely classify claims [16]. More recent article-generation frameworks such as QRAFT frame fact-checking article writing as a dedicated generation task that resembles the workflow of professional fact-checkers [17]. However, generated explanations and articles can suffer from weak evidence attribution, hallucinated support, and inaccurate citation use, making evidence selection and attribution central to trustworthy generation [18]. These concerns motivate our Task 3 pipeline, which separates source usability triage, claim-aware evidence extraction, verdict-aware evidence packet construction, and final article generation. Unlike direct generation from all retrieved sources, our approach restricts the generator to selected evidence identifiers and inserts citations deterministically, reducing the risk of invented or malformed source links.

3. Datasets

3.1. Task 1: Source Retrieval for Scientific Web Claims

Task 1 uses the official multilingual claim-source retrieval dataset, where English, French, and German social-media claims are mapped to gold scientific articles from a 10,000-document collection [19]. Table 1

summarizes the split sizes.

Table 1
Task 1 query dataset distribution.

Language	Train	Dev	Test
English	14,977	3,905	6,080
French	2,807	702	1,220
German	1,460	386	876
Total	19,244	4,993	8,176

3.2. Task 2: Fact-Checking Numerical Claims

Task 2 consists of numerical fact-checking claims in English, Spanish, and Arabic [20]. For each claim, the dataset provides 20 LLM-generated reasoning traces that must be ranked according to their usefulness for verification. The English split is based on QuanTemp, while the Spanish and Arabic splits are adapted from previous CheckThat! numerical claim verification data.

Table 2
Task 2 dataset statistics.

Language	Claims
English	8,000
Spanish	2,808
Arabic	3,260

3.3. Task 3: Generating Full Fact-Checking Articles

Task 3 uses the WatClaimCheck dataset for full fact-checking article generation [21]. Each example provides a claim, its veracity label, and a set of evidence documents. For our final run, we used the official test split, containing 1,158 claims and 8,979 total evidence source references, with an average of 7.75 sources per claim.

4. System Description

4.1. Task 1: Source Retrieval for Scientific Web Claims

Task 1 was approached as a multi-stage retrieval and reranking problem. Given a multilingual scientific web claim in English, French, or German, the goal was to retrieve the correct source or best-evidence publication from a collection of 10,000 scientific articles. Our final system evolved through several stages: an initial multilingual dense retriever, a stronger Qwen-based retriever, listwise neural reranking, and finally calibrated multi-model fusion.

4.1.1. Early Dense Retrieval Baseline

Our initial system used `intfloat/multilingual-e5-large` as a dense dual-encoder retriever. Scientific documents were represented using structured metadata strings containing title, abstract, venue, and author fields. We also experimented with chunked tokenization, where long document representations were split into overlapping 510-token chunks. To make training more robust, we mined hard negatives: incorrect papers with high lexical overlap with the claim. These hard negatives acted as "trick papers", forcing the model to distinguish true source relevance from superficial vocabulary overlap. The retriever was trained using Multiple Negatives Ranking Loss.

4.1.2. Retriever Replacement with Qwen3-Embedding-8B

To address the retriever bottleneck, we replaced the e5 baseline with Qwen3-Embedding-8B and fine-tuned it using LoRA. Queries were formatted using an instruction prompt describing the task, while documents were represented using title and abstract text. The model was trained using Cached Multiple Negatives Ranking Loss, with an effective batch size of 128.

This retriever substantially improved Stage 1 performance. The multilingual-e5-large baseline achieved 0.6282 MRR@5, while the Qwen3-Embedding-8B LoRA retriever reached approximately 0.6793 MRR@5. This confirmed that better candidate generation was essential for the rest of the pipeline.

4.1.3. Listwise Reranking over Qwen Candidates

After improving the retriever, we used the Qwen3-Embedding-8B LoRA model to generate top-10 candidate sets for each query. These candidates were then used to train listwise rerankers. Each training group contained one gold document and nine hard negatives retrieved by the Qwen retriever. The reranker was trained with a listwise softmax cross-entropy loss.

We tested multiple rerankers. A listwise `BAAI/bge-reranker-v2-m3` reranker improved performance to 0.7094 MRR@5 after fusion with dense retrieval scores. This first alpha-fusion experiment showed that the retriever and reranker were complementary, since pure retrieval and pure reranking were both weaker than their weighted combination. Therefore, later reranker experiments retained fusion as a core part of the pipeline. We then trained `Qwen/Qwen3-Reranker-8B` using LoRA. This model scored each query–document pair using a yes/no relevance prompt, where the final relevance score was computed as the difference between the "yes" and "no" logits. Due to high compute cost, the Qwen3-Reranker-8B model was trained for one epoch only, but it still achieved 0.7400 MRR@5 as a pure reranker and 0.7435 MRR@5 after fusion with the dense retriever.

