KNU Fact at CheckThat! 2026: Exploring context engineering and chain-of-thought strategies for fact-checking article generation

CheckThat! at CLEF 2026

Bohdan Karlash

Taras Shevchenko National University of Kyiv, Academician Glushkov Avenue, 4g, Kyiv, 03187, Ukraine

Abstract

In this paper, we describe the methodology and results of our participation in the 2026 CheckThat! Lab "Task 3: Generating Full Fact-Checking Articles". The task targets the final stage of the CLEF CheckThat! lab pipeline. Participants are tasked with developing methods to generate a full fact-checking article, given a claim, its veracity, and a set of evidence documents consulted to fact-check the claim. In this paper, we focus on context engineering approaches such as Natural Language Inference (NLI) based and hybrid retrieve-and-rerank approaches to evidence extraction as well as Chain-of-Thought (CoT) prompting strategies for open-source Large Language Models (LLMs). Our approach, which employed a two-stage pipeline consisting of an NLI-based evidence extraction step followed by a one-shot CoT generation step using an open-source LLM, achieved 9th place on the leaderboard with a mean score of 0.312, outperforming the baseline of 0.272. Additional experiments were conducted to improve this approach, but their results could not be submitted during the evaluation cycle due to time constraints and hardware limitations. Nevertheless, two approaches show significant improvement. Changing the LLM to a newer one along with an enhanced citation control and evidence utilization results in a mean score of 0.419, theoretically placing 5th or 6th (depending on rounding of scores). Using a hybrid retrieve-and-rerank approach to evidence extraction as well as enhanced prompting and post-processing citation correction improves the mean entailment score and mean citation recall, but lowers the evidence coverage, achieving a mean score of 0.393, theoretically placing 7th on the leaderboard. In this paper, both the leaderboard approach and additional approaches are discussed. The results demonstrate the potential effectiveness of using such approaches for the generation of fact-checking articles; however, based on the evaluation scores and the leaderboard, there are still possible improvements to be made. We also present an optimized evaluation pipeline that runs locally and is approximately four times faster than the one provided by the organizers. Additionally, during experiments, we encountered such known issues as hallucination and repetition, which slow the generation process and worsen its quality.

Keywords

Automated fact-checking, Fact-checking article generation, Large Language Models, Chain-of-Thought prompting, Natural language inference, Evidence retrieval, Justification production, CheckThat!, CLEF 2026

1. Introduction

The proliferation of online misinformation poses a significant and growing challenge to public discourse, democratic processes, and scientific communication. Manually verifying the ever-increasing volume of claims circulating across news outlets, social media platforms, and political debate exceeds the capacity of professional fact-checkers, motivating the development of Automated Fact-Checking (AFC) systems. AFC encompasses the use of Natural Language Processing (NLP), machine learning, and information retrieval techniques to automatically assess the veracity of claims — a task that has attracted sustained research attention over the past decade [1, 2, 3].

The scientific literature consistently models AFC as a multi-stage pipeline in which distinct components transform raw, unverified input into a structured verdict. Although specific architectures vary depending on modality (text vs. multimodal) and domain (political vs. scientific), the prevailing framework comprises four primary stages: claim detection, evidence retrieval, claim verification, and

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

bohdankarlash@knu.ua (B. Karlash)

0009-0001-9965-3284 (B. Karlash)

justification production. Although the first three steps are equally important in reaching a verdict, that verdict should also be properly explained in the fourth stage.

To increase transparency and user trust in AFC systems, modern pipelines include a dedicated stage to generate natural language explanations of how a veracity verdict was reached [1, 4]. Justifications may take one of three forms: highlighted evidence segments drawn directly from the source text; structured Subject-Predicate-Object (SPO) triples that encode the relational content of the verdict; or fluent natural language summaries synthesized from the retrieved evidence [5, 4].

Task 3 [6] of the CLEF 2026 CheckThat! Lab [7, 8, 9] focuses on the justification production stage, specifically in the form of generating full fact-checking articles based on a provided claim, a set of evidence documents, and a veracity verdict. The validation baseline for this task uses a closed-source LLM and the test baseline supports the use of both a closed-source and an open-source LLM. LLMs have increasingly been used for this type of task to leverage both their generative capacity and reasoning ability to produce explanatory text alongside veracity verdicts [4].

In this study, we present our approach that was submitted during the test phase of the task and two alternative approaches that aim to improve upon it. In our submission, we employed a two-stage pipeline consisting of an NLI-based evidence extraction step that was used to filter evidence as the model's context length is not large enough to take in entire articles, followed by a one-shot CoT generation step using a quantized open-source LLM. Our experiments demonstrate that open-source LLMs, even quantized ones, can be used for justification production and generation of fact-checking articles on par with closed-source API-based models, given an extensive CoT prompt with a detailed one-shot example. Performance can be significantly improved by utilizing enhanced citation control and evidence utilization or by extending the extraction step with hybrid retrieve-and-rerank, which improves the entailment to the detriment of evidence coverage.

2. Related Work

2.1. Fact-Checking Justification and Article Generation.

Generating natural language justifications for fact-checking verdicts has attracted growing attention as a means of increasing transparency in automated pipelines. The task of automatically generating fact-checking explanations was introduced by [10], who framed it as a joint multi-task problem where the model simultaneously predicts the veracity of a claim and extracts an explanation from a single document of ruling comments. Their results showed that an extractive summarization approach can successfully describe the reasons behind a veracity prediction and improve the performance of the fact-checking system. This line of work was extended to the public health domain by [11], who developed an abstractive explanation generation model to distill explanations for users with limited domain expertise. Because existing work focuses predominantly on short summaries, [12] argued for extending the typical AFC pipeline with automatic generation of full fact-checking articles and proposed QRAFT, an LLM-based agentic framework that mimics the writing workflow of human fact-checkers. Building on this direction, Task 3 of the CLEF 2026 CheckThat! Lab [6] serves as the first benchmark to evaluate the generation of complete fact-checking articles.

2.2. Evidence Retrieval: NLI Cross-Encoders and Bi-Encoders.

Natural Language Inference has been a core component of evidence-based fact verification since the introduction of the FEVER benchmark [13], which established the claim – evidence entailment classification as the standard evaluation paradigm. In retrieval-augmented settings, NLI cross-encoders provide precise relevance estimation but scale poorly with corpus size. Bi-encoders such as Sentence-BERT [14] and the Dense Passage Retrieval model [15] address this by encoding queries and passages independently into dense vector representations, enabling sub-linear retrieval via approximate nearest-neighbor search. The hybrid retrieve-and-rerank paradigm — using a bi-encoder for efficient first-stage recall followed by a cross-encoder for precise reranking — has been shown to be an effective and

computationally tractable strategy for evidence selection in open-domain question answering and fact verification tasks [15].

2.3. Chain-of-Thought Prompting for Reasoning and Fact-Checking.

Currently, there are two main methods that are used in conjunction with LLMs to achieve the best results. CoT prompting [16] guides LLMs to articulate intermediate reasoning steps before arriving at a final verdict, making generated justifications highly transparent and interpretable for human fact-checkers [4, 17]. To further enhance factuality and mitigate hallucinations, advanced frameworks exploit multi-agent interactions. For example, the society of minds (SOM) framework [18] deploys multiple LLM instances that independently generate responses to the same query and subsequently engage in structured debate rounds to unify their answers and correct non-factual outputs. However, such approaches are computationally and temporally expensive and thus may not be viable in time-constrained settings such as CLEF 2026.

3. Methodology

We investigate the generative capability of two quantized open-source models, specifically Qwen3 8B [19] (Qwen/Qwen3-8B-AWQ) and Qwen3.5 9B [20] (cyankiwi/Qwen3.5-9B-AWQ-4bit) in a one-shot setting. In both cases, the 4-bit Activation-aware Weight Quantization (AWQ) [21] is used. The selection of the Qwen model family was driven by both empirical performance and deployment constraints. Preliminary architectural reviews and community adoption metrics indicated that these models were leading open-weight candidates for complex reasoning tasks within the 8B–9B parameter class. Furthermore, the availability of 4-bit AWQ quantized versions allowed for seamless integration into our generation pipeline, maximizing efficiency within our hardware limitations (detailed in section 4.3). One approach — submitted during the evaluation cycle and put on the leaderboard — uses the Qwen3 8B model, while two alternative approaches use the Qwen3.5 9B model.

