Outsider at CheckThat! 2026: Retrieval, Verification, and Generation for Multilingual Fact-Checking

Notebook for the CheckThat! Lab at CLEF 2026

Quy Thanh Le[1,*], Ismail Badache[1] and Maamar El Amine Hamri[1]

[1]Aix Marseille Université, CNRS, LIS, Marseille, France

Abstract

This notebook describes the Outsider team's participation in the CheckThat! Lab at CLEF 2026. The three tasks cover different parts of a fact-checking workflow: source retrieval for scientific web claims, verification of numerical and temporal claims, and full article generation with inline citations. We handled them as three separate engineering problems. Task 1 received the heaviest retrieval and reranking stack. Task 2 was kept deliberately compact. Task 3 put most of the control effort before generation, at the evidence and citation stage.

The models are all open-source foundation models, but they play different roles. In Task 1, microsoft/harrier-oss-v1-27b retrieves five candidate papers and Qwen3 rerankers reorder them. In Task 2, Mistral-Small-3.1-24B-Base-2503 scores claim–verdict pairs built from the verdict conclusions attached to the released reasoning traces. In Task 3, Llama-3.2-1B prepares short citation sentences and selects usable evidence documents; Mistral-Small-4-119B-2603-NVFP4 then writes the article body from those filtered evidence documents.

The official results were competitive. Our Task 1 run ranked second overall with an MRR@5 of 0.758, including first place for German with 0.7228, second place for English with 0.7802, and third place for French with 0.7709. Task 2 ranked fourth for English with an average score of 0.4217, second for Arabic with 0.6004, and sixth for Spanish with 0.4451. Task 3 ranked first with a mean score of 0.546. The main lesson from these runs is practical: heavy retrieval helped most in Task 1, the cheap verifier was acceptable but uneven in Task 2, and citation filtering carried much of the Task 3 result.

Keywords

Automated Fact-Checking, Source Retrieval, Claim Verification, Article Generation, Large Language Models, Multilingual Datasets, Foundation Models

1. Introduction

Online misinformation is not a single format. The CheckThat! 2026 tasks require participating systems to handle short social media posts about scientific papers, claims whose truth depends on numbers or dates, and full fact-checking articles that need citations. These cases share the same broad fact-checking goal, but they stress very different parts of a system. A post about a paper mostly tests retrieval. A numerical claim tests whether the verifier can choose among plausible verdicts. A generated article tests whether the explanation remains tied to evidence. Professional fact-checking already has to manage these steps under time pressure [1]; automated support is useful only if it helps with the specific bottleneck at hand [2, 3]. Early end-to-end systems such as ClaimBuster demonstrated the feasibility of computational support for fact-checking workflows [4], while more recent research has broadened the scope toward multilingual, cross-lingual, and open-domain verification settings [5, 6].

Automated fact-checking is commonly modeled as a pipeline composed of several interdependent stages, including claim detection, evidence retrieval, claim verification, and explanation generation [7, 2]. Large-scale benchmarks such as FEVER, FEVEROUS, MultiFC, and AVeriTeC have helped formalize evidence-based claim verification under different evidence assumptions, ranging from encyclopedic sources to heterogeneous web evidence [8, 9, 10, 11]. However, realistic fact-checking remains difficult because claims can be underspecified, context-dependent, multilingual, or supported and contradicted by evidence scattered across different sources. This is even more challenging when claims contain

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

*Corresponding author.

quy-thanh.le@lis-lab.fr (Q. T. Le); ismail.badache@lis-lab.fr (I. Badache); amine.hamri@lis-lab.fr (M. E. A. Hamri)

quantities or dates, since numerical and temporal information often requires precise comparison, contextualization, and reasoning [12, 13, 14].

The CheckThat! Lab has been organized over several editions to encourage research on tasks that support the verification pipeline used by journalists and fact-checkers [15]. The CLEF 2026 edition returns to core stages of the pipeline and introduces three complementary tasks across five languages [16, 17]. Task 1, Source Retrieval for Scientific Web Claims, focuses on social media posts that contain scientific claims and implicitly refer to scientific papers without necessarily providing explicit URLs or formal citations [18]. Informal scientific web discourse often mentions studies through partial titles, institutions, authors, or vague descriptions, not precise bibliographic references [19]. Task 2, Fact-Checking Numerical and Temporal Claims, addresses the verification of claims involving quantities and temporal expressions in English, Spanish, and Arabic [20]. In addition to predicting a final verdict, the task provides reasoning traces and asks systems to identify traces that are useful for reaching the correct decision. Task 3, Generating Full-Fact-Checking Articles, moves beyond short verdicts and explanations by requiring a full article with inline citations, given a claim, its veracity label, and evidence documents [21]. The task reflects professional fact-checking as an explanatory writing practice, where evidence must be selected, organized, and presented clearly to readers [22, 23, 24].

In this paper, we present the participation of the Outsider team in all three CheckThat! 2026 tasks. The submission was built around a simple working rule: spend computation where the task clearly needs it, and keep the rest of the pipeline as small as possible. Task 1 therefore uses a strong dense retriever followed by reranking. Task 2 is intentionally sparse, keeping only the verdict conclusions attached to the released traces. Task 3 separates citation construction from article writing, because we found citation grounding easier to control at the sentence level than inside one long generation prompt.

The contribution is not a single universal architecture. It is a set of task-specific choices with visible trade-offs: strong but expensive retrieval, cheap but incomplete verdict scoring, and article generation helped by a conservative citation filter. The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the datasets, models, and methods. Section 4 discusses the experimental results. Section 5 summarizes the main findings and future work.

2. Related Work

Automated fact-checking is often studied as a modular pipeline that includes identifying check-worthy claims, retrieving evidence, predicting veracity, and producing explanations for end users [7, 2, 3]. Early benchmarks such as FEVER formalized evidence-based verification with claims and supporting or refuting evidence from Wikipedia [8], while later datasets extended the setting to multiple domains, structured evidence, and real-world web sources [10, 9, 11]. The CheckThat! Lab follows this line of work by defining shared tasks that target different stages of the verification workflow in multilingual and practical settings [15, 17].

The first task is closely related to evidence retrieval and scientific claim verification. SciFact introduced a benchmark where systems must retrieve scientific abstracts and identify evidence rationales for expert-written scientific claims [25], and SciFact-Open later moved to a much larger open-domain corpus of scientific abstracts [26]. CheckThat! 2026 Task 1 is less controlled in the form of the query: the input is an informal social media post that may refer to a scientific paper through a paraphrase, partial description, or contextual clue [18]. In practice, this makes the task closer to noisy bibliographic source linking. Prior work on scientific document representations and dense retrieval remains relevant. Models such as SPECTER learn document-level representations for scientific papers using citation-informed pretraining [27], while Sentence-BERT and DPR show how dense embeddings can support efficient semantic retrieval beyond exact lexical matching [28, 29]. These results motivated our retrieve-then-rerank setup.

The second task relates to claim verification with numerical and temporal information. Numerical claims are challenging because systems must compare quantities, units, dates, rates, and contextual conditions, not just match entities or topical evidence. Recent work has introduced benchmarks and methods for this problem, including QuanTemp for real-world numerical claim verification [13] and

the CheckThat! 2025 numerical claim task [30]. Systems from the 2025 task also explored heavier open-source LLM pipelines, such as combining question generation, evidence retrieval, and veracity prediction for numerical claims [31]. CheckThat! 2026 Task 2 continues this line with numerical claims and released reasoning traces in a multilingual setting [20]. Numerical reasoning methods such as NT5 and ELASTIC show that language models often benefit from explicit training or symbolic components when handling arithmetic and quantitative comparison [32, 33]. Other work has explored program-guided reasoning for complex fact-checking [34], while recent studies point to temporal robustness issues in LLMs [14]. Our Task 2 run sits at the lightweight end of this line of work: it tests the value of verdict conclusions without modeling the full trace.