We also tested `nvidia/Llama-nemotron-rerank-1b-v2`. Although Nemotron was weaker than Qwen3-Reranker-8B on its own, it provided complementary ranking behavior. The Nemotron reranker achieved 0.7179 MRR@5 as a pure reranker and 0.7284 MRR@5 after fusion with the Qwen dense retriever.

4.1.4. Final Multi-Model Fusion

The final system combines three complementary signals: the Qwen3-Embedding-8B dense retriever score, the Qwen3-Reranker-8B score, and the Nemotron reranker score. Since these models produce scores on different scales, direct score addition is not ideal. Prior retrieve-and-rerank work has used softmax-normalized retriever and reranker scores as comparable passage-level distributions, motivating our use of query-level score normalization before fusion [22]. We therefore used temperature-scaled softmax/log-probability fusion, where each model's scores are first normalized at the query level over the top-10 candidates:

$$s = w_d \log \mathrm{softmax}(s_d/T_d) + w_q \log \mathrm{softmax}(s_q/T_q) + w_n \log \mathrm{softmax}(s_n/T_n).$$

Here, s_d, s_q, and s_n denote the dense retriever, Qwen reranker, and Nemotron scores.

4.2. Task 2: Fact-Checking Numerical Claims

We explored two directions: parameter scaling and relational trace ranking. The relational methods compare traces directly using pairwise prompting, PREPAIR-style scorecards, and DPO with Bradley–Terry reward modeling.

4.2.1. Baseline Approach

The baseline utilizes a Llama 3.2 1B Instruct model in a point-wise ranking configuration. In this setup, the model processes a single reasoning trace in isolation for each claim.

4.2.2. Parameter Scaling

The first optimization phase explores the impact of model scale by swapping the 1B baseline for the more sophisticated Llama 3.2 3B Instruct model. The model is loaded using 4-bit quantization. This phase tests the hypothesis that the superior reasoning capabilities of the 3B model can overcome the logical errors observed in the 1B baseline, even when compressed, while keeping the same point-wise architecture intact.

4.2.3. Pairwise Ranking - Round Robin Tournament

The second approach implements a Pairwise Cross-Encoder [23] using the base Llama 3.2 1B model. By processing pairs of candidate traces simultaneously, the model leverages self-attention for direct relational reasoning. Traces compete in a Round-Robin tournament, accumulating points to yield a normalized win-score. During inference, these scores aggregate into a Best-of-N (BoN) selection to determine the final veracity verdict. Keeping the base model constant isolates the methodology's specific impact on ranking accuracy and class bias.

4.2.4. Pairwise Ranking - Elimination Tournament

To overcome compute limitations, an elimination bracket replaces the round-robin tournament. Traces are paired based on their BM25 seed scores to prevent top-tier, well-reasoned traces from eliminating each other early on. The final five remaining traces are designated as the top five by rank.

4.2.5. Pairwise Ranking Prompting

Adapted from Pairwise Ranking Prompting (PRP) [24], every pair is evaluated bidirectionally, both (A vs. B) and (B vs. A), to eliminate positional bias. Averaging the logit outputs from both directions yields a final win-margin, ensuring rankings reflect logical merit rather than document order.

4.2.6. PREPAIR

PREPAIR [25] attempts to mitigate context collapse by first extracting independent, unbiased insights from each trace to create a pointwise scorecard. This scorecard acts as a structured rationale behind the model's attention to specific logical flaws or numerical certainties. These are aggregated for final pairwise comparison, thus balancing absolute and relational logic, as well as mitigating the greater time complexity issue with naive pairwise ranking.

4.2.7. DPO & Bradley-Terry Reward Modeling

To bridge pairwise accuracy and pointwise speed, we converted pairwise trace preferences into preferred–rejected training examples. Instead of performing all pairwise comparisons at inference time, we used DPO-style preference optimization to train the model to assign higher scores to traces that were preferred during pairwise comparison.