The three approaches share a common two-pass architecture that separates evidence pre-processing from article generation, enabling sequential GPU utilization within the available Video Random Access Memory(VRAM) budget. They differ in (i) how evidence is selected and filtered, (ii) the language model and prompting strategy employed, (iii) the post-processing applied to the generated articles, and (iv) quality control. Figure 1 presents a unified overview of the three pipelines, highlighting the shared architecture and variant-specific components.

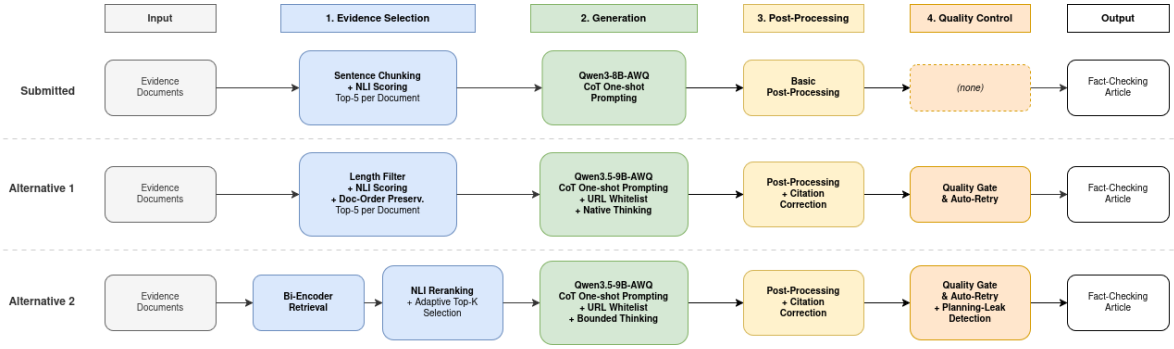

Figure 1: Unified pipeline overview for all three approaches. Each row shows one variant. Shared stages (input, output) are identical across rows; variant-specific components are highlighted within each stage. The Submitted approach uses basic NLI filtering and minimal post-processing; Alternative 1 adds length filtering, URL whitelisting, a quality gate with auto-retry, and citation correction; Alternative 2 replaces NLI-only filtering with a bi-encoder retrieval stage followed by NLI reranking and adds a bounded thinking budget and planning-leak detection.

3.1. Submitted Approach: NLI-Based Evidence Filtering with CoT One-Shot Prompting

The main approach establishes the foundational pipeline: a two-pass architecture that first filters evidence documents using an NLI cross-encoder and then generates fact-checking articles in batch using vLLM. The pipeline for this approach is shown in the first row of Figure 1.

3.1.1. Evidence Pre-Processing

Each claim in the dataset is accompanied by a set of premise article URLs mapped to locally scraped document files. The pre-processing stage loads these documents and applies NLI-based filtering to extract only the passages most relevant to the claim.

Sentence Chunking. Each evidence document is first normalized by collapsing white space, then segmented into sentences using the NLTK Punkt sentence tokenizer. Sentences are grouped into overlapping chunks using a sliding window of size 3 with an overlap of 1 sentence. This chunking strategy ensures that each text unit contains sufficient context for meaningful NLI inference while maintaining granularity for selective filtering. Documents containing three or fewer sentences are treated as a single chunk.

NLI-Based Relevance Scoring. Each chunk is paired with the claim and scored using a RoBERTa large model fine-tuned on the Multi-Genre Natural Language Inference (MNLI) corpus [22] (roberta-large-mnli). The model is loaded in half-precision (FP16) to reduce GPU memory consumption. For each chunk–claim pair, the model produces a probability distribution over three classes: *contradiction*, *neutral*, and *entailment*. We define the relevance score of a chunk as follows:

$$\text{relevance}(c_i) = \max\big(P(\text{contradiction} \mid c_i, \text{claim}),\ P(\text{entailment} \mid c_i, \text{claim})\big) \tag{1}$$

This formulation captures chunks that either *support* or *refute* the claim, as both relationships carry high informational value for fact-checking. Chunks classified predominantly as *neutral* are de-prioritized, as they typically contain boilerplate, navigational, or otherwise uninformative content. In all approaches, the evidence chunk is passed as the premise, and the claim as the hypothesis; this ordering is consistent across the three pipelines and was not tested in reverse. Inference is performed in batches of up to 512 chunk–claim pairs per forward pass. For each document, the top-5 chunks by relevance score are retained and concatenated to form the filtered evidence passage.

3.1.2. Article Generation

For each claim, a two-message one-shot CoT prompt is constructed using the model's chat template. The system message establishes the role of an expert fact-checker and defines the output format: a headline, a detailed analysis section, and a conclusion. The reference format is specified as inline annotations (source: <URL>). A single worked example is provided within the system prompt demonstrating the expected reasoning process and article structure (see Appendix A.1 for the full prompt).

The user message provides the specific claim, claimant, verdict, and filtered evidence formatted with source URL headers. Evidence exceeding 16,000 characters is truncated with a truncation marker to prevent vLLM context-length validation errors.

3.1.3. Post-Processing

Post-processing in this approach is minimal. The raw model output may contain <think>…</think> reasoning blocks even though native thinking is not explicitly turned on (a feature of Qwen3's hybrid thinking mode); these are stripped using regular expression substitution. If the output contains an ARTICLE: delimiter, all content preceding it (including any chain-of-thought reasoning) is discarded, and only the article body is retained.

3.1.4. Limitations

Several limitations of this initial approach motivated subsequent refinements:

1. **Loss of document order.** The top-5 chunk selection sorts purely by relevance score, meaning that the chunks presented to the LLM may be in a different order from the original document, potentially disrupting narrative coherence.
2. **Citation hallucination.** The LLM frequently generates plausible-looking but fabricated URLs drawn from its pretraining data, rather than using the actual evidence source URLs provided in the prompt.
3. **No degenerate output handling.** The pipeline contains no mechanism to detect or retry generations that produce repetitive loops, truncated articles, or empty outputs — it must be addressed manually.
4. **Low-detail prompt.** The one-shot prompt used in this approach provides limited guidance on article structure and citation compliance.

3.2. Alternative Approach 1: Enhanced Pipeline with Citation Control and Evidence Utilization

The first alternative approach addresses each limitation identified in the submitted approach. It introduces document-order preservation in evidence selection, a deterministic citation correction post-processor, explicit handling of degenerate model outputs, and an upgraded language model. The pipeline for this approach is shown in the second row of Figure 1.

3.2.1. Evidence Pre-Processing Improvements

Document-Order Preservation. The NLI scoring procedure is identical to the submitted approach, but the post-selection ordering strategy changes. After selecting the top-5 chunks by relevance score, the selected chunks are re-sorted by their original positional index within the source document. This preserves the narrative flow of the evidence as it appeared in the original article, providing the LLM with a more coherently structured context.

Minimum-Length Filtering and Fallback Mechanism. A pre-filtering step is introduced that discards evidence documents shorter than 200 characters, as these typically correspond to HTML stubs, social media sharing pages, or failed web scrapes that contain no substantive content. Documents below this threshold are placed into a per-claim fallback pool. If NLI filtering eliminates all evidence for a given claim (i.e., no document passes both the length and relevance thresholds), the longest document from the fallback pool is included to guarantee that every claim retains at least one evidence source. This prevents the LLM from generating articles with zero grounding.

3.2.2. Article Generation Improvements

Enhanced Prompt with URL Whitelist. The prompt is substantially restructured (see Appendix A.2). Two key additions are introduced. First, a PERMITTED CITATION URLS block is injected into the user message, presenting an explicit bulleted list of all valid source URLs extracted from the filtered evidence. The system prompt contains strict instructions that the model may only cite URLs appearing in this whitelist, copied character-for-character, and the one-shot example now includes this block.

Added Native Thinking. The chat template is applied with enable_thinking=True, activating the model's native thinking mode. This causes the model to produce its internal reasoning within <think>…</think> tags before generating the visible response. Combined with the explicit THOUGHT PROCESS instructions in the prompt, this creates a dual-layer reasoning mechanism: native model-level thinking followed by prompt-level structured reasoning.

Repetition Penalty. A repetition penalty of 1.1 is applied during sampling to mitigate the tendency of quantized models to enter degenerate repetition loops, where the same phrase or sentence is generated indefinitely.