The third task connects fact-checking with explanation and long-form article generation. Professional fact-checking is not limited to assigning a veracity label: it also involves selecting evidence, organizing it into a readable argument, and communicating uncertainty or context to readers [22]. Earlier work on generating fact-checking explanations and briefs showed that automatic systems can produce supporting summaries for verification decisions, but that evidence selection and faithfulness remain central challenges [35, 36]. Retrieval-augmented generation further provides a general framework for grounding generated text in external evidence [37], which is particularly important when generated outputs must include citations. More recent work on fact-checking article writing has moved toward complete article generation workflows and found that LLM-based systems can support this process, although expert-written articles remain difficult to match [24]. CheckThat! 2026 Task 3 follows this direction, but the inline-citation requirement also creates a separate evidence-control problem: the generated article must read coherently while individual cited statements remain tied to the provided documents [21].

3. Retrieval, Verification, and Generation in the Fact-Checking Process

This section describes the datasets, models, and methods used for the three CheckThat! 2026 tasks. Because the tasks have different inputs, outputs, and evaluation settings, we present the methodology separately for source retrieval, verdict-only verification, and full article generation.

We did not force the same model interface on all three tasks. Retrieval needed fast corpus search plus slower pairwise scoring. Verification allowed a compact claim–verdict formulation. Article generation required more bookkeeping around evidence documents, because the output is judged both as text and as a cited explanation.

3.1. Dataset

The CheckThat! Lab at CLEF 2026 provides three datasets corresponding to three different fact-checking settings: retrieving scientific sources from social media posts, verifying numerical and temporal claims, and generating full fact-checking articles [17, 18, 20, 21]. Because these tasks differ in their input format, output requirements, language coverage, and evaluation metrics, we describe each dataset separately in the following subsections.

3.1.1. Task 1 Dataset

Task 1 builds on the scientific web discourse setting, where social media posts contain scientific claims and implicitly refer to research papers without necessarily including explicit links or formal citations [19, 18]. According to the task overview, the English training data reuses existing post–paper pairs from earlier scientific web discourse data, while new evaluation data are provided for English and newly annotated multilingual data are provided for German and French [18]. Table 1 reports the number of social media posts in each split for the three languages, together with the 10,000 corpus publications in the candidate paper collection.

The collection data contains paper identifiers, titles, abstracts, and bibliographic metadata such as authors, journal, publication time, and DOI. Our retrieval pipeline keeps only `cord_uid`, title, and

abstract. The identifier is `cord_uid`; the document text is the title concatenated with the abstract.

Table 1
Dataset statistics for Task 1: Source Retrieval for Scientific Web Claims. Language rows count social media posts, and collection data counts corpus publications.

Language / file	Train	Dev	Test	Total
German	1,460	386	876	2,722
English	14,977	3,905	6,076	24,958
French	2,807	702	1,220	4,729
Collection data	–	–	–	10,000

3.1.2. Task 2 Dataset

Task 2 focuses on naturally occurring claims that contain numerical quantities or temporal expressions. The English data comes from the QuanTemp numerical claim verification benchmark, while the Arabic and Spanish claims are taken from the CLEF 2025 CheckThat! task with corresponding evidence [13, 30, 20]. In the released task data, each claim is accompanied by evidence documents and a set of LLM-generated reasoning traces, each associated with a verdict conclusion. For training and development instances, the gold claim verdict is also provided; the public test files contain the reasoning traces and verdict conclusions but not the gold verdict. Spanish and Arabic contain 20 reasoning traces per claim across all splits. English contains 20 traces for the test split and all validation instances contain 15 traces; in the training split, 6,204 instances contain 20 traces and 196 instances contain 15 traces. Table 2 reports the statistics of the local dataset used in our experiments.

Table 2
Dataset statistics for Task 2: Fact-Checking Numerical and Temporal Claims.

Language	Train	Dev	Test	Total claims	Traces/claim
English	6,400	1,600	2,558	10,558	15/20
Spanish	2,246	562	1,164	3,972	20
Arabic	2,608	652	511	3,771	20

3.1.3. Task 3 Dataset

Task 3 addresses English full fact-checking article generation [21]. For training and development, the data comes from WatClaimCheck; each instance includes claim metadata, a normalized veracity rating, evidence article references, and a reference fact-checking article. The public test set combines examples from ExClaim, introduced by Zeng and Gao [38], and AmbiguousSnopes. In Table 3, evidence references denote the total number of evidence documents listed across all claims in each split.

Table 3
Dataset statistics for Task 3: Generating Full Fact-Checking Articles.

Source	Split	Claims	Evidence refs.
WatClaimCheck	Train	26,976	206,162
WatClaimCheck	Dev	3,372	26,083
ExClaim / AmbiguousSnopes	Test	1,158	8,979

3.2. Foundation Models Used in Experiments

Table 4 summarizes the open-source foundation models used in the experiments. As above, "foundation models" covers embedding models, rerankers, and generative LLMs.

Table 4
Foundation models used in our CheckThat! 2026 system.

Model	Size	Type	Role in our system
Harrier-OSS-v1	27B	Multilingual embedding model	First-stage source retrieval for Task 1
Linq-Embed-Mistral	7B	Embedding model	Additional first-stage retrieval model for Task 1
Qwen3-VL-Reranker	8B	Reranker	Second-stage reranking for Task 1
Qwen3-Reranker	8B / 4B / 0.6B	Reranker	Additional second-stage reranking models for Task 1
Mistral-Small 3.1 Base	24B	Base language model	Verdict-only verification for Task 2
Llama 3.2	1B	Generative LLM	Evidence filtering and citation preparation for Task 3
Mistral-Small 4 NVFP4	119B	Quantized LLM	Full fact-checking article generation for Task 3

Harrier-OSS-v1-27B[1] is a large multilingual embedding model developed by Microsoft. It is designed to encode texts from different languages and domains into dense vector representations, making it suitable for semantic search and retrieval settings where lexical overlap is limited.

Linq-Embed-Mistral[2] is an embedding model based on the Mistral architecture. It serves as an additional dense retrieval backbone in the Task 1 first-stage experiments.

Qwen rerankers include Qwen3-VL-Reranker-8B[3] and Qwen3-Reranker models at 8B, 4B, and 0.6B scales[4][5][6]. Unlike embedding models that independently encode queries and documents, rerankers directly score query–candidate pairs and can therefore provide finer-grained relevance judgments over a smaller candidate set.

Mistral-Small-3.1-24B-Base-2503[7] is a 24B-parameter base language model from Mistral AI. As a base model, it provides a strong general-purpose language modeling backbone that can be adapted to downstream scoring or verification tasks without relying on a heavily specialized architecture.

Llama-3.2-1B[8] is a compact open-weight language model from the Llama family. Its small size makes it practical for lightweight text processing steps that need to be repeated many times, while still retaining the instruction-following and generation capabilities of modern LLMs.

Mistral-Small-4-119B-2603-NVFP4[9] is a large quantized model from the Mistral Small 4 series. The NVFP4 format reduces the memory and inference cost of the 119B-parameter checkpoint, making it more practical for long-form generation while preserving the capacity of a large model.

3.3. Methodology

3.3.1. Task 1: Source Retrieval for Scientific Web Claims

Task 1 aims to retrieve the scientific paper implicitly mentioned in a social media post [18]. Given a post and a fixed paper collection, our pipeline first retrieves a small candidate set with a bi-encoder and then reranks it with a cross-encoder. The same pipeline is applied to English, German, and French. Figure 1 summarizes the workflow.