The Bradley-Terry model maps a trace's latent quality to an unobserved reward value r. The probability that trace i is preferred over trace j ($i \succ j$) is defined as:

$$P(i \succ j) = \sigma(r_i - r_j) = \frac{1}{1 + e^{-(r_i - r_j)}} \tag{1}$$

The reward model is trained by minimizing the negative log-likelihood of these preferences over a dataset D:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim D} \left[\log \sigma \left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right] \tag{2}$$

where x is the context, and y_w and y_l are the preferred and rejected completions, respectively.

4.3. Task 3: Generating Full Fact-Checking Articles

The core rationale behind our staged architecture was to strictly constrain the generative model. Because end-to-end LLMs frequently hallucinate source URLs and conflate evidence, separating evidence extraction from generation allowed us to enforce deterministic, programmatically controlled citations.

For Task 3, we generated fact-checking articles using only the provided claim, veracity label, and evidence documents. The system did not retrieve new evidence or predict labels; instead, it selected usable evidence and generated cited articles consistent with the given verdict.

The pipeline used Vertex AI Gemini models for LLM-based stages: `gemini-2.5-flash` for Stage 1 usability triage, `gemini-3-flash-preview` for Stage 2 evidence extraction and Stage 4 generation, and Python-only procedures for Stage 0 preprocessing, Stage 3 packet construction, post-generation rescue, and citation validation.

4.3.1. Stage 0 and Stage 1: Source Flattening and Usability Triage

Stage 0 flattened each claim's evidence files into source-level records containing claim metadata, URL, file name, and source text. Stage 1 then performed usability triage, marking sources as processable only if the scraped text was readable and inspectable; it did not judge support or refutation.

4.3.2. Stage 2: Claim-Aware Evidence Extraction

Stage 2 processed only sources marked PROCESS. Using `gemini-3-flash-preview`, it extracted claim-aware fields such as `source_relation`, `evidence_role`, `stance_to_claim`, `claim_match_score`, `snippets`, `safe_source_fact`, and `risk_flags`, without using the final rating or reference article.

Deterministic guardrails prevented claim-origin pages, satire or fake-news origins, and weak background context from being promoted as direct proof.

4.3.3. Stage 3: Verdict-Aware Evidence Packet Builder

Stage 3 grouped Stage 2 outputs by claim and built compact verdict-aware packets using the original rating. Each packet contained `must_cite_evidence`, `optional_context`, `excluded_evidence`, risk flags, and a short verdict-logic summary, selected according to evidence role, stance, score, and risk.

4.3.4. Stage 4: Defensive Article Generation with Manual Citation Control

Stage 4 generated a structured sentence plan from the Stage 3 packet using `gemini-3-flash-preview`. A defensive filter selected up to six evidence sources while excluding weak context, duplicate claim-origin pages, no-content sources, and boilerplate. The model produced sentences with evidence identifiers rather than URLs; Python then inserted the exact citation URLs.

This prevented the model from inventing or modifying citation links. The system then validated missing citations, raw URLs, invalid evidence IDs, invented URLs, and citation mismatches; rows with validation errors or zero citations were handled through deterministic rescue and coverage-boosting.

5. Results and Discussion

5.1. Task 1: Source Retrieval for Scientific Web Claims

To evaluate the evolution of our Task 1 system, we first report development-set results for the major retrieval, reranking, and fusion stages. These results were used for model selection and ablation analysis.

Replacing `multilingual-e5-large` with LoRA-tuned Qwen3-Embedding-8B increased MRR@5 from 0.6282 to 0.6793, making candidate generation the largest early bottleneck.

We then trained task-specific listwise rerankers over the top-10 candidates retrieved by Qwen3-Embedding-8B. Figure 2 shows the first fusion experiment, where combining BGE-M3 reranker scores