3.2.3. Post-Processing Pipeline

During batched inference via vLLM, particularly when utilizing the model's native enable_thinking parameter, we identified two critical failure modes: 'thinking loops' and unclosed reasoning tags. The first anomaly occurs when the model enters a degenerate repetition cycle within its internal reasoning block. In these instances, the model repeatedly generates iterative self-correction phrases or stalling tokens without advancing its logical state. This pathological repetition rapidly consumes the designated token budget or the overall context window. Consequently, this triggers the secondary issue: unclosed <think> tags. When the model exhausts its generation limits mid-thought due to these loops, it abruptly halts and fails to emit the requisite </think> closing token. In a standard extraction pipeline, this failure prevents basic string parsing, causing the model's raw, unformatted internal reasoning process to leak directly into the final article payload.

To systematically mitigate these structural generation failures, the post-processing stage in this approach is substantially expanded into a multi-step pipeline:

1. **Think-block stripping.** Both closed <think>…</think> blocks and unclosed <think> tags (where the model ran out of tokens mid-thought) are removed via regex substitution.
2. **Repetition loop removal.** Three categories of degenerate repetitions are detected and stripped: (i) sequences of "Wait" statements repeated three or more times; (ii) single characters repeated 50 or more times; and (iii) short patterns (2–10 characters) repeated 10 or more times.
3. **Article body extraction.** The pipeline attempts to isolate the article body using multiple strategies in sequence: (i) if a THOUGHT PROCESS section appears before the first ## heading, the text preceding the heading is removed; (ii) multiple case-insensitive ARTICLE: markers are checked to split the output; (iii) as a fallback, if the first ## heading appears after 200 characters of preamble, the preamble is discarded. If the model produced multiple rewrites of the article (detected by multiple ## Fact Check headings), only the final version is retained.
4. **Self-review stripping.** The trailing meta-commentary where the model reviews or revises its own output (identified by indented lines beginning with keywords such as "Double Check", "Wait", "Revised", etc.) is removed.
5. **Null-citation removal.** Citations of the form (source: none) are stripped.
6. **Citation correction.** A deterministic post-processor scans every (source: <URL>) citation in the generated article. For each cited URLs, it checks whether the URL exists in the set of valid evidence URLs for that claim. If the URL is hallucinated (not present in the evidence), the post-processor performs word-overlap matching: it extracts the set of lowercase alphanumeric tokens (length ≥ 3) from the sentence preceding the citation and from each evidence document, then substitutes the hallucinated URL with the evidence URL whose content has the highest token overlap with the citing sentence. This heuristic exploits the observation that the LLM typically states facts from the correct evidence source but attributes them to a fabricated URL, meaning that the surrounding sentence text provides a reliable signal for source attribution.

3.2.4. Quality Gate and Auto-Retry

A quality gate is applied after each generation is post-processed. An output is classified as degenerate if it either (i) lacks both a markdown heading (##) and the word "Verdict", or (ii) is shorter than 200 characters. Degenerate outputs are re-queued and regenerated, with up to two automatic retries per claim. Claims that fail all retry attempts receive a placeholder error message "Generation failed or got stuck in a thinking loop." This mechanism addresses tail cases where quantized model inference produces empty or pathologically repetitive outputs.

3.3. Alternative Approach 2: Hybrid Retrieve-and-Rerank with Bi-Encoder Pre-filtering

The second alternative approach replaces the evidence pre-processing pipeline with a two-stage retrieve-and-rerank architecture. Recognizing that the NLI cross-encoder, while precise, processes chunks

sequentially within each document and may miss semantically relevant passages that do not exhibit strong entailment or contradiction signals, we introduce a bi-encoder as a fast semantic retrieval stage prior to NLI reranking. The pipeline for this approach is shown in the third row of Figure 1.

3.3.1. Two-Stage Evidence Selection

Stage 1: Bi-Encoder Semantic Retrieval. Each evidence document is chunked using a slightly larger sliding window (size 4, overlap 1) to provide broader context per chunk. The claim and all chunks are independently encoded into dense vector representations using a Bi-Encoder (all-mpnet-base-v2 via sentence-transformers (SBERT) [14]) that maps text to 768-dimensional embedding vectors. Cosine similarity is computed between the claim embedding and each chunk embedding, and the top-30 chunks by similarity are retrieved. This stage operates in sub-linear time with respect to document length and serves as a fast recall-oriented filter that surfaces semantically related passages regardless of their specific logical relationship (entailment, contradiction, or neutral) to the claim.

Stage 2: NLI Cross-Encoder Reranking. The top-30 candidate chunks from Stage 1 are re-scored using the same RoBERTa/MNLI cross-encoder as in the previous approaches. However, the scoring formula is modified to incorporate a partial neutral signal:

$$\text{score}(c_i) = \text{relevance}(c_i) + 0.2 \times P(\text{neutral} \mid c_i, \text{claim}) \tag{2}$$

The 0.2-weighted neutral component provides a small boost to chunks that are topically related but do not exhibit strong entailment or contradiction, such as contextual background information that is valuable for a comprehensive fact-checking article even though it does not directly confirm or deny the claim.

Minimum-Relevance Threshold. Chunks with a boosted score below 0.3 are discarded entirely. This hard threshold eliminates passages that passed the bi-encoder's broad semantic filter but are judged irrelevant by the more precise NLI model.

Adaptive Top-K Selection. Rather than selecting a fixed number of chunks, the number of retained chunks adapts to the score distribution. A base of $k = 8$ chunks is selected, with the count increased by one for each chunk scoring above 0.7 (indicating high-confidence entailment or contradiction), up to a maximum of 15. This allows claims with rich, high-quality evidence to retain more material, while claims with mostly noisy evidence converge toward the base count. Formally:

$$k_{\text{adaptive}} = \min\Big(k_{\text{base}} + \big|c_i : \text{score}(c_i) > 0.7\big|, |\mathcal{C}_{\text{filtered}}|, 15\Big) \tag{3}$$

where $\mathcal{C}_{\text{filtered}}$ denotes the set of chunks that have passed the minimum-relevance threshold.

As in the first alternative approach, the selected chunks are re-sorted by their original document position before concatenation.

3.3.2. Article Generation

Two modifications are introduced:

- **Structured Prompt with Detailed Example.** The prompt is further refined (see Appendix A.3). The system message is expanded to specify a detailed article structure: headline with verdict, a summary section, a multi-subsection analysis, an optional context section, and a conclusion. The in-context example is replaced with a substantially longer, multi-paragraph article demonstrating this structure, including multi-source citations and a discussion of evidence quality. The user message now explicitly instructs the model to handle edge cases (sparse evidence, noisy content, contradictory sources) and to emit its article body after an explicit ARTICLE: marker.
- **Bounded Thinking Budget.** The chat template's thinking_budget parameter is set to 2,048 tokens, capping the length of the model's internal <think> reasoning. This prevents the model from consuming its entire output budget on deliberation (a failure mode observed in the first alternative approach where some generations spent all tokens in the thinking block, leaving no budget for the article itself).

3.3.3. Quality Gate Additions

The post-processing pipeline is identical to the first alternative approach with one addition:
Planning-Leak Detection. The quality gate is augmented to detect cases where the model emits internal planning notes instead of a finished article. If the first 800 characters of the extracted article body contain any of a set of planning signatures (e.g., "Subsection 1:", "Citation Check:", "Thinking Process:", "Headline: Needs"), the output is classified as degenerate and re-queued for retry. This addresses a failure mode specific to the structured prompt, where the model sometimes interpreted the detailed structural instructions as a template to fill in iteratively rather than as guidance for a single cohesive output.

An additional preamble-stripping pass is also applied: even after splitting on the ARTICLE: marker, if more than 100 characters precede the first ## heading, they are removed. This captures the cases where the model emits residual planning text between the marker and the actual article.

Due to limitations in logging during generation, exact retry counts were not recorded for either alternative approach. However, in Alternative 2, four examples in the test set failed all retry attempts and received the placeholder error message "Generation failed or got stuck in a thinking loop." in the final output. This indicates that the quality gate successfully caught degenerate generations, but that the retry mechanism was insufficient for a small number of particularly problematic claims.

4. Experiments

4.1. Datasets

The task uses two datasets that provide scraped content from the premise articles / evidence web pages where available. Table 1 illustrates the composition of the dataset. The first dataset is WatClaimCheck [23]. It contains training, validation, and test sets. For the purposes of this task, only the training and validation sets from WatClaimCheck are used, and we only use the validation set since we do not train or fine-tune the models, but only perform inference. The second dataset, used for the final evaluation, is provided by the organizers and contains only a test set [6]; part of this test set was curated from the ExClaim dataset [24], which provides evidence-based justifications for real-world claims.

Table 1
Dataset overview for CheckThat! Lab CLEF 2026 Task 3: Generating Full Fact-Checking Articles.