[1] https://huggingface.co/microsoft/harrier-oss-v1-27b
[2] https://huggingface.co/Linq-AI-Research/Linq-Embed-Mistral
[3] https://huggingface.co/Qwen/Qwen3-VL-Reranker-8B
[4] https://huggingface.co/Qwen/Qwen3-Reranker-8B
[5] https://huggingface.co/Qwen/Qwen3-Reranker-4B
[6] https://huggingface.co/Qwen/Qwen3-Reranker-0.6B
[7] https://huggingface.co/mistralai/Mistral-Small-3.1-24B-Base-2503
[8] https://huggingface.co/meta-llama/Llama-3.2-1B
[9] https://huggingface.co/mistralai/Mistral-Small-4-119B-2603-NVFP4

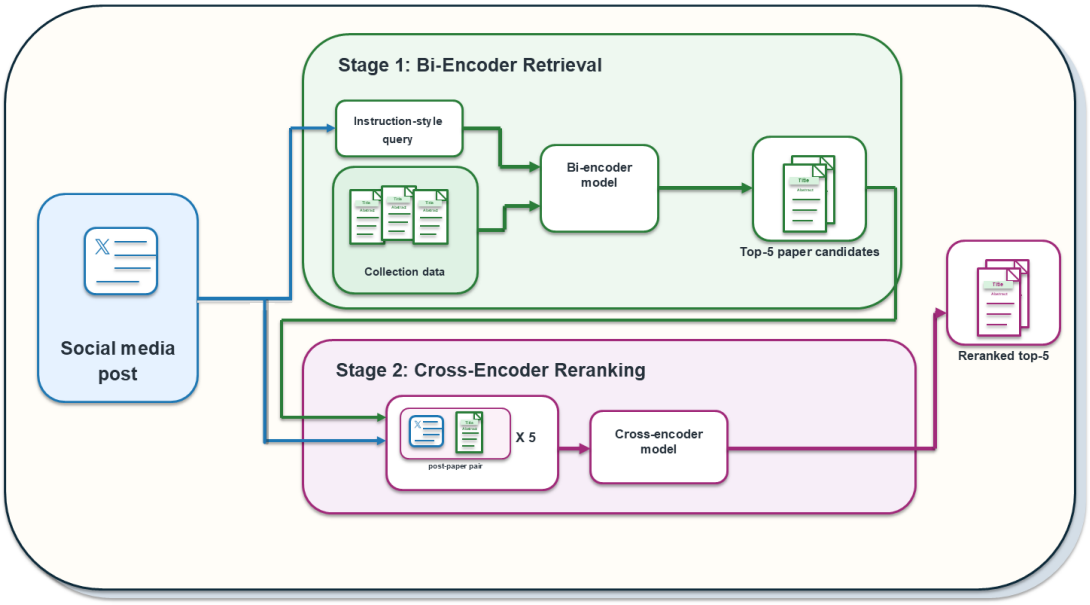

Figure 1: Two-stage source retrieval workflow.

3.3.1.1. Stage 1: Bi-Encoder Retrieval

Overview The first stage is a dense retriever. Besides Harrier-OSS-v1-27B, we also run Linq-Embed-Mistral as a second embedding backbone. Each paper is represented by its title and abstract, encoded once, and stored in the corpus index. A social media post is then encoded as an instruction-style query and compared with the paper embeddings. The retriever returns the top five candidate `cord_uids` and their similarity scores.

The query and paper text are formatted separately. The query format is:

```
Instruct: Given a question, retrieve passages that answer the question
Query: <post>
```

The paper format is:

```
Title: <title>
Abstract: <abstract>
```

The `Title` and `Abstract` fields are not encoded separately; they are joined into a single paper text before being passed to the bi-encoder.

The query and document are encoded independently. We L2-normalize the embeddings and rank all papers by vector similarity.

Training procedure During training, the candidate set is refreshed after each epoch. Before the first epoch, the base Harrier checkpoint retrieves the top 100 papers for every training query. From this refreshed pool, only the top five papers are used to build training pairs. A query–paper pair is labeled as positive when the candidate paper has the same `cord_uid` as the gold paper; all other top-five candidates are treated as hard negatives. If the gold paper is not present in the top five for a query, that query contributes only hard-negative pairs for that epoch. After each epoch, the updated checkpoint retrieves a new top-100 pool, allowing the next epoch to train on harder candidates mined by a stronger version of the model.

The bi-encoder is optimized with the SentenceTransformers pairwise `ContrastiveLoss`, using cosine distance and a margin of 0.5. At test time, only the top five papers are passed to the reranker, which keeps the second stage small.

3.3.1.2. Stage 2: Cross-Encoder Reranking

Overview The second stage reranks the five candidate papers returned by Stage 3.3.1.1. We first remove the Stage 1 instruction prefix from the stored query text and keep only the raw social media post. The reranker then scores each post–paper pair, where the paper is represented by the same title-plus-abstract text used in the first stage. The inference prompt is:

```
<Instruct>: Given a web search query, retrieve relevant passages that answer the
query
<Query>: <post>
<Document>: <title> <abstract>
```

The prompt asks the model to judge whether the document satisfies the query and restricts the answer to "yes" or "no". For each pair, we compute the relevance score as the difference between the next-token logits of `yes` and `no`. The five candidates are then sorted by this score in descending order, producing the final Task 1 ranking.

Training procedure The second-stage reranker is trained with a listwise objective over top-five candidate groups. Each group contains the post, one positive paper when the gold `cord_uid` is present, and the remaining candidates as negatives.

Training instances are kept only when the gold paper appears in the top-five candidate pool. In our run, the dataset builder processed 12,853 query groups, wrote 11,685 listwise training rows, and skipped 1,168 groups where the gold paper was missing. The reranker therefore learns to reorder the retriever's candidates; it cannot recover papers that never enter the shortlist.

3.3.2. Task 2: Verdict-Based Verification of Numerical and Temporal Claims

Task 2 focuses on verifying claims involving numerical quantities and temporal expressions [20]. The official setup provides claims, evidence, reasoning traces, and corresponding verdicts. While the task formulation includes ranking reasoning traces according to their usefulness, our system ablates the trace-content component and keeps only the final verdict conclusion attached to each reasoning trace. We therefore cast the task primarily as a verdict-based prediction problem. This is a deliberately limited design: it reduces computation and gives a useful baseline for how much can be recovered from verdict candidates alone, but it does not test whether the reasoning traces themselves contain additional information. The same verifier pipeline is used for English, Arabic, and Spanish. Figure 2 gives an overview of this scoring workflow; the index j is used to preserve the mapping between each reasoning trace and its attached verdict, while the verifier itself scores only the claim c and verdict v_j.

3.3.2.1. Overview

For each claim, the released data includes a list of LLM-generated reasoning traces and a parallel list of verdict conclusions. Our system uses the verdict conclusions as the candidate decisions and ignores the body of the reasoning traces when computing scores. Thus, the model input is a claim–verdict pair, and the model output is a scalar compatibility score. The label space contains three verdicts: `true`, `false`, and `conflicting`. The input format used by both training and inference is:

```
Claim: <claim>
Verdict: <verdict>
```

This format excludes evidence documents and trace text. The verifier only learns whether a candidate verdict is likely to match the gold label.