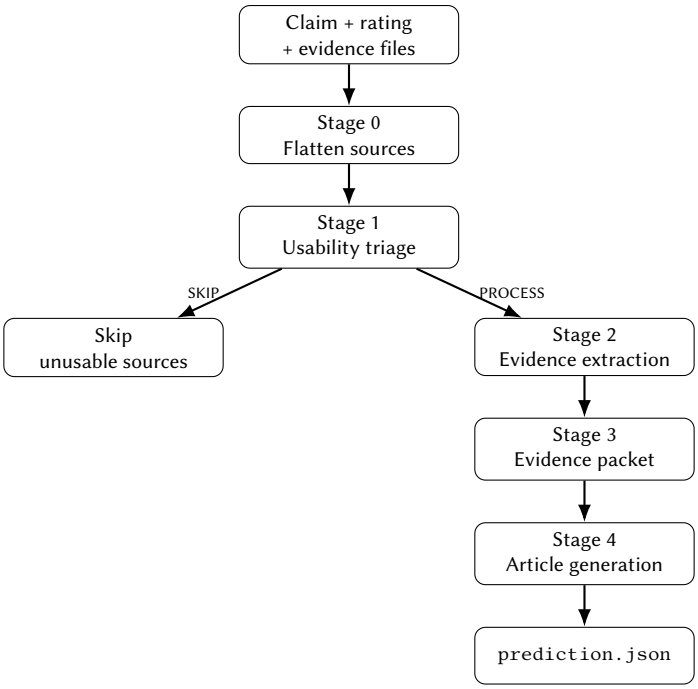

Figure 1: Task 3 staged fact-checking article-generation pipeline.

Table 3
Task 1 development-set performance across major system stages.

System	MRR@1	MRR@5	MRR@10	Recall@5	Recall@10
multilingual-e5-large retriever baseline	0.5548	0.6282	0.6350	0.7416	0.7917
Qwen3-Embedding-8B LoRA retriever	0.6147	0.6793	0.6876	0.7735	0.8348
Qwen3 retriever + BGE-reranker-v2-m3	0.6492	0.7094	0.7141	0.7996	0.8348
Qwen3 retriever + Nemotron reranker	0.6742	0.7284	0.7323	0.8060	0.8348
Qwen3 retriever + Qwen3-Reranker-8B	0.6956	0.7435	0.7466	0.8132	0.8348
Final three-model softmax fusion	**0.7146**	**0.7567**	**0.7594**	**0.8158**	0.8348

with Qwen3 retriever scores improved MRR@5 from 0.6764 for pure reranking and 0.6793 for pure retrieval to 0.7094 at $\alpha = 0.5$. This result justified retaining fusion in later reranker experiments. Nemotron reached 0.7284 MRR@5 after fusion, while Qwen3-Reranker-8B was the strongest individual reranker, reaching 0.7435 MRR@5 after one epoch of LoRA training.

The final system combined the Qwen3 dense retriever, Qwen3-Reranker-8B, and Nemotron reranker using temperature-scaled softmax/log-probability fusion. To minimize variance and reduce the development-to-test generalization gap, we empirically tuned the fusion weights and temperatures via grid search on the development set. We found that language-specific configurations yielded the best results. For English, the optimal weights were $w_d = 0.495$, $w_q = 0.380$, and $w_n = 0.125$, with temperatures $T_d = 0.25$, $T_q = 4.0$, and $T_n = 4.0$. For French, the weights were $w_d = 0.300$, $w_q = 0.360$, and $w_n = 0.340$ ($T_d = 0.25, T_q = 4.0, T_n = 4.0$). For German, the weights were $w_d = 0.475$, $w_q = 0.355$, and $w_n = 0.170$ ($T_d = 0.25, T_q = 4.0, T_n = 3.0$). This optimized configuration achieved our peak development score of 0.7567 MRR@5, indicating that the rerankers provided complementary relevance signals rather than merely duplicating the retriever ranking.

5.1.1. Ablation Findings

Several ablations did not improve performance. Query cleaning and LLM-based style transfer sometimes removed useful lexical cues. German paraphrasing and back-translation gave no stable gain despite

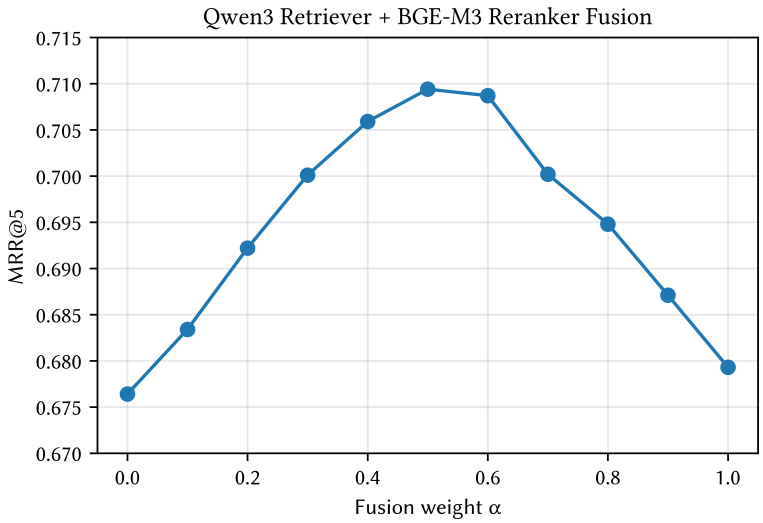

Figure 2: Development MRR@5 across dense/reranker fusion weights.