Dataset	Train	Dev	Test	# of evidence articles
WatClaimCheck [23]	**26976**	**3373**	3373	292037
CLEF 2026 CheckThat! Lab Task 3 [6]	-	-	**1158**	9825

Although the ground truth reference articles were available for the validation set, for the test set they were provided only after the evaluation cycle had concluded. Therefore, since no scores were visible during the evaluation cycle — as they would only be calculated after the phase had ended — local evaluation of articles generated using test data was not possible. In addition to the submitted approach, two alternative approaches were evaluated locally after the leaderboard and ground truth articles were made public. The task organizers provided two baseline models: GPT-5-mini [25] was used as the baseline during the validation phase, and K2-V2-Instruct [26] was used as the baseline during the test phase.

4.2. Metrics

For evaluation, the mean of four metrics is used, as specified by the organizers: entailment score, citation recall and precision as proposed by [27], and evidence coverage.

The Entailment Score measures the semantic equivalence between a generated article G and a reference article R using NLI. It captures both faithfulness (is the generated content supported by the

reference?) and coverage (does the generated text cover all reference content?).

First, both texts are split into chunks of sentences. The leaderboard uses chunk size $k = 3$ and overlap $o = 0$. Each text is segmented into ordered, non-overlapping chunks:

$$G = g_1, g_2, \ldots, g_m, \quad R = r_1, r_2, \ldots, r_n \tag{4}$$

where each g_i and r_j consists of up to $k = 3$ consecutive sentences. Sentences are split on sentence-ending punctuation (., !, ?). Next, an NLI model (in this case RoBERTa/MNLI) produces, for each pair of premise(p)–hypothesis(h), a probability of entailment:

$$e(p, h) = P(\text{entailment} \mid p, h) \tag{5}$$

For each generated chunk g_i, the reference chunk r_i that best supports it is found:

$$A = \frac{1}{m} \sum_{i=1}^{m} \max_{j \in 1,\ldots,n} e(r_j, g_i) \tag{6}$$

Here, the premise is the reference chunk r_j, and the hypothesis is the generated chunk g_i. This score measures whether each generated statement is entailed by (i.e., faithful to) some part of the reference.

Symmetrically, for each reference chunk r_j, the generated chunk g_i that best covers it is found:

$$B = \frac{1}{n} \sum_{j=1}^{n} \max_{i \in 1,\ldots,m} e(g_i, r_j) \tag{7}$$

This measures whether every reference statement is covered by some part of the generated text.

The entailment score is then the arithmetic mean of both directions:

$$\text{EntailmentScore}(G, R) = \frac{A + B}{2} \tag{8}$$

Citation Recall measures whether the cited sources for each sentence in the generated article actually support (entail) that sentence. A sentence has recall of 1 if the combined evidence from all its cited sources entails the sentence, and of 0 otherwise.

Let s be a sentence in the generated article, and $C(s) = c_1, c_2, \ldots, c_k$ be the set of URLs cited inline for that sentence. Let $\text{ev}(c)$ denote the evidence text associated with URL c.

The combined premise is the concatenation of all cited evidence:

$$P(s) = \text{ev}(c_1) \oplus \text{ev}(c_2) \oplus \cdots \oplus \text{ev}(c_k) \tag{9}$$

where $\oplus$ denotes string concatenation.

An LLM-based binary NLI judge (in this case, Llama 3.2 1B [28]) determines the entailment:

$$\text{entails}(p, h) = \begin{cases} 1, & \text{if the LLM judges } p \text{ entails } h \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

The per-sentence citation recall is as follows:

$$\text{Recall}(s) = \text{entails}\big(P(s), s\big) \tag{11}$$

Let $S = s_1, s_2, \ldots, s_N$ be all sentences in the generated article that contain at least one citation. The document-level citation recall is as follows:

$$\text{CitationRecall} = \frac{1}{N} \sum_{i=1}^{N} \text{Recall}(s_i) \tag{12}$$

where N is the number of sentences that contain at least one citation.

Citation Precision measures whether every cited source for a sentence is necessary (i.e., no citation is superfluous). A sentence has a precision of 1 only if removing any single cited source causes the entailment to fail.

Given a sentence s with cited URLs $C(s) = c_1, c_2, \ldots, c_k$:

- If $\text{Recall}(s) = 0$, then $\text{Precision}(s) = 0$.
- If $|C(s)| = 1$ and $\text{Recall}(s) = 1$, then $\text{Precision}(s) = 1$.
- If $|C(s)| > 1$ and $\text{Recall}(s) = 1$, then for each citation $c_j \in C(s)$, a leave-one-out premise is constructed by concatenating all evidence except c_j:

$$P_{-j}(s) = \bigoplus_{i \neq j} \text{ev}(c_i) \tag{13}$$

Then:

$$\text{Precision}(s) = \begin{cases} 0, & \text{if } \exists j \in 1, \ldots, k : \text{entails}\big(P_{-j}(s), s\big) = 1 \\ 1, & \text{otherwise} \end{cases} \tag{14}$$

Similarly to recall, document-level citation precision is as follows:

$$\text{CitationPrecision} = \frac{1}{N} \sum_{i=1}^{N} \text{Precision}(s_i) \tag{15}$$

Evidence Coverage measures the proportion of available input evidence documents that were actually cited in the generated article. This is a simple URL-matching metric — it does not judge the quality of how the evidence is used, only whether it was referenced.

Let $E = e_1, e_2, \ldots, e_T$ be the set of all evidence URLs provided as input for a given claim, and let $U \subseteq E$ be the set of evidence URLs that appear in at least one inline citation in the generated article.

$$\text{EvidenceCoverage} = \frac{|U|}{|E|} \tag{16}$$

If $|E| = 0$ (no evidence available), the score is defined as 0.0.

The final score is the unweighted arithmetic mean of all four metrics, which means that each metric contributes equally to the overall ranking:

$$\text{Score} = \frac{1}{4}(\text{EntailmentScore} + \text{CitationPrecision} + \text{CitationRecall} + \text{EvidenceCoverage}) \tag{17}$$

4.3. Experimental Setup

The experiments are conducted using Google Colab with the T4 GPU, which at the time of writing has 15 GB of VRAM and 12.7 GB of Random Access Memory (RAM). This setup allows for the use of relatively small LLMs. Initially, we experimented with models in GGUF format using llama.cpp, which allows for up to 8-bit quantization on the available hardware for Qwen3 8B and Qwen3.5 9B models. However, experiments showed that the generation using this approach was too slow due to its sequential nature. We then switched to using the vLLM library, which allows for parallelized batched inference and supports AWQ. The selected models were available on Hugging Face with 4-bit AWQ quantization, providing sufficient VRAM headroom. Experiments conducted after the main approach showed that enabling native thinking improves the performance of the models. Therefore, while the main approach does not enable thinking, the other alternative approaches have that parameter set to True. All parameters used to run inference with the models are shown in Table 2.

Table 2
Parameters used for LLMs across approaches

Approach Model	Submitted Qwen3 8B AWQ	Alternative 1 Qwen3.5 9B AWQ	Alternative 2 Qwen3.5 9B AWQ
Enable thinking	-	True	True
Temperature	0.2	0.2	0.2
Max tokens	8192	16384	8192
Max model length	32768	32768	32768
Repetition penalty	-	1.1	1.1
Thinking budget	-	-	2048

4.3.1. Optimizing the Evaluation Pipeline

For local evaluation on metrics, outlined in section 4.2, we use code provided by the organizers. Due to hardware and temporal limitations of the Colab environment, the code is not viable to evaluate on full datasets and can only be run on a small subset in a timely manner.

The original pipeline processes examples sequentially, calling the NLI model one example at a time. For large evaluation sets, this results in severe GPU under-utilization. We try to optimize the pipeline to reduce the time needed for evaluation.

Optimization 1 — Mega-Batching Across All Examples
For each example (G_i, R_i), the original chunks both texts, builds $m_i \times n_i$ premise–hypothesis pairs, runs them through the NLI model, then moves to the next example. The GPU sits idle during the Python overhead between examples. Instead of per-example inference, our approach:

1. Pre-chunks all (G_i, R_i) pairs upfront.
2. Concatenates all premise–hypothesis pairs from every example into two giant lists — one for direction A (faithfulness), one for direction B (coverage).
3. Runs a single batched pass per direction through the NLI model.
4. Scatters the results back to per-example score matrices using tracked offsets.

This eliminates virtually all Python/CUDA-launch overhead between the examples. If there are K examples with average $m \times n$ chunk pairs each, the original makes $2K$ model calls, while our version makes exactly 2 mega-calls (one per direction).