3.3.2.2. Training Data Construction

We train a binary verifier with the same data construction strategy for all three languages. Each training item provides a claim, the gold label, and a generated verdict field. Instead of expanding the data into one row for every reasoning trace, we convert each item into one claim–verdict example. The target is positive when the generated verdict matches the gold label after lowercasing, and negative otherwise. For the English split, this produces 6,400 examples, with no items skipped. The data is then split into

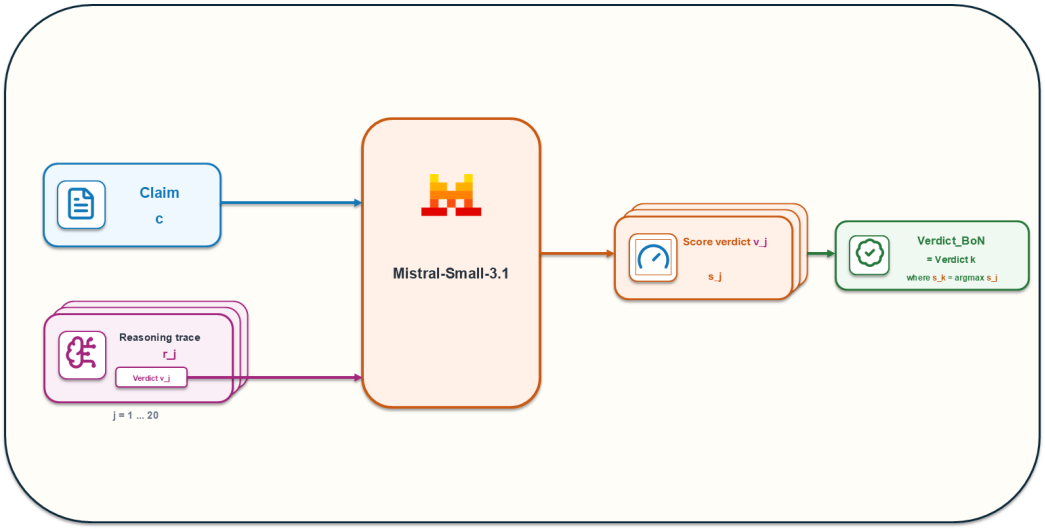

Figure 2: Verdict-only scoring workflow for Task 2.

5,760 training examples and 640 internal validation examples using a stratified 90/10 split. During this step, the evidence documents, reasoning trace bodies, and justifications are not used as model features.

The verifier is built on Mistral-Small-3.1-24B-Base-2503. We load the base model in 4-bit NF4 quantization and attach LoRA adapters to the query, key, and value projection modules. The final hidden state of the last non-padding token is passed to a linear classification head that outputs one logit for the claim–verdict pair. The model is optimized with binary cross-entropy with logits.

3.3.2.3. Inference and Output Construction

For inference, each claim is paired with the verdicts attached to its reasoning traces. We lowercase and deduplicate the verdict list, score each unique verdict once, and copy the score back to all traces with that verdict. Traces with identical verdict conclusions therefore receive identical scores.

The selected verdict is the unique candidate with the highest verifier score. The prediction file stores it as `Verdict_BoN`, together with `BoN_Verdict_list` and `score_list`. Cleaned reasoning traces remain in the output file for scorer compatibility, but they are not scored by the verifier.

3.3.2.4. Efficiency Compared with Trace-Level Scoring

The main computational advantage comes from replacing trace-level scoring with verdict-level scoring. A trace-level baseline would train and evaluate one example per reasoning trace. Our setup uses one compact claim–verdict row per claim, reducing the number of training rows by approximately 17.6–17.8 times across the three languages.

The same idea also reduces inference cost. A trace-level baseline scores every reasoning trace independently. Our system scores only the unique verdict conclusions for each claim and then maps those scores back to the original traces. Since the released test items contain 20 traces per claim but usually only one to three distinct verdicts, the number of verifier calls is reduced by 12.56–14.97 times on the test sets. Table 5 reports these savings by language and cost type, showing the current count, the corresponding trace-level baseline count, and the resulting reduction factor.

3.3.3. Task 3: Generating Full Fact-Checking Articles

Task 3 requires generating a full fact-checking article given a claim, its original rating, and a set of evidence documents [21]. The evidence and rating are already provided, so our work focuses on

Table 5
Efficiency of verdict-level scoring compared with a trace-level baseline.

Language	Cost type	Current	Trace-level baseline	Reduction
English	Training rows	5,760	101,616	17.64×
	Inference score calls	3,418	51,160	14.97×
Arabic	Training rows	2,347	41,728	17.78×
	Inference score calls	814	10,220	12.56×
Spanish	Training rows	2,021	35,936	17.78×
	Inference score calls	1,649	23,280	14.12×

selecting usable evidence, preparing citations, and writing the final article. We run two branches: a lightweight citation branch identifies documents that can support short cited sentences, and a larger generation branch writes the article using only the evidence documents retained by that citation branch. Figure 3 summarizes this workflow.

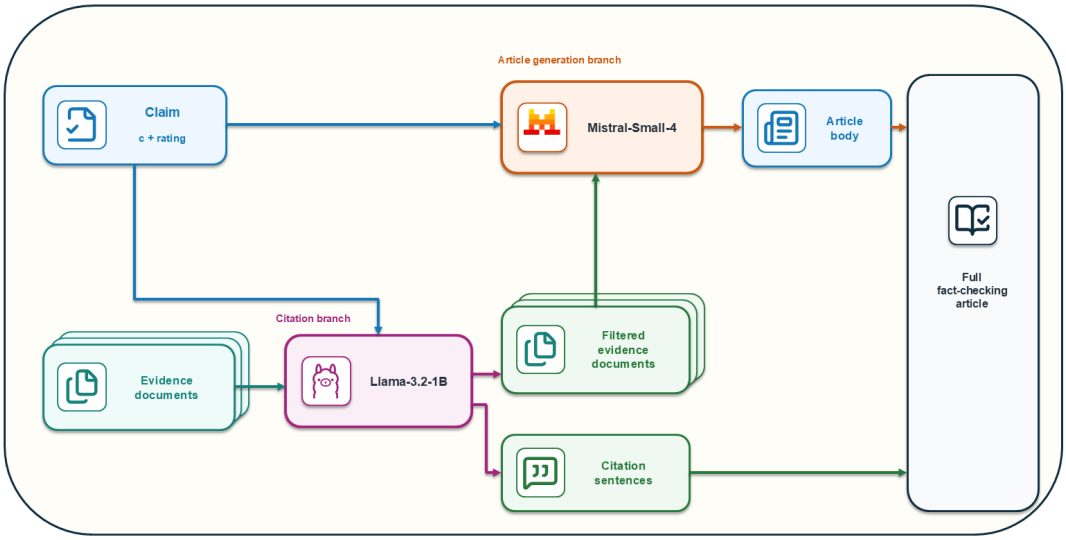

Figure 3: Two-branch article generation workflow for Task 3.

3.3.3.1. Input Formatting

The test data is loaded from the raw claim file and the evidence document directory. Each instance is converted into a structured item containing an identifier, the claim text, the claimant, the original rating, and a mapping from evidence documents to flattened evidence text. This structured item initially keeps the provided evidence documents. After the citation branch runs, the Mistral generation prompt is built only from the accepted evidence excerpts, formatted as source–text pairs and truncated to keep the context manageable.

3.3.3.2. Evidence Filtering and Citation Sentence Generation

The first branch uses Llama-3.2-1B through local Ollama workers as a lightweight sentence extraction module over the provided evidence documents. For each claim–evidence document pair, the model is prompted to produce exactly one short factual sentence grounded in the evidence excerpt. The prompt encourages copying or minimally shortening an explicit factual statement from the evidence and asks the model to avoid inference, explanation, vague language, and source identifiers in the generated sentence. The generated sentence is then cleaned and wrapped as a citation sentence of the form:

```
<sentence> (source: <document>).
```

To reduce unsupported citation sentences, we check each generated sentence against the evidence excerpt that produced it. A sentence is kept only if it is short, factual, and supported by the excerpt without adding extra interpretation. We reject sentences that are vague, introduce unsupported details, include source labels inside the sentence, or fail the support check. Rejected sentences trigger up to three Llama retries. If those attempts fail, we choose an exact sentence from the evidence. If no document for a claim yields a retained sentence, the citation branch emits a short status sentence instead of forcing a citation. In the final run, 4,134 evidence documents were retained and 4,845 were rejected; 78 claims had no retained evidence sentence. Table 6 summarizes these counts.