lower German data volume. Document chunking or hard-negative mining during the initial e5 retrieval stage was also not beneficial. Alternative retrievers, including BGE-M3, Harrier-OSS, and F2LLM-v2-14B, did not consistently outperform Qwen3-Embedding-8B, and zero-shot BGE reranking was weaker than task-trained reranking. Overall, the largest gains came from stronger candidate generation, listwise reranking on hard negatives, and calibrated score fusion.

We also submitted our final Task 1 system to Codabench under the team name *CheckMate*. Table 4 reports our Codabench development and official evaluation-phase results.

Table 4
Codabench Task 1 results for team *CheckMate*.

Phase	Avg. MRR@5	DE	EN	FR
Dev phase	0.7600	0.7000	0.7600	0.8000
Evaluation/Test phase	0.7194	0.6861	0.7343	0.7378

The Codabench development-phase result of 0.7600 average MRR@5 is consistent with our internal development result of 0.7567 MRR@5 after rounding. However, the official evaluation-phase score dropped to 0.7194 average MRR@5. This indicates that the final fusion strategy generalized reasonably but was still tuned heavily on the development split. The largest drop occurred in French, where the development score was 0.8000 but the evaluation score was 0.7378. German remained the most difficult language in both phases, with 0.7000 MRR@5 on the development phase and 0.6861 on the evaluation phase.

Table 5
Official Codabench evaluation-phase leaderboard excerpt for Task 1.

Team	Avg. MRR@5	DE	EN	FR
claim2source	0.7628	0.7153	0.7819	0.7912
team-outsider	0.7580	0.7228	0.7802	0.7709
mattiaskvist	0.7464	0.7070	0.7571	0.7752
CheckMate	0.7194	0.6861	0.7343	0.7378

Although our development score was competitive, the evaluation-phase leaderboard shows a larger generalization gap for our final system compared with the top submissions. This suggests that future

work should tune fusion parameters using more robust validation strategies, such as cross-validation or held-out language-specific calibration splits, rather than relying heavily on the public development set.

5.2. Task 2: Fact-Checking Numerical Claims

The evaluation metrics of interest for Task 2 are macro F1 and Recall@5. Recall@5 measures the quality of the ranked reasoning traces, while macro F1 measures final claim-verification performance.

Table 6 reports validation results across the baseline, parameter-scaling, pairwise-ranking, PRP, PREPAIR, and DPO+BT approaches.

In the English subset, the transition from the 1B baseline to the quantized 3B model yielded the most substantial performance gains. We observe a 15% relative increase in Macro-F1 and a modest improvement in ranking utility (Recall@5). Whereas a comparison between the Pointwise 1B baseline and the Pairwise 1B model reveals that the Pairwise model improved the Macro-F1 by 9% over the baseline. However, interestingly, the Pairwise model showed degradation in Recall@5 (0.3010 $\rightarrow$ 0.2757). This drop suggests that the tournament approach is superior at identifying the single best trace (improving F1), but the aggregated win-scores probably introduce slight noise in the lower ranks compared to absolute pointwise scoring.

Pairwise Ranking Prompting (PRP) successfully recovers some of the lost ranking utility in English, raising Recall@5 to 0.2835 and further boosting the Macro-F1 to 0.5170. This underscores the importance of debiasing techniques when using small-parameter models for relational reasoning.

The PREPAIR approach achieved the highest Macro-F1 among the 1B models in both English (0.5194) and Arabic (0.5297). By injecting scorecards derived from the pointwise model into the pairwise prompt, PREPAIR provides a cognitive anchor that helps the model avoid context collapse. However, this comes at the cost of lower sensitivity to the top-five ranking distribution, as reflected by the decline in English Recall@5 to 0.2695.

In contrast, the DPO + Bradley-Terry (BT) approach yielded mixed results. While it underperformed in English, it proved remarkably effective for the Spanish subset, achieving the highest Macro-F1 (0.4049) and Recall@5 (0.2574) for that language. This could indicate that the Bradley-Terry framework offers a stable, transitive ranking signal on smaller, more focused datasets where dense pairwise tournaments might lead to overfitting.