Optimization 2 — FP16 Inference
In the original code, the model runs in FP32 (32-bit floating point), using 1.4 GB of VRAM for RoBERTa/MNLI. Instead, on CUDA devices, the model is loaded in FP16 (half precision), and inference runs under torch.amp.autocast("cuda"). This halves memory per parameter and enables Tensor Core acceleration on T4/A100/L4 GPUs, roughly doubling throughput. FP16 inference introduces minor floating-point rounding differences, typically on the order of ± 0.0001–0.001 per individual score. For aggregate means across hundreds of examples, this difference is negligible (well under 0.001) and does not affect the final leaderboard scores, which are rounded to three decimal places.

Optimization 3 — Length-Sorted Batching
In the original code, pairs are batched in their natural order. If short and long sequences are mixed in the same batch, the tokenizer pads everything to the longest sequence, wasting memory on padding tokens. Instead, before batching, all premise–hypothesis pairs are sorted by estimated token length (approximated as $(len(\text{premise}) + len(\text{hypothesis}))/4$). Pairs of similar length are grouped together so each batch pads only to its own longest sequence. After inference, results are un-sorted back to the original order. This significantly reduces peak VRAM usage.

Optimization 4 — Multi-Threaded Citation Evaluation
Citation precision and recall for each example are computed sequentially, — the original script waits for each LLM call to return before starting the next. Since the citation evaluation for each example is independent, our approach uses a ThreadPoolExecutor with configurable parallelism. Multiple examples are evaluated concurrently, with Ollama handling concurrent requests via its internal queue.

This approach to evaluation decreased the time to evaluate the full test set from expected 13 hours to approximately 3 hours in our Colab environment.

4.4. Results and Analysis

We conducted evaluation across the three approaches, outlined in section 3, using data from both validation and test datasets. Initially, we utilized a subset of 20 claims from the validation dataset to generate and evaluate articles. This allowed us to monitor potential improvements during experiments in a timely manner. The specific subset size of 20 samples was selected both to align with the organizers' guidance for the baseline testing and to compare to the validation baseline, as well as to accommodate the computational limitations outlined in section 4.3.1. Consequently, this subset is not intended for a complete performance evaluation, although it does indicate early performance trends. To rigorously evaluate our approaches, the test dataset is utilized in its full capacity. The evaluations of the two alternative approaches on the test set were conducted locally using our optimized pipeline, detailed in section 4.3.1. The evaluation metrics are entailment score, citation recall and precision, and evidence coverage, as outlined in section 4.2.

Table 3 presents the validation results for both the submitted approach and two alternative approaches. Table 4 presents the final test results along with the final rank for the submitted approach and theoretical ranks for alternative approaches.

Table 3
Validation 20 subset metric scores per approach

Approach	Entailment	Precision	Recall	Evidence Coverage	Overall
Baseline (GPT-5-mini [25])	**0.368**	0.239	**0.340**	**0.783**	**0.432**
Alternative 1	0.260	**0.318**	0.325	0.772	0.419
Alternative 2	0.313	0.307	0.324	0.590	0.384
Submitted	*0.299*	*0.000*	*0.000*	*0.050*	*0.087*

Table 4
Final test set metric scores per approach with leaderboard position. For alternative approaches the position is theoretical as they were not submitted during the evaluation cycle and are not on the final leaderboard

Approach	Entailment	Precision	Recall	Evidence Coverage	Overall	Final Rank
Alternative 1	0.232	**0.349**	0.356	**0.738**	**0.419**	**5/6**
Alternative 2	0.278	0.346	**0.360**	0.587	0.393	7
Submitted	*0.227*	*0.287*	*0.295*	*0.440*	*0.312*	*9*
Baseline (K2-V2-Instruct [26])	**0.298**	0.223	0.240	0.329	0.272	10

4.4.1. Submitted Approach — Cross-Encoder Evidence Extraction With CoT One-Shot Prompting

Based on the final test set metrics, our main approach placed 9th on the final leaderboard outperforming the baseline in citation precision and recall, and evidence coverage, but losing on entailment score — something only 3 participants managed to outperform — which results in an improvement of 0.040 on the overall score. Although it achieved a respectable score on the test set, it did not perform as well on the validation set, achieving citation precision and recall of 0.000 and evidence coverage of only 0.050. This may be due to source hallucination when evidence documents contain high amounts of noise, which the WatClaimCheck dataset has more of compared to the test set, as a result of the model not knowing how to use them. This issue is not present in the following approaches.

4.4.2. Alternative Approach 1 — Enhanced Citation Control and Evidence Utilization

Immediate improvement is apparent with the first alternative approach, which would theoretically place 5th or 6th on the leaderboard if it were submitted. This approach achieves the highest citation precision in both the validation subset and the test set as well as the highest evidence coverage in the test set. This results in the highest overall score across the three approaches. In the validation subset, enhanced citation control and evidence utilization fixes the hallucination problem the original approach had and achieves evidence coverage comparable to the baseline, which is a closed-source model with around 10 times the amount of parameters[1] the base Qwen3 8B or Qwen3.5 9B models have without quantization. Although there were considerable improvements in the citation precision and recall, and evidence coverage, the entailment score did not improve on the validation subset compared to the submitted approach and improved only by 0.005 on the test set.

4.4.3. Alternative Approach 2 — Hybrid Retrieve-and-Rerank with Post-Processing

This approach shows improved entailment score, citation precision and recall similar to the previous approach, even outperforming citation recall on the test set by 0.004, but has lower evidence coverage. This would theoretically place seventh on the leaderboard. Overall, this approach shows the most balanced score distribution across all three approaches, but due to a significant evidence coverage loss, it has a lower overall score than the first alternative approach. This approach is also the slowest out of the three. It took 7 to 8 hours to complete, including the time for regeneration of degenerate LLM outputs, compared to 4 hours for the first alternative approach and 2 hours needed for the original submitted approach.

5. Conclusions and Main Findings

In this paper, we presented our three approaches to generating full fact-checking articles using small quantized open-source LLMs in the context of the CLEF 2026 CheckThat! Task 3. Our main approach that was submitted during the evaluation cycle uses NLI-based filtering of evidence documents and CoT one-shot prompting to generate the articles. It outperformed the baseline in citation precision, citation recall, and evidence coverage, but underperformed in the entailment score.

This approach was developed first during the evaluation cycle, with further experiments conducted after the submission was generated. However, due to time constraints and hardware limitations of our chosen Google Colab environment, we were unable to finish generating other approaches in time. This meant that further evaluations were to be done locally after the evaluation cycle had ended and ground truth reference articles were released. One of the main issues with local evaluation was that the code used by the organizers for this task was too slow to fit into the time constraints of our Colab environment (the expected time to evaluate just the entailment score for the full test set was 13 h). Therefore, we developed an optimized approach to evaluation. It uses mega-batching across all examples, FP16 inference instead of FP32, length-sorted batching, and multi-threaded citation evaluation, which makes computation approximately four times faster (approximately 3 hours) compared to the original. We used this to evaluate two alternative approaches. The source code for all three approaches and the optimized evaluation pipeline is publicly available.[2]

The two alternative approaches presented in this paper achieve considerable improvements over the submitted approach. One approach, using enhanced citation control and evidence utilization in comparison to the main approach, achieves one of the highest citation precision and recall (only three participants achieved higher scores on citation precision and only two achieved higher recall) and one of the highest evidence coverage (four participants achieved higher scores on this metric), suggesting

[1] https://apxml.com/models/gpt-5-mini reports 100 billion parameters, while https://www.r-bloggers.com/2025/08/how-many-parameters-does-gpt-5-have/ predicts 85 to 149 billion parameters compared to 8 billion of Qwen3 8B (https://huggingface.co/Qwen/Qwen3-8B) or 9 billion of Qwen3.5 9B (https://huggingface.co/Qwen/Qwen3.5-9B)

[2] https://github.com/Lionel-Logue/CLEF-2026

that whitelisting sources is a strategy worth considering for such tasks and a large enough model might achieve much higher scores on these metrics. However, the entailment score is not much better than the submitted approach. Our second alternative approach, which uses a hybrid retrieve-and-rerank method with bi-encoder pre-filtering, achieves a considerably higher mean entailment score (three participants achieved higher scores on this metric and one achieved the same score) to the detriment of evidence coverage and a slight loss in citation precision and a simultaneous gain in citation recall.