Table 6
Citation sentence construction statistics for Task 3.

Statistic	Count
Original evidence documents	8,979
Retained evidence documents	4,134
Claims with no retained evidence sentence	78

3.3.3.3. Article Generation from Filtered Evidence

The second branch generates the main fact-checking article with Mistral-Small-4-119B-2603-NVFP4 through an OpenAI-compatible API. For each claim, we keep only the evidence documents accepted by the citation branch and build a prompt containing the claim, claimant, original rating, and those filtered evidence excerpts. The prompt asks the model to write a complete fact-checking article with a headline, detailed analysis, and a final verdict with rationale.

In the final run, Mistral writes the narrative article body from the citation-filtered evidence set. The citation branch supplies the source-grounded sentences appended later in the output.

3.3.3.4. Output Merging

The final prediction file is constructed by concatenating the Mistral article body with the Llama citation-sentence block for the same claim ID. The merged output therefore has the following structure:

```
<Mistral fact-checking article>
<blank line>
<Llama/Ollama citation sentences>
```

The final file contains 1,158 records with the required fields `id` and `factchecking_article`. Each output article combines a narrative fact-checking body with an explicit set of source-linked factual sentences.

Table 7
Configuration of the Task 3 article generation workflow.

Component	Setting
Input instances	1,158 test claims
Citation branch model	Llama-3.2-1B through Ollama
Citation branch objective	Generate one short sentence entailed by each evidence excerpt
Citation validation	Strict support check against the corresponding evidence excerpt
Retry and fallback	Up to 3 Llama retries; exact evidence-sentence fallback
Main article model	Mistral-Small-4-119B-2603-NVFP4
Generation input	Claim, claimant, original rating, and citation-filtered evidence excerpts
Generation setting	Low-temperature parallel generation
Final output	Mistral article body concatenated with source-linked citation sentences

4. Experimental Results

4.1. Implementation

All experiments were executed on a shared GPU cluster used for training, local inference, preprocessing, and output construction. We assigned different GPU budgets to the tasks according to memory and throughput needs. Task 1 consumed most of the heavy model budget because it required full-corpus encoding and reranking. Task 2 was kept small enough to train separate verifiers for three languages. Task 3 split work between a small local model for repeated evidence processing and a larger model for the final article text. The task-specific implementation settings are summarized below.

4.1.1. Task 1 Implementation

Task 1 uses two separate training and inference stages on the cluster.

4.1.1.1. Stage 1: Bi-Encoder Retrieval

The first-stage retrieval implementation uses Harrier-OSS-v1-27B and Linq-Embed-Mistral as dense embedding backbones. The retriever is trained with LoRA and periodically refreshes hard-negative candidates from the full paper collection. Table 8 reports the main retrieval-stage parameters.

Table 8
Configuration of the Stage 1 bi-encoder retrieval model.

Parameter	Value
Embedding backbones	microsoft/harrier-oss-v1-27b; Linq-AI-Research/Linq-Embed-Mistral
Training epochs	2
Candidate refresh size	Top 100 papers per query
Training candidates per query	Top 5 from the refreshed candidate pool
Loss	Pairwise contrastive loss with cosine distance
Margin	0.5
Batch size	4
Gradient accumulation steps	4
Effective batch size	16
Learning rate	2×10^{-5}
Encoding batch size	8
Precision	bfloat16
Optimizer	Adafactor
LoRA rank / alpha / dropout	32 / 64 / 0.05

4.1.1.2. Stage 2: Cross-Encoder Reranking

The second-stage reranker is trained on the top-five candidate groups produced by the retriever. Besides Qwen3-VL-Reranker-8B, we also experiment with Qwen3-Reranker at 8B, 4B, and 0.6B scales. Its implementation focuses on listwise reranking over small candidate sets, not full-corpus retrieval. Table 9 gives the reranking configuration.

4.1.2. Task 2 Implementation

For Task 2, we fine-tune the verdict-only verifier with quantization and LoRA adapters to reduce memory use. The same training setup is used for English, Arabic, and Spanish, with the claim–verdict format described in Section 3. Table 10 summarizes the verifier configuration.

4.1.3. Task 3 Implementation

For Task 3, the citation branch uses Llama-3.2-1B through local Ollama workers, while the article branch uses Mistral-Small-4-119B-2603-NVFP4 through an OpenAI-compatible interface. Table 11 reports the

Table 9
Configuration of the Stage 2 reranking setup.

Parameter	Value
Training type	Full fine-tuning
Training objective	Listwise reranking
Training candidates per query	Top 5 from Stage 1
Positive / negative candidates	1 positive and 4 negatives
Training rows	11,685
Skipped groups	1,168 groups without gold in top 5
Training epochs	1
Maximum sequence length	12,144
Gradient accumulation steps	16
Learning rate	6×10^{-6}
Scheduler / warmup	Cosine / 0.05
Optimizer	AdamW fused
Weight decay	0.1
Precision	bfloat16
Reranking score	`logit(yes)` minus `logit(no)`

Table 10
Configuration of the Task 2 verdict-only verifier.

Component	Setting
Base model	Mistral-Small-3.1-24B-Base-2503
Loss	Binary cross-entropy with logits
Quantization	4-bit NF4 with bfloat16 computation
Parameter-efficient tuning	LoRA on `q_proj`, `k_proj`, and `v_proj`
LoRA configuration	Rank 8, alpha 16, dropout 0.0
Training epochs	2
Learning rate	1×10^{-4} with cosine schedule and 5% warmup

generation parameters used for both branches. In the final run, the Mistral branch generated 1,158 article bodies with concurrency 10. Its evidence input came from the citation-filtered documents: 1,080 instances had retained evidence documents and 78 instances had none.

Table 11
Generation parameters for the Task 3 citation and article branches.

Branch	Model	Max new tokens	Temperature	Top p	Top k
Citation sentence generation	Llama-3.2-1B	512	0.0	0.9	30
Article generation	Mistral-Small-4-119B-2603-NVFP4	3,000	0.1	0.9	30

4.2. Results and Discussions

We report development-set results for the components where local validation was available and official test-set leaderboard results for final comparison. Task 1 and Task 3 official results are taken from the CheckThat! 2026 public leaderboards[10][11], while Task 2 test results were submitted through the official CodaBench competitions listed on the task page[12].

[10] https://checkthat.gitlab.io/clef2026/task1/
[11] https://checkthat.gitlab.io/clef2026/task3/
[12] https://checkthat.gitlab.io/clef2026/task2/

4.2.1. Task 1 Results

4.2.1.1. Development Set Results

We report development-set results separately for the two stages of the Task 1 pipeline. The first-stage bi-encoder is evaluated as a full-corpus retriever, while the second-stage cross-encoder is evaluated as a reranker over the top-five candidates returned by the retriever. This split also makes the error source easier to interpret: if the gold paper is absent from the top five, the reranker cannot recover it; if it is present but not ranked first, the bottleneck is the second stage.

Stage 1: Bi-Encoder Retrieval

For first-stage retrieval, Table 12 first lists the bi-encoder runs used in our development experiments, including the model, training language, and epoch. Table 13 then reports the retrieval metrics for these runs. This separation is useful because a single checkpoint can be evaluated on more than one language; in particular, we evaluate the best English Harrier checkpoint on French and German to examine cross-language transfer.

Table 12
Task 1 first-stage bi-encoder run definitions.