Lastly, the Arabic subset highlights a distinct performance gap. While PrePAIR boosts classification accuracy significantly (0.5297), no pairwise or advanced ranking approach is able to beat the simple 1B Pointwise baseline in Recall@5 (0.2786). A hypothesis for this is that Arabic has increased token density and structural complexity for pairwise comparison, which essentially doubles the prompt length and may exceed the reasoning density, degrading the output quality.

The table below shows the results obtained on the validation dataset:

Table 6
Comparative Performance Across Languages and Model Architectures.

Language	Metric	Baseline	Llama 3.2 3B	Pairwise	PRP	PREPAIR	DPO + BT
English	Macro-F1	0.4628	0.5324	0.5061	0.5170	**0.5194**	0.4936
	Recall@5	0.3010	0.3129	0.2757	0.2835	0.2695	0.2659
Arabic	Macro-F1	0.4917	—	0.5210	0.3555	**0.5297**	0.5172
	Recall@5	**0.2786**	—	0.2400	0.2349	0.2466	0.2485
Spanish	Macro-F1	0.3654	—	0.3804	0.3807	0.4003	**0.4049**
	Recall@5	0.2401	—	0.2425	0.2422	0.2090	**0.2574**

English is reported as an internal result, whereas Spanish and Arabic have official submissions.

Due to last-minute submission issues, only the Arabic and Spanish runs were officially submitted on Codabench; the English result is reported as an internal project result.

Table 7
Task 2 results. Arabic and Spanish are official Codabench test results; English is reported as an internal result.

Language	Rank	Avg Score	Macro-F1	Recall@5
Spanish	9th	0.4093	0.5662	0.2525
Arabic	11th	0.5599	0.8407	0.2792
English (internal)	N/A	0.4115	0.2320	0.5909

5.3. Task 3: Generating Full Fact-Checking Articles

Table 8 reports the official Task 3 results for our final submission. The system achieved an overall mean score of 0.328, with stronger citation precision, citation recall, and evidence coverage than entailment. This suggests that deterministic citation control helped maintain valid source use, but the generated sentences were not always tightly supported by the cited evidence.

Table 8
Official Task 3 Codabench scores for team *CheckMate*.

Metric	Score
Mean entailment score	0.221
Mean citation precision	0.350
Mean citation recall	0.350
Mean evidence coverage	0.391
Overall mean score	0.328

Internal diagnostics suggest that the pipeline applied conservative source-use decisions before article generation. Of 8,979 flattened source-level records, 5,741 were passed forward as PROCESS, while 3,238 were filtered out at the usability-triage stage. After repaired Stage 2 extraction, sources were distributed across `useful_context` (2,251), `direct_evidence` (2,103), `off_topic` (986), `claim_origin` (939), and `no_content` (282). Rather than treating all evidence uniformly, the system attempted to distinguish direct verification evidence from contextual or claim-origin material. This made the generation process safer, but may also have limited the amount of evidence available for broader coverage and stronger sentence-level entailment.

Stage 4 validation reflects the same precision-oriented design. The initial article set contained 96 validation-error rows and 156 zero-citation rows. A deterministic post-generation rescue pass reduced validation errors to zero and reduced zero-citation rows to 33, while preserving the required output structure. However, the official scores show that structural validity and citation control did not fully translate into strong entailment performance. This suggests that future versions should relax evidence-use decisions more carefully, while adding sentence-level entailment checks to ensure that generated claims remain closely supported by their citations.

6. Conclusion

This paper presented our participation as team *CheckMate* in the CheckThat! 2026 Lab across three tasks.

For Task 1, the strongest gains came from replacing the initial dense retriever with Qwen3-Embedding-8B, adding task-trained reranking, and applying calibrated multi-model fusion; the final system achieved 0.7194 average MRR@5 on the official evaluation phase.

For Task 2, we found that both model scale and ranking methodology affected numerical claim verification. Moving from the 1B pointwise baseline to a quantized 3B model produced the strongest English-language gain. Pairwise methods improved final verdict prediction in several settings, but sometimes reduced Recall@5, suggesting that identifying the single best trace and preserving a strong

top-five ranking are not always aligned. PREPAIR performed strongly in English and Arabic by adding structured scorecard information before comparison, while DPO with Bradley-Terry reward modeling was most effective for Spanish. Future work could explore reinforcement-learning-based ranking strategies such as Iterative Exclusion Ranker, where the model learns to eliminate weaker candidates step by step rather than scoring all traces independently [26]. With greater computational resources, full round-robin pairwise tournaments could also be revisited, since they may provide richer comparison signals than the more efficient elimination-based setup used in our experiments.