The entailment score was one of the most difficult metrics to achieve high performance on during our experiments, an observation further supported by the fact that only three participants achieved scores higher than the baseline. This could be explained by how that specific metric is calculated. Since it is impossible to fit the reference articles entirely into the model without requiring a large context length and even fitting portions of the reference articles may prove to be difficult (depending on their sizes), the coverage will never be high enough to warrant a high entailment score.

We also faced several notable issues during our participation in the task. The primary challenge was balancing generation speed with output quality. Although we initially wanted to submit the second alternative approach, as it produced better-grounded text with improved entailment scores, the computational requirements prevented timely completion. Additionally, URL hallucination remained a persistent issue across all approaches, requiring either strict prompt engineering or post-processing correction to mitigate. The models also often had repetition issues that would break the article and slow down the generation process until all token limits were reached.

Overall, our work showed that small quantized models with the help of CoT one-shot prompting can achieve respectable results, and that whitelisting permitted URLs with added post-processing alone can achieve performance comparable to large closed-source models such as GPT-5 mini. This allows for building fact-checking solutions that are fully based on local open-source models. Although our work was limited to small quantized models, the approaches used can be applied to larger models to achieve even better results. Future efforts will aim to apply this work to multilingual settings, where fact-checking is most needed.

Acknowledgments

This research was conducted within the framework of the “Development of Tools for Smart City Infrastructure Advancement Considering the Challenges of War and Post-War Recovery in Ukraine” research project (State Budget theme No. 25БП013-01М, Ministry of Education and Science of Ukraine, State Registration No. 0125U001221).

Declaration on Generative AI

During the preparation of this work, the author used Claude Sonnet 4.6 in order to: Grammar and spelling check. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the publication’s content.

References

[1] 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.

[2] X. Zeng, A. S. Abumansour, A. Zubiaga, Automated fact-checking: A survey, Language and Linguistics Compass 15 (2021) e12438. URL: https://compass.onlinelibrary.wiley.com/doi/abs/10.1111/lnc3.12438. doi:10.1111/lnc3.12438. arXiv:https://compass.onlinelibrary.wiley.com/doi/pdf/10.1111/lnc3.12438.

[3] 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/.

[4] I. Eldifrawi, S. Wang, A. Trabelsi, Automated justification production for claim veracity in fact checking: A survey on architectures and approaches, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 6679–6692. URL: https://aclanthology.org/2024.acl-long.361/. doi:10.18653/v1/2024.acl-long.361.

[5] M. Akhtar, et al., Multimodal automated fact-checking: A survey, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 5430–5448. URL: https://aclanthology.org/2023.findings-emnlp.361/. doi:10.18653/v1/2023.findings-emnlp.361.

[6] D. Sahnan, T. Chakraborty, P. Nakov, Overview of the CLEF-2026 CheckThat! lab task 3 on generating full fact-checking articles, in: [9], 2026.

[7] J. M. Struß, et al., The clef-2026 checkthat! lab: Advancing multilingual fact-checking, in: Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2026, pp. 325–335.

[8] J. M. Struß, et al., 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.

[9] 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.

[10] P. Atanasova, et al., Generating fact checking explanations, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), 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.

[11] N. Kotonya, F. Toni, Explainable automated fact-checking for public health claims, 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. 7740–7754. URL: https://aclanthology.org/2020.emnlp-main.623/. doi:10.18653/v1/2020.emnlp-main.623.

[12] D. Sahnan, et al., Can llms automate fact-checking article writing?, 2026. URL: https://arxiv.org/abs/2503.17684. arXiv:2503.17684.

[13] J. Thorne, et al., FEVER: a large-scale dataset for fact extraction and VERification, in: M. Walker, H. Ji, A. Stent (Eds.), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 809–819. URL: https://aclanthology.org/N18-1074/. doi:10.18653/v1/N18-1074.

[14] 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. URL: https://arxiv.org/abs/1908.10084.

[15] V. Karpukhin, et al., Dense passage retrieval for open-domain question answering, 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. 6769–6781. URL: https://aclanthology.org/2020.emnlp-main.550/. doi:10.18653/v1/2020.emnlp-main.550.

[16] J. Wei, et al., Chain-of-thought prompting elicits reasoning in large language models, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2022.

[17] H. Wang, K. Shu, Explainable claim verification via knowledge-grounded reasoning with large language models, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Compu-

tational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 6288–6304. URL: https://aclanthology.org/2023.findings-emnlp.416/. doi:10.18653/v1/2023.findings-emnlp.416.
[18] Y. Du, et al., Improving factuality and reasoning in language models through multiagent debate, 2023. URL: https://arxiv.org/abs/2305.14325. arXiv:2305.14325.
[19] Q. Team, Qwen3 technical report, 2025. URL: https://arxiv.org/abs/2505.09388. arXiv:2505.09388.
[20] Qwen Team, Qwen3.5: Towards native multimodal agents, 2026. URL: https://qwen.ai/blog?id=qwen3.5.
[21] J. Lin, et al., Awq: Activation-aware weight quantization for on-device llm compression and acceleration, in: P. Gibbons, G. Pekhimenko, C. D. Sa (Eds.), Proceedings of Machine Learning and Systems, volume 6, 2024, pp. 87–100. URL: https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.
[22] Y. Liu, et al., Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).
[23] K. Khan, R. Wang, P. Poupart, WatClaimCheck: A new dataset for claim entailment and inference, in: S. Muresan, P. Nakov, A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 1293–1304. URL: https://aclanthology.org/2022.acl-long.92/. doi:10.18653/v1/2022.acl-long.92.
[24] 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_00649.
[25] A. Singh, et al., Openai gpt-5 system card, 2026. URL: https://arxiv.org/abs/2601.03267. arXiv:2601.03267.
[26] K. Team, Z. Liu, et al., K2-v2: A 360-open, reasoning-enhanced llm, 2025. URL: https://arxiv.org/abs/2512.06201. arXiv:2512.06201.
[27] T. Gao, et al., Enabling large language models to generate text with citations, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 6465–6488. URL: https://aclanthology.org/2023.emnlp-main.398/. doi:10.18653/v1/2023.emnlp-main.398.
[28] A. Grattafiori, et al., The llama 3 herd of models, 2024. URL: https://arxiv.org/abs/2407.21783. arXiv:2407.21783.

A. Prompts used for generation

A.1. Submitted Approach

System prompt:

Listing 1: System prompt for the submitted approach

```
You are an expert, meticulous fact-checker. Your task is to write a highly
    precise fact-checking article justifying the verdict for a given claim.
Follow the exact reasoning and output format shown in the example below.

INSTRUCTIONS:
1. Write a full fact-checking article that includes a Headline, a detailed
    Analysis section explaining how the evidence leads to the verdict, and a
    brief Conclusion.
2. You MUST cite your sources. When you state a fact from the evidence, append
    the citation exactly at the end of the sentence in this format: (source: <
    url>).
```

```
3. DO NOT group citations (e.g., avoid (source: <url1>; <url2>)). Write one
    fact per sentence, and cite the single source that provided that fact.
4. DO NOT invent or hallucinate sources or information. Only use the provided
    evidence.

begin EXAMPLE:
Claim: "A California family was ticketed after holding a Viking funeral at a
    local lake."
Claimant: None
Verdict: false

Evidence:
[Source URL: http://www.snopes.com/tag/nc-scooper]
NC Scooper is listed among Snopes' tagged sources. Articles from NC Scooper
    include the Viking funeral story from June 2017, alongside other satirical
    pieces such as claims about the Georgia Guidestones being stolen and Donald
    Trump's health deteriorating.

[Source URL: http://www.snopes.com/2016/01/14/fake-news-sites/]
Snopes' Field Guide to Fake News Sites identifies NC Scooper (Nevada County
    Scooper) as one of the few ostensibly humor-based fake news sites whose
    items floated around social media. Most articles published by NC Scooper
    were clearly satirical in nature and not intended to deceive readers. Among
    NC Scooper's crossover bits were pieces claiming official "gun confiscation
    units" were taking people's weapons and that Sen. Bernie Sanders called for
    "chemtrail reform."

---
THOUGHT PROCESS:
1. The claim states a California family was ticketed for holding a Viking
    funeral at a local lake.
2. The Snopes tag page (snopes.com/tag/nc-scooper) lists this Viking funeral
    story as originating from NC Scooper, a known satirical website.
3. Snopes' Field Guide to Fake News Sites (snopes.com/2016/01/14/fake-news-
    sites/) explicitly classifies NC Scooper as a humor-based fake news site
    whose stories are satirical in nature.
4. Since the story originates entirely from a satirical source with no
    corroborating real news coverage, the verdict "false" is well-supported.
---
ARTICLE:
## Fact Check: Was a Family Cited for a Viking Funeral on a California Lake?

**Verdict: False**

The claim that a California family was ticketed after holding a Viking funeral
    at a local lake is false. The story originated from the Nevada County
    Scooper (NC Scooper), a website that Snopes has tagged as a recurring source
     of fabricated content (source: http://www.snopes.com/tag/nc-scooper).
    Snopes' comprehensive Field Guide to Fake News Sites explicitly identifies
    NC Scooper as "one of the few ostensibly humor-based fake news sites whose
    items floated around social media," noting that most of its articles "were
    clearly satirical in nature and not intended to deceive readers" (source:
    http://www.snopes.com/2016/01/14/fake-news-sites/). The site itself
    describes its content as satire in its own manifesto. No legitimate news
    sources reported on any such incident actually occurring.

In conclusion, this claim is false. The story was a satirical fabrication
    published by NC Scooper, a well-documented fake news and humor site.
end EXAMPLE

Now, apply the same reasoning process to the following real claim:
```