Run ID	Model	Train data	Epoch
H-BASE	microsoft/harrier-oss-v1-27b	–	0
L-BASE	Linq-AI-Research/Linq-Embed-Mistral	–	0
H-EN-FT	microsoft/harrier-oss-v1-27b	English	2
L-EN-FT	Linq-AI-Research/Linq-Embed-Mistral	English	2
H-FR-FT	microsoft/harrier-oss-v1-27b	French	2
H-DE-FT	microsoft/harrier-oss-v1-27b	German	2

Table 13
Task 1 development results for first-stage bi-encoder retrieval.

Eval lang.	Run ID	Epoch	R@1	R@5	MRR@5	NDCG@5
en	**H-EN-FT**	**2**	**0.7022**	**0.8415**	**0.7579**	**0.7790**
en	L-EN-FT	2	0.6615	0.8038	0.7188	0.7402
en	H-BASE	0	0.6095	0.7700	0.6725	0.6970
en	L-BASE	0	0.5811	0.7365	0.6419	0.6657
fr	H-EN-FT	2	**0.6952**	0.8433	**0.7582**	**0.7798**
fr	H-FR-FT	2	0.6880	0.8433	0.7522	0.7753
de	H-DE-FT	2	**0.6244**	**0.7902**	0.6862	0.7121
de	H-EN-FT	2	**0.6244**	0.7824	**0.6903**	**0.7137**

The first-stage results led us to keep the English fine-tuned Harrier checkpoint for all three languages. This was not our initial expectation. On English, fine-tuning Harrier gives a clear gain over both pretrained embedding baselines. On French, the English checkpoint slightly outperforms the French-only checkpoint. On German, the German-specific checkpoint retrieves marginally more gold papers within the top five, but the English checkpoint gives slightly better MRR@5 and NDCG@5. The larger English split seems to provide a stronger training signal than the smaller target-language splits.

Stage 2: Cross-Encoder Reranking

For the second-stage cross-encoder, Table 14 defines the reranker runs, and Table 15 reports the corresponding metrics. The multilingual experiments evaluate the English fine-tuned VL reranker directly on French and German, and compare it with separate VL checkpoints trained on the corresponding target-language data.

The English reranking experiments compare several models from the Qwen reranker family. Increasing model capacity generally improves the ranking metrics: Qwen3-Reranker-0.6B is clearly weaker than the 4B and 8B variants, and the 8B rerankers give the best English results. Among them, Qwen3-VL-Reranker-8B obtains the highest R@1, MRR@5, and NDCG@5, slightly outperforming the text-only Qwen3-Reranker-8B. We therefore use the 8B VL reranker as the main checkpoint for the follow-up multilingual comparison.

Table 14
Task 1 second-stage cross-encoder reranker run definitions.

Run ID	Model	Train data	Epoch
CE-EN-8B	Qwen3-Reranker-8B	English	1
CE-EN-VL	Qwen3-VL-Reranker-8B	English	1
CE-EN-4B-E1	Qwen3-Reranker-4B	English	1
CE-EN-0.6B	Qwen3-Reranker-0.6B	English	1
CE-FR-VL-FT	Qwen3-VL-Reranker-8B	French	1
CE-DE-VL-FT	Qwen3-VL-Reranker-8B	German	1

Table 15
Task 1 development results for second-stage cross-encoder reranking.

Eval lang.	Run ID	Epoch	R@1	R@5	MRR@5	NDCG@5
en	CE-EN-8B	1	0.7383	0.8410	0.7817	0.7968
en	**CE-EN-VL**	**1**	**0.7429**	**0.8410**	**0.7844**	**0.7988**
en	CE-EN-4B-E1	1	0.7327	0.8410	0.7774	0.7935
en	CE-EN-0.6B	1	0.6909	0.8410	0.7514	0.7741
fr	**CE-EN-VL**	**1**	**0.7422**	**0.8419**	**0.7859**	**0.8002**
fr	CE-FR-VL-FT	1	0.7179	0.8419	0.7726	0.7903
de	**CE-EN-VL**	**1**	**0.6606**	**0.7824**	**0.7120**	**0.7299**
de	CE-DE-VL-FT	1	0.6554	0.7824	0.7085	0.7272

The multilingual reranking results show a similar pattern to the first-stage retrieval results. The English fine-tuned VL reranker transfers well to French and German, outperforming the separate checkpoints trained on the smaller target-language training sets. Since R@5 is fixed by the top-five candidate pool from the first stage, the main differences appear in R@1, MRR@5, and NDCG@5. We therefore use the English fine-tuned VL reranker for the final multilingual reranking setup instead of the additional French- or German-specific fine-tuning runs. This was one of the more useful development findings, because it avoided maintaining three separate reranker checkpoints without hurting the validation metrics.

4.2.1.2. Test Set Leaderboard

Table 16 reports the top five teams on the official Task 1 evaluation leaderboard, which lists 37 participating teams. Outsider ranked second overall with an average MRR@5 of 0.7580, only 0.0048 behind Claim2Source and 0.0116 ahead of the third-ranked team. This narrow gap indicates that the final system was competitive with the best submitted retrieval pipelines despite using the same two-stage architecture for all three languages.

The language-level results are not driven by only one language. The strongest relative performance is German, where Outsider ranked first with an MRR@5 of 0.7228, 0.0075 above the next best score. On English, the run trails Claim2Source by only 0.0017 MRR@5. French is weaker, although still within the top three. The shared multilingual setup was therefore a reasonable final choice, but French remains the clearest place where a language-specific improvement might matter.

Table 16
Task 1 official test leaderboard results: top five teams by average MRR@5.

Rank	Team	Avg.	De	En	Fr
1	Claim2Source	0.7628	0.7153	0.7819	0.7912
2	**Outsider**	**0.7580**	**0.7228**	**0.7802**	**0.7709**
3	Boyes Boys	0.7464	0.7070	0.7571	0.7752
4	CheckMate	0.7194	0.6861	0.7343	0.7378
5	RETRIX	0.7108	0.6806	0.7117	0.7400

4.2.2. Task 2 Results

4.2.2.1. Development Set Results

Task 2 uses the same verdict-only pipeline for English, Arabic, and Spanish, but the verifier is trained separately for each language using the corresponding training split. Thus, the input construction and scoring procedure are shared across languages, while the final checkpoint is language-specific. Table 17 reports development performance for the checkpoints used in our submissions.

For Task 2, we trained one main verdict-only configuration per language. Arabic obtains the strongest classification performance, with high F1 for both False and True labels. English gives the highest development MRR@5. Spanish is less balanced: False claims are handled well, but True and Conflicting claims are weaker. This is the main warning sign for the compact setup. If the verdict candidates are not informative enough, the verifier has no evidence text or trace content to fall back on.

Table 17
Task 2 development results for verdict-only verification.

Language	MRR@5	Macro F1	F1 False	F1 True	F1 Conflicting
English	0.3278	0.5548	0.7307	0.4745	0.4591
Arabic	0.2800	0.8095	0.8343	0.7846	–
Spanish	0.2832	0.4619	0.8720	0.2385	0.2752

4.2.2.2. Test Set Leaderboard

Table 18 reports the official test leaderboard results for each Task 2 language. The English leaderboard includes 21 participating teams, the Spanish leaderboard includes 14 teams, and the Arabic leaderboard includes 13 teams. For English and Arabic, we list the top five teams; for Spanish, we include the top six teams so that the Outsider submission is shown. The Arabic run gave the strongest relative result, ranking second with an average score of 0.6004, only 0.0039 behind the first-ranked team. English ranked fourth with 0.4217, within 0.0069 of the best score. Spanish was more difficult for our setup: Outsider ranked sixth with 0.4451, 0.0396 below the top score.