For Task 3, we built a staged LLM-assisted article-generation pipeline with conservative evidence selection, deterministic citation insertion, and post-generation validation. The final submission achieved an overall mean score of 0.328, with zero validation-error rows after rescue. However, the lower entailment score shows that valid citation formatting alone is insufficient; future work should better balance citation control, evidence coverage, and sentence-level support.

Across all three tasks, our results suggest that decomposed fact-checking pipelines are useful because retrieval, reasoning-trace ranking, evidence selection, and article generation introduce different failure modes. Future work should focus on stronger validation strategies, multilingual calibration, and tighter integration between citation-aware generation and entailment-based checking.

Declaration on Generative AI

During the preparation of this work, the authors used Generative AI tools and services, including Gemini and NotebookLM, in order to: Generate literature review support from author-selected sources, Drafting content assistance, Paraphrase and reword, Improve writing style, and Grammar and spelling check. After using these tools/services, the authors reviewed, edited, and verified the content as needed and take full responsibility for the publication's content. The scientific contributions, system design, experiments, results, analysis, and conclusions were produced and validated by the human authors.

Furthermore, separate from the writing process, Vertex AI Gemini models were utilized as core functional components within the implemented Task 3 system pipeline for source usability triage, claim-aware evidence extraction, and fact-checking article generation. These model uses are part of the described experimental system rather than authorship or manuscript preparation.

References

[1] J. M. Struß, S. Schellhammer, S. Dietze, V. V., V. Setty, T. Chakraborty, P. Nakov, A. Anand, P. Chungkham, S. Hafid, D. Sahnan, K. Todorov, The clef-2026 checkthat! lab: Advancing multilingual fact-checking, in: R. Campos, A. Jatowt, Y. Lan, M. Aliannejadi, C. Bauer, S. MacAvaney, A. Anand, Z. Ren, S. Verberne, N. Bai, M. Mansoury (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2026, pp. 325–335.

[2] S. Hafid, Y. S. Kartal, S. Schellhammer, K. Boland, D. Dimitrov, S. Bringay, K. Todorov, S. Dietze, Overview of the CLEF-2025 CheckThat! lab task 4 on scientific web discourse, 2025. URL: https://ceur-ws.org/Vol-4038/paper_54.pdf.

[3] M. Staudinger, A. El-Ebshihy, W. Kusa, F. Piroi, A. Hanbury, ATOM at CheckThat! 2025: Retrieve the implicit - scientific evidence retrieval, 2025. URL: https://ceur-ws.org/Vol-4038/paper_98.pdf.

[4] P. J. Sager, A. Kamaraj, B. F. Grewe, T. Stadelmann, Deep retrieval at CheckThat! 2025: Identifying scientific papers from implicit social media mentions via hybrid retrieval and re-ranking, 2025. URL: https://ceur-ws.org/Vol-4038/paper_89.pdf.

[5] P. M. Thapliyal, R. S. Chavan, S. Samridh, C. Zuo, R. Banerjee, SCIRE at CheckThat! 2025: Bridging social media, scientific discourse, and scientific literature, 2025. URL: https://ceur-ws.org/Vol-4038/paper_99.pdf.

[6] C. Ashbaugh, L. Baumgärtner, T. Greß, N. Sidorov, D. Werner, AIRwaves at CheckThat! 2025: Retrieving scientific sources for implicit claims on social media with dual encoders and neural re-ranking, 2025. URL: https://ceur-ws.org/Vol-4038/paper_63.pdf.