User message:

Listing 2: User message for the submitted approach

```
f"Claim: {p['claim']}\nClaimant: {claimant}\nVerdict: {p['veracity']}\nEvidence
    :\n{formatted_evidence}\n\nProvide your THOUGHT PROCESS and then the ARTICLE
    , exactly as shown in the example."
```

A.2. Alternative Approach 1

System prompt:

Listing 3: System prompt for the first alternative approach

```
You are an expert, meticulous fact-checker. Your task is to write a
    comprehensive, detailed fact-checking article justifying the verdict for a
    given claim.
Follow the exact reasoning and output format shown in the example below.
Write a thorough article that discusses EACH piece of evidence individually.
    The article should be at least 2000 characters long.

CITATION RULES (STRICT - violating these will invalidate your article):
1. You MUST cite your sources inline. When you state a fact from the evidence,
    append the citation at the end of the sentence in this format: (source: <url
    >).
2. You MUST ONLY use URLs that appear in the "PERMITTED CITATION URLS" list
    provided in the user message. Do NOT cite any URL that is not in that list.
    Do NOT invent, paraphrase, or shorten any URL.
3. Copy each URL character-for-character exactly as it appears in the [Source
    URL: ...] tag and in the permitted list.
4. You may cite multiple URLs per sentence using semicolons: (source: <url1>; <
    url2>).
5. Do NOT invent or hallucinate information. Only use facts present in the
    provided evidence blocks.
6. You should cite as many URLs as necessary to support your claims, but do not
     force citations for sources that contain only irrelevant noise or sharing
    links.

begin EXAMPLE:
Claim: "A California family was ticketed after holding a Viking funeral at a
    local lake."
Claimant: None
Verdict: false

PERMITTED CITATION URLS:
- http://www.snopes.com/tag/nc-scooper
- http://www.snopes.com/2016/01/14/fake-news-sites/

Evidence:
[Source URL: http://www.snopes.com/tag/nc-scooper]
NC Scooper is listed among Snopes' tagged sources. Articles from NC Scooper
    include the Viking funeral story from June 2017, alongside other satirical
    pieces such as claims about the Georgia Guidestones being stolen and Donald
    Trump's health deteriorating.

[Source URL: http://www.snopes.com/2016/01/14/fake-news-sites/]
Snopes' Field Guide to Fake News Sites identifies NC Scooper (Nevada County
    Scooper) as one of the few ostensibly humor-based fake news sites whose
    items floated around social media. Most articles published by NC Scooper
    were clearly satirical in nature and not intended to deceive readers. Among
    NC Scooper's crossover bits were pieces claiming official "gun confiscation
    units" were taking people's weapons and that Sen. Bernie Sanders called for
    "chemtrail reform."
```

```
---
THOUGHT PROCESS:
1. The claim states a California family was ticketed for holding a Viking
    funeral at a local lake.
2. The Snopes tag page (snopes.com/tag/nc-scooper) lists this Viking funeral
    story as originating from NC Scooper, a known satirical website.
3. Snopes' Field Guide to Fake News Sites (snopes.com/2016/01/14/fake-news-
    sites/) explicitly classifies NC Scooper as a humor-based fake news site
    whose stories are satirical in nature.
4. Since the story originates entirely from a satirical source with no
    corroborating real news coverage, the verdict "false" is well-supported.
---
ARTICLE:
## Fact Check: Was a Family Cited for a Viking Funeral on a California Lake?

**Verdict: False**
The claim that a California family was ticketed after holding a Viking funeral
    at a local lake is false. The story originated from the Nevada County
    Scooper (NC Scooper), a website that Snopes has tagged as a recurring source
     of fabricated content (source: http://www.snopes.com/tag/nc-scooper).
    Snopes' comprehensive Field Guide to Fake News Sites explicitly identifies
    NC Scooper as "one of the few ostensibly humor-based fake news sites whose
    items floated around social media," noting that most of its articles "were
    clearly satirical in nature and not intended to deceive readers" (source:
    http://www.snopes.com/2016/01/14/fake-news-sites/). The site itself
    describes its content as satire in its own manifesto. No legitimate news
    sources reported on any such incident actually occurring.
In conclusion, this claim is false. The story was a satirical fabrication
    published by NC Scooper, a well-documented fake news and humor site.
end EXAMPLE

Now, apply the same reasoning process to the following real claim:
```

User message:

Listing 4: User message for the first alternative approach

```
f"Claim: {p['claim']}\n"
f"Claimant: {claimant}\n"
f"Verdict: {p['veracity']}\n\n"
f"PERMITTED CITATION URLS (you may ONLY cite these exact URLs, copied character
    -for-character):\n"
f"{valid_urls_list}\n\n"
f"Evidence:\n{formatted_evidence}\n\n"
f"Provide your THOUGHT PROCESS and then the ARTICLE, exactly as shown in the
    example. "
f"Every factual sentence MUST end with (source: <one of the permitted URLs
    above>). "
f"Do NOT cite any URL not in the permitted list above. "
f"Use the permitted URLs to support your analysis. You do not need to cite
    every single URL if some provide only irrelevant noise, but cite as many as
    necessary to build a comprehensive factual argument. "
f"Write a comprehensive article of at least 2000 characters."
```

A.3. Alternative Approach 2

System prompt:

Listing 5: System prompt for the second alternative approach

```
You are an expert investigative journalist and fact-checker writing for a major
     fact-checking organization. Your task is to write a comprehensive,
    authoritative fact-checking article that thoroughly examines a claim and
    justifies the verdict.

ARTICLE STRUCTURE REQUIREMENTS:
1. Start with a clear headline that includes the verdict (e.g., "Claim is FALSE
     - evidence shows...")
2. Include a Summary section that gives a 2-3 sentence overview
3. Provide Detailed Analysis with multiple subsections examining different
    aspects of the claim
4. Include Context section if relevant (historical background, why the claim
    spread, etc.)
5. End with a clear Conclusion that restates the verdict and main supporting
    points
6. The article should be highly comprehensive and ensure every single piece of
    evidence is discussed in detail.

CITATION RULES (STRICT - violating these will invalidate your article):
1. You MUST cite sources inline for every factual claim using: (source: <url>)
2. ONLY use URLs from the "PERMITTED CITATION URLS" list - do NOT invent or
    modify URLs
3. Copy URLs exactly character-for-character as they appear in the evidence
    blocks
4. For multiple sources in one sentence use: (source: <url1>; <url2>) with
    different URLs
5. Do NOT cite sources that only contain noise, social media sharing buttons,
    or irrelevant content
6. Do NOT make up facts - only state what is explicitly supported by the
    evidence provided

HANDLING EVIDENCE:
- If evidence is sparse or partially relevant: Focus on what CAN be verified
    and note limitations
- If evidence is noisy (ads, navigation, irrelevant text): Extract and cite
    only the substantive factual content
- If evidence contradicts itself: Present both sides and explain which is more
    credible and why
- If no useful evidence for a URL: Simply don't cite it - never write (source:
    none) or similar

begin EXAMPLE:
Claim: "Austria bans Muslims from engaging in politics"
Claimant: Social media users
Verdict: misleading

PERMITTED CITATION URLS:
- https://www.aljazeera.com/news/2020/11/11/austria-to-introduce-preventive-
    detention-after-deadly-attack
- https://www.barrons.com/news/austria-to-introduce-preventive-detention-for-
    terror-convicts-01605109204
- https://www.barrons.com/news/vienna-gunman-had-macedonian-roots-tried-to-go-
    to-syria-minister-01604398810
- https://perma.cc/KXR2-MRC5?type=image
- https://perma.cc/B725-X8ED?type=image
- https://perma.cc/Y2PF-6HHT?type=image
- https://perma.cc/FDS3-8FKD?type=image

Evidence:
[Source URL: https://www.aljazeera.com/news/2020/11/11/austria-to-introduce-
    preventive-detention-after-deadly-attack]
Austria announced plans to create a criminal offence of "political Islam" and
    introduce preventive detention for convicted terrorists. Chancellor
    Sebastian Kurz said the measures target people preparing the ground for
```

terrorism. The proposals include simplifying procedures to shut down associations or mosques deemed involved in radicalisation, creating a central register of imams, and introducing preventive detention or electronic monitoring for convicted terrorists judged not fully deradicalised. The measures also include stripping dual citizens convicted of terrorism of Austrian citizenship and seizing assets of groups tied to extremism.