Across the three test leaderboards, the verdict-only approach is competitive in Arabic and English and weaker in Spanish. The result is useful mainly as a resource-efficiency baseline. A fairer accuracy comparison still requires trace-aware and evidence-aware variants trained under the same budget.

Table 18
Task 2 official test leaderboard results by language.

Lang.	Rank	Team	Avg.	Macro F1	R@5	True F1	False F1
en	1	ahmadraza123	0.4286	0.6150	0.2421	0.7125	0.8888
en	2	VeriFind	0.4261	0.5988	0.2534	0.7294	0.8881
en	3	GOS-DSGT	0.4234	0.6002	0.2466	0.7400	0.8903
en	**4**	`team-outsider`	**0.4217**	**0.5979**	**0.2456**	**0.7114**	**0.8909**
en	5	nadie	0.4217	0.6011	0.2424	0.7180	0.8896
ar	1	CheckMates	0.6043	0.8679	0.3406	0.8426	0.8933
ar	**2**	`team-outsider`	**0.6004**	**0.8583**	**0.3426**	**0.8325**	**0.8841**
ar	3	mircean2	0.5992	0.8602	0.3382	0.8345	0.8860
ar	4	Sagnik Sinha	0.5913	0.8485	0.3341	0.8219	0.8752
ar	5	mohamed_mamdouh	0.5876	0.8430	0.3322	0.8169	0.8691
es	1	CheckMates	0.4847	0.6749	0.2946	0.6154	0.9131
es	2	mircean2	0.4778	0.6607	0.2949	0.5742	0.9061
es	3	ahmadraza123	0.4777	0.6508	0.3046	0.5949	0.8881
es	4	5people	0.4606	0.6382	0.2829	0.5982	0.9024
es	5	mary_toapanta	0.4582	0.6342	0.2821	0.5182	0.9080
es	**6**	`team-outsider`	**0.4451**	**0.6135**	**0.2766**	**0.6108**	**0.8937**

4.2.3. Task 3 Results

Task 3 is evaluated only on the official test set in our experiments. Table 19 reports the top five teams on the official leaderboard, which includes 10 participating teams. Outsider ranked first overall with a mean score of 0.546, ahead of UTS by 0.062 and ahead of the third-ranked DS GT CheckThat team by 0.118.

The main difference between Outsider and the other top systems is citation quality. UTS achieved higher entailment and evidence coverage, but its citation precision and recall were both 0.299. Outsider obtained 0.671 for both citation precision and citation recall. The citation branch therefore appears to be the most important part of the Task 3 run. It reduces evidence coverage, but it gives the article generator a cleaner evidence set and leaves a citation block that is easier to audit.

Table 19
Task 3 official test leaderboard results: top five teams by mean score.

Rank	Team	Entailment	Citation Precision	Citation Recall	Evidence Coverage	Mean Score
1	**Outsider**	**0.278**	**0.671**	**0.671**	**0.563**	**0.546**
2	UTS	0.341	0.299	0.299	0.996	0.484
3	DS GT CheckThat	0.295	0.240	0.276	0.902	0.428
4	Facty	0.229	0.231	0.254	0.995	0.427
5	UCPH RMT	0.364	0.144	0.174	0.993	0.419

The Task 3 result also shows a trade-off that is easy to miss from the mean score alone. Entailment is lower than in some competing systems. This does not mean that Mistral used evidence outside the filter: the article prompt was built from the evidence documents accepted by the citation branch. The issue is narrower. The citation branch checks short sentences one by one, while the article body is a longer generated text from the accepted excerpts. During article writing, Mistral can paraphrase, compress, or combine information from those accepted excerpts in ways that are not checked by the sentence-level citation filter. Filtering evidence helps control the citation block; it does not fully verify the semantics of the whole article.

The fallback behavior also matters. Exact evidence sentences are less elegant than generated ones, but they are easier to justify. When no document passes the support check, the pipeline emits a status sentence instead of forcing a citation. Those cases are conservative, and the article branch then has little or no citation-filtered evidence to work with.

5. Conclusion

In this paper, we described the Outsider team's participation in the CheckThat! Lab at CLEF 2026. The three tasks required different parts of a fact-checking pipeline, so the final submission used three different strategies.

For Task 1, the two-stage retrieval architecture was the strongest part of the submission. A large bi-encoder retriever followed by a cross-encoder reranker worked well for matching informal social media posts to scientific papers, ranking second overall and first on German. The cost is that this component is heavy: the gains came from large models, hard-negative refreshes, and reranking, which makes efficiency the main issue for future use.

For Task 2, the verdict-only verifier intentionally ignored evidence documents and reasoning trace bodies, using only compact claim–verdict pairs. This reduced training and inference cost by a large factor and still produced competitive official results, especially for Arabic and English. The Spanish result was weaker, which is where the compact input seems least reliable.

For Task 3, the best result came from treating citation construction as a separate step before article generation. The run ranked first overall mainly because the retained citation sentences were more

consistently tied to evidence documents. The weaker entailment score points to the next problem: the article body needs its own verification step.

Future work should therefore focus on the weak point of each task. For Task 1, we will investigate lighter retrieval and reranking variants, including smaller rerankers, model distillation, embedding caching, and more efficient multilingual transfer strategies. For Task 2, we will compare verdict-only, reasoning-trace-aware, and evidence-aware inputs under the same training and inference budget. For Task 3, we will add stronger article-level verification, such as entailment-aware reranking or post-generation checking, while keeping the citation filtering step that was useful in the official run.

Acknowledgments

This work was supported by the Eiffel Excellence Scholarship awarded to Quy Thanh Le by the French government, as well as by the R2I and MoFED teams of the LIS Laboratory. This work was granted access to the Jean Zay supercomputer under the GENCI allocation AD011017091.

Declaration on Generative AI

During the preparation of this work, the author(s) used GPT-5 for grammar and spelling checks, as well as for paraphrasing and rewording. After using these tools, the author(s) reviewed and edited the content as needed, and take full responsibility for the publication's content.

References

[1] S. Vosoughi, D. Roy, S. Aral, The spread of true and false news online, Science 359 (2018) 1146–1151. URL: https://www.science.org/doi/abs/10.1126/science.aap9559. doi:10.1126/science.aap9559. arXiv:https://www.science.org/doi/pdf/10.1126/science.aap9559.

[2] Z. Guo, M. Schlichtkrull, A. Vlachos, A survey on automated fact-checking, Transactions of the Association for Computational Linguistics 10 (2022) 178–206. URL: https://aclanthology.org/2022.tacl-1.11/. doi:10.1162/tacl_a_00454.

[3] X. Zeng, A. S. Abumansour, A. Zubiaga, Automated fact-checking: A survey, Language and Linguistics Compass 15 (2021) e12438.

[4] N. Hassan, G. Zhang, F. Arslan, J. Caraballo, D. Jimenez, S. Gawsane, S. Hasan, M. Joseph, A. Kulkarni, A. K. Nayak, V. Sable, C. Li, M. Tremayne, Claimbuster: the first-ever end-to-end fact-checking system, Proc. VLDB Endow. 10 (2017) 1945–1948. URL: https://doi.org/10.14778/3137765.3137815. doi:10.14778/3137765.3137815.

[5] R. Panchendrarajan, A. Zubiaga, Claim detection for automated fact-checking: A survey on monolingual, multilingual and cross-lingual research, Natural Language Processing Journal 7 (2024) 100066. URL: http://dx.doi.org/10.1016/j.nlp.2024.100066. doi:10.1016/j.nlp.2024.100066.

[6] V. Setty, Factcheck editor: Multilingual text editor with end-to-end fact-checking, 2024. URL: https://arxiv.org/abs/2404.19482. arXiv:2404.19482.