[7] V. V, A. Anand, A. Anand, V. Setty, Quantemp: A real-world open-domain benchmark for fact-checking numerical claims, 2024. URL: https://arxiv.org/abs/2403.17169.
[8] V. Venktesh, D. Prabhu, A. Anand, A benchmark for open-domain numerical fact-checking enhanced by claim decomposition, 2025. URL: https://arxiv.org/abs/2510.22055.
[9] P. Chungkham, V. Venktesh, V. Setty, A. Anand, Think right, not more: Test-time scaling for numerical claim verification, 2025. URL: https://arxiv.org/abs/2509.22101.
[10] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, K. Cobbe, Let's verify step by step, 2023. URL: https://arxiv.org/abs/2305.20050.
[11] A. Setlur, C. Nagpal, A. Fisch, X. Geng, J. Eisenstein, R. Agarwal, A. Agarwal, J. Berant, A. Kumar, Rewarding progress: Scaling automated process verifiers for llm reasoning, 2024. URL: https://arxiv.org/abs/2410.08146.
[12] J. Zhao, R. Liu, K. Zhang, Z. Zhou, J. Gao, D. Li, J. Lyu, Z. Qian, B. Qi, X. Li, B. Zhou, Genprm: Scaling test-time compute of process reward models via generative reasoning, 2025. URL: https://arxiv.org/abs/2504.00891. arXiv:2504.00891.
[13] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, J. Schulman, Training verifiers to solve math word problems, 2021. URL: https://arxiv.org/abs/2110.14168. arXiv:2110.14168.
[14] J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving, I. Higgins, Solving math word problems with process- and outcome-based feedback, 2022. URL: https://arxiv.org/abs/2211.14275. arXiv:2211.14275.
[15] W. Li, Y. Li, Process reward model with q-value rankings, 2025. URL: https://arxiv.org/abs/2410.11287. arXiv:2410.11287.
[16] P. Atanasova, J. G. Simonsen, C. Lioma, I. Augenstein, Generating fact checking explanations, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2020, pp. 7352–7364. URL: https://aclanthology.org/2020.acl-main.656/.
[17] D. Sahnan, D. Corney, I. Larraz, G. Zagni, R. Miguez, Z. Xie, I. Gurevych, E. Churchill, T. Chakraborty, P. Nakov, Can LLMs automate fact-checking article writing?, Transactions of the Association for Computational Linguistics 14 (2026) 489–509. URL: https://aclanthology.org/2026.tacl-1.23/. doi:10.1162/TACL.a.644.
[18] R. Xing, T. Baldwin, J. H. Lau, Evaluating evidence attribution in generated fact checking explanations, in: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, 2025, pp. 5475–5496. URL: https://aclanthology.org/2025.naacl-long.282/.
[19] S. Schellhammer, S. Hafid, Y. S. Kartal, K. Boland, D. Dimitrov, K. Todorov, S. Dietze, Overview of the CLEF-2026 CheckThat! lab task 1 on source retrieval for scientific web claims, in: [27], 2026.
[20] V. V, V. Setty, P. Chungkham, A. Anand, F. Alam, Overview of the CLEF-2026 CheckThat! lab task 2 on fact-checking numerical claims, in: [27], 2026.
[21] D. Sahnan, T. Chakraborty, P. Nakov, Overview of the CLEF-2026 CheckThat! lab task 3 on generating full fact-checking articles, in: [27], 2026.
[22] M. Glass, G. Rossiello, M. F. M. Chowdhury, A. Naik, P. Cai, A. Gliozzo, Re2G: Retrieve, rerank, generate, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Seattle, United States, 2022, pp. 2701–2715. URL: https://aclanthology.org/2022.naacl-main.194/. doi:10.18653/v1/2022.naacl-main.194.
[23] J. Lee, J. Hockenmaier, Evaluating step-by-step reasoning traces: A survey, 2025. URL: https://aclanthology.org/2025.findings-emnlp.94.pdf.
[24] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, L. Yan, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, M. Bendersky, Large language models are effective text rankers with pairwise ranking prompting, 2024. URL: https://arxiv.org/abs/2306.17563.
[25] H. Jeong, C. Park, J. Hong, H. Lee, J. Choo, The comparative trap: Pairwise comparisons amplifies biased preferences of LLM evaluators, in: Y. Belinkov, A. Mueller, N. Kim, H. Mo-

hebbi, H. Chen, D. Arad, G. Sarti (Eds.), Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, Association for Computational Linguistics, Suzhou, China, 2025, pp. 79–108. URL: https://aclanthology.org/2025.blackboxnlp-1.5/. doi:10.18653/v1/2025.blackboxnlp-1.5.

[26] T. Feng, Z. Hua, Z. Lei, Y. Xie, S. Yang, B. Long, J. You, R1-ranker: Teaching llm rankers to reason, arXiv preprint arXiv:2506.21638 (2025). URL: https://arxiv.org/abs/2506.21638.

[27] E. S. Salido, A. Barrón-Cedeño, Alba García Seco de Herrera, S. MacAvaney, J. M. Struß (Eds.), CLEF 2026 Working Notes, CLEF 2026, Jena, Germany, 2026.