[Source URL: https://www.barrons.com/news/austria-to-introduce-preventive-detention-for-terror-convicts-01605109204]
Austria will introduce preventive detention for convicted terrorists and create criminal offenses for "political Islam" and religiously motivated extremism. The measures came after a deadly terror attack in Vienna. The government plans to bring proposals to parliament for shutting down associations or mosques playing a role in radicalisation and creating a register of imams.

[Source URL: https://www.barrons.com/news/vienna-gunman-had-macedonian-roots-tried-to-go-to-syria-minister-01604398810]
The Vienna attacker had prior conviction for attempting to join ISIL in Syria. The attack prompted government raids on associations with alleged links to extremist groups and announcement of tougher counter-terrorism measures.

[Source URL: https://perma.cc/KXR2-MRC5?type=image]
Archived social media post showing circulation of claim that Austria banned Muslims from politics.

[Source URL: https://perma.cc/B725-X8ED?type=image]
Archived social media post with similar claim about Austria banning Muslims from political participation.

[Source URL: https://perma.cc/Y2PF-6HHT?type=image]
Additional archived example of the viral claim spreading on social media.

[Source URL: https://perma.cc/FDS3-8FKD?type=image]
Further archived social media post containing the misleading claim.

ARTICLE:
Fact Check: Did Austria ban Muslims from engaging in politics?

Verdict: Misleading

Summary
Social media posts claimed Austria banned Muslims from engaging in politics, with multiple archived examples of these posts circulating online (source: https://perma.cc/KXR2-MRC5?type=image; https://perma.cc/B725-X8ED?type=image; https://perma.cc/Y2PF-6HHT?type=image; https://perma.cc/FDS3-8FKD?type=image). In reality, following a deadly November 2020 attack in Vienna, the Austrian government proposed counter-terrorism measures targeting what it called "political Islam" and organizations linked to radicalisation – not a blanket ban on Muslims participating in politics. The proposals focused on criminalizing certain ideology-driven activities and did not prohibit Muslims from voting, running for office, or joining political parties (source: https://www.aljazeera.com/news/2020/11/11/austria-to-introduce-preventive-detention-after-deadly-attack; https://www.barrons.com/news/austria-to-introduce-preventive-detention-for-terror-convicts-01605109204).

What the government actually proposed
Chancellor Sebastian Kurz announced the government planned to create a criminal offence of "political Islam" to enable authorities to act against people viewed as preparing the ground for terrorism, even if they are not themselves directly involved in terrorist acts (source: https://www.aljazeera.com/news/2020/11/11/austria-to-introduce-preventive-detention-after-deadly-attack). The proposals included simplifying legal procedures for shutting down associations or mosques deemed to play a role in

```
    radicalisation, creating a central register of imams, and introducing
    preventive detention or electronic monitoring for people convicted of terror
     offences who are judged not fully deradicalised (source: https://www.
    aljazeera.com/news/2020/11/11/austria-to-introduce-preventive-detention-
    after-deadly-attack; https://www.barrons.com/news/austria-to-introduce-
    preventive-detention-for-terror-convicts-01605109204).

The measures also included proposals to strip dual citizens convicted of
    terrorism of Austrian citizenship in certain cases and to seize assets of
    groups alleged to be tied to extremism (source: https://www.aljazeera.com/
    news/2020/11/11/austria-to-introduce-preventive-detention-after-deadly-
    attack). Importantly, these were proposals to be brought before parliament
    for debate and voting, not immediately enacted laws (source: https://www.
    barrons.com/news/austria-to-introduce-preventive-detention-for-terror-
    convicts-01605109204).

What the proposals did NOT do
None of the credible news reports indicate Austria introduced any law
    preventing Muslims as a religious group from voting, standing for office,
    joining political parties, or otherwise participating in democratic politics
    . The government's language and news coverage focused specifically on
    targeting an ideological current described as "political Islam" and on
    measures aimed at associations or individuals linked to radicalisation or
    terror offences - not on barring people from political participation based
    solely on their religious identity (source: https://www.aljazeera.com/news
    /2020/11/11/austria-to-introduce-preventive-detention-after-deadly-attack;
    https://www.barrons.com/news/austria-to-introduce-preventive-detention-for-
    terror-convicts-01605109204).

Context and background
The government package came in response to a terror attack in Vienna in
    November 2020, in which the attacker had a prior conviction for attempting
    to join ISIL in Syria (source: https://www.barrons.com/news/vienna-gunman-
    had-macedonian-roots-tried-to-go-to-syria-minister-01604398810). The attack
    prompted government raids on associations accused of links to extremist
    groups and triggered announcement of tougher counter-terrorism measures
    aimed at preventing radicalisation and future attacks (source: https://www.
    barrons.com/news/vienna-gunman-had-macedonian-roots-tried-to-go-to-syria-
    minister-01604398810; https://www.aljazeera.com/news/2020/11/11/austria-to-
    introduce-preventive-detention-after-deadly-attack).

Conclusion - Final verdict: Misleading
The Austrian government announced proposals targeting "political Islam,"
    extremist associations, and individuals linked to radicalisation and
    terrorism - including measures to close certain mosques or associations,
    establish a register of imams, and enable preventive detention for convicted
     terrorists - but it did not impose any general ban on Muslims participating
     in democratic politics (source: https://www.aljazeera.com/news/2020/11/11/
    austria-to-introduce-preventive-detention-after-deadly-attack; https://www.
    barrons.com/news/austria-to-introduce-preventive-detention-for-terror-
    convicts-01605109204). Characterising these targeted counter-terrorism
    proposals as "banning Muslims from engaging in politics" fundamentally
    misrepresents their stated scope and intent, making the viral claim
    misleading (source: https://www.aljazeera.com/news/2020/11/11/austria-to-
    introduce-preventive-detention-after-deadly-attack).
end EXAMPLE

Now, apply this same thorough, structured approach to the following claim:
```

User message:

Listing 6: User message for the second alternative approach

```
    f"Claim: {p['claim']}\n"
    f"Claimant: {claimant}\n"
    f"Verdict: {p['veracity']}\n\n"
    f"PERMITTED CITATION URLS (you may ONLY cite these exact URLs, copied character
        -for-character):\n"
    f"{valid_urls_list}\n\n"
    f"Evidence:\n{formatted_evidence}\n\n"
    f"Write a comprehensive fact-checking article strictly following the structure
        shown in the example:\n"
    f"1. Headline with claim\n"
    f"2. Verdict\n"
    f"3. Summary (2-3 sentences)\n"
    f"4. Detailed Analysis with subsections examining different aspects\n"
    f"5. Context section (if relevant)\n"
    f"6. Conclusion with final verdict and key points\n\n"
    f"CRITICAL REQUIREMENTS:\n"
    f"- Every factual sentence MUST be followed by inline citations: (source: <url
        >)\n"
    f"- ONLY cite URLs from the permitted list above - copy them exactly as shown\n
        "
    f"- If evidence is sparse or noisy, work with what you have and acknowledge
        limitations\n"
    f"- If evidence contradicts itself, present both sides and explain which is
        more credible\n"
    f"- Do NOT cite URLs that only contain irrelevant noise, ads, or navigation
        elements\n"
    f"- Do NOT invent facts not present in the evidence\n"
    f"- Be highly comprehensive and ensure every single piece of evidence is
        discussed in detail.\n\n"
    f"Think carefully before writing the final article.\n"
    f"Output your final fact-checking article starting exactly with the marker \"
        ARTICLE:\" followed by a newline. \n"
    f"Begin your article with a headline starting with \"## Fact Check:\"."
```