[7] J. Thorne, A. Vlachos, Automated fact checking: Task formulations, methods and future directions, in: E. M. Bender, L. Derczynski, P. Isabelle (Eds.), Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 3346–3359. URL: https://aclanthology.org/C18-1283/.

[8] J. Thorne, A. Vlachos, C. Christodoulopoulos, A. Mittal, FEVER: A large-scale dataset for fact extraction and VERification, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, 2018, pp. 809–819. URL: https://aclanthology.org/N18-1074/. doi:10.18653/v1/N18-1074.

[9] R. Aly, Z. Guo, M. Schlichtkrull, J. Thorne, A. Vlachos, C. Christodoulopoulos, O. Cocarascu, A. Mittal, Feverous: Fact extraction and verification over unstructured and structured information, 2021. URL: https://arxiv.org/abs/2106.05707. arXiv:2106.05707.

[10] I. Augenstein, C. Lioma, D. Wang, L. Chaves Lima, C. Hansen, C. Hansen, J. G. Simonsen, MultiFC: A real-world multi-domain dataset for evidence-based fact checking of claims, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Association for Computational Linguistics, 2019, pp. 4685–4697. URL: https://aclanthology.org/D19-1475/. doi:10.18653/v1/D19-1475.

[11] M. Schlichtkrull, Z. Guo, A. Vlachos, AVeriTeC: A dataset for real-world claim verification with evidence from the web, in: Advances in Neural Information Processing Systems, volume 36, 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/2ac07edfab90a96a93f0b044ce3e71a5-Abstract-Datasets_and_Benchmarks.html.

[12] N. Sagara, E. Peters, Consumer understanding and use of numeric information in product claims, in: D. R. Deeter-Schmelz (Ed.), Proceedings of the 2010 Academy of Marketing Science (AMS) Annual Conference, Springer International Publishing, Cham, 2015, pp. 245–245. doi:10.1007/978-3-319-11797-3_140.

[13] V. V, A. Anand, A. Anand, V. Setty, Quantemp: A real-world open-domain benchmark for fact-checking numerical claims, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 650–660. URL: https://doi.org/10.1145/3626772.3657874. doi:10.1145/3626772.3657874.

[14] J. Wallat, A. Abdallah, A. Jatowt, A. Anand, A study into investigating temporal robustness of llms, 2025. arXiv:2503.17073.

[15] F. Alam, J. M. Struß, T. Chakraborty, S. Dietze, S. Hafid, K. Korre, A. Muti, P. Nakov, F. Ruggeri, S. Schellhammer, V. Setty, M. Sundriyal, K. Todorov, V. V., The clef-2025 checkthat! lab: Subjectivity, fact-checking, claim normalization, and retrieval, in: C. Hauff, C. Macdonald, D. Jannach, G. Kazai, F. M. Nardini, F. Pinelli, F. Silvestri, N. Tonellotto (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2025, pp. 467–478.

[16] 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.

[17] 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, Overview of the CLEF-2026 CheckThat! Lab: Advancing multilingual fact-checking, in: M. Hagen, M. Potthast, B. Stein, P. Schaer, E. Zangerle, S. MacAvaney, J. M. Struß, E. S. Salido, A. Barrón-Cedeño, Alba García Seco de Herrera (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Springer, 2026.

[18] 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: [39], 2026.

[19] 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, in: Working Notes of CLEF 2025 - Conference and Labs of the Evaluation Forum, Madrid, Spain, 2025.

[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: [39], 2026.

[21] D. Sahnan, T. Chakraborty, P. Nakov, Overview of the CLEF-2026 CheckThat! lab task 3 on generating full fact-checking articles, in: [39], 2026.

[22] L. Graves, Anatomy of a fact check: Objective practice and the contested epistemology of fact checking, Communication, Culture and Critique 10 (2017) 518–537. doi:10.1111/cccr.12163.

[23] K. Khan, R. Wang, P. Poupart, WatClaimCheck: A new dataset for claim entailment and inference, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics,

Association for Computational Linguistics, 2022, pp. 1293–1304. URL: https://aclanthology.org/2022.acl-long.92/. doi:10.18653/v1/2022.acl-long.92.

[24] 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 (2026).

[25] D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, H. Hajishirzi, Fact or fiction: Verifying scientific claims, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2020, pp. 7534–7550. URL: https://aclanthology.org/2020.emnlp-main.609/. doi:10.18653/v1/2020.emnlp-main.609.

[26] D. Wadden, K. Lo, L. L. Wang, A. Cohan, I. Beltagy, H. Hajishirzi, SciFact-Open: Towards open-domain scientific claim verification, in: Findings of the Association for Computational Linguistics: EMNLP 2022, Association for Computational Linguistics, 2022, pp. 4719–4734. URL: https://aclanthology.org/2022.findings-emnlp.347/. doi:10.18653/v1/2022.findings-emnlp.347.

[27] A. Cohan, S. Feldman, I. Beltagy, D. Downey, D. S. Weld, SPECTER: Document-level representation learning using citation-informed transformers, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2020, pp. 2270–2282. URL: https://aclanthology.org/2020.acl-main.207/. doi:10.18653/v1/2020.acl-main.207.

[28] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using siamese BERT-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019, pp. 3982–3992. URL: https://aclanthology.org/D19-1410/. doi:10.18653/v1/D19-1410.

[29] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2020, pp. 6769–6781. URL: https://aclanthology.org/2020.emnlp-main.550/. doi:10.18653/v1/2020.emnlp-main.550.

[30] V. Venktesh, V. Setty, A. Anand, M. Hasanain, B. Bendou, H. Bouamor, F. Alam, G. Iturra-Bocaz, P. Galuščáková, Overview of the CLEF-2025 CheckThat! lab task 3 on fact-checking numerical claims, in: [40], 2025.

[31] Q. T. Le, I. Badache, A. Yacoub, M. E. A. Hamri, LIS at CheckThat! 2025: Multi-stage open-source large language models for fact-checking numerical claims, in: [40], 2025, pp. 1009–1023. URL: https://ceur-ws.org/Vol-4038/paper_78.pdf.

[32] P.-J. Yang, Y. T. Chen, Y. Chen, D. Cer, Nt5?! training t5 to perform numerical reasoning, 2021. URL: https://arxiv.org/abs/2104.07307. arXiv:2104.07307.

[33] J. Zhang, Y. Moshfeghi, Elastic: Numerical reasoning with adaptive symbolic compiler, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 12647–12661. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/522ef98b1e52f5918e5abc868651175d-Paper-Conference.pdf.

[34] L. Pan, X. Wu, X. Lu, A. T. Luu, W. Y. Wang, M.-Y. Kan, P. Nakov, Fact-checking complex claims with program-guided reasoning, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 6981–7004. URL: https://aclanthology.org/2023.acl-long.386/. doi:10.18653/v1/2023.acl-long.386.

[35] 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, Online, 2020, pp. 7352–7364. URL: https://aclanthology.org/2020.acl-main.656/. doi:10.18653/v1/2020.acl-main.656.

[36] A. Fan, A. Piktus, F. Petroni, G. Wenzek, M. Saeidi, A. Vlachos, A. Bordes, S. Riedel, Generating fact checking briefs, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference

on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 7147–7161. URL: https://aclanthology.org/2020.emnlp-main.580/. doi:10.18653/v1/2020.emnlp-main.580.

[37] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W.-t. Yih, T. Rocktaschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 9459–9474. URL: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[38] F. Zeng, W. Gao, JustiLM: Few-shot justification generation for explainable fact-checking of real-world claims, Transactions of the Association for Computational Linguistics 12 (2024) 334–354. URL: https://aclanthology.org/2024.tacl-1.19/. doi:10.1162/tacl_a_00650.

[39] 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.

[40] G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 - Conference and Labs of the Evaluation Forum, CLEF 2025, Madrid, Spain, 2025.