OpenFact at CheckThat! 2025: Application of Self-Reflecting and Reasoning LLMs for Fact-Checking Claim Normalization

Notebook for the CheckThat! Lab at CLEF 2025

Marcin Sawiński[1,*], Krzysztof Węcel[1] and Ewelina Księżniak[1]

[1]Department of Information Systems, Poznań University of Economics and Business, Al. Niepodległości 10, 61-875 Poznań, Poland

Abstract

This paper presents a system for the claim normalization task, developed for the CLEF 2025 CheckThat! Task 2 competition. We evaluated large language models of approximately 8B parameters, including *LLaMA 3.1*, *DeepSeek-R1*, and *GPT-4.1-mini*, using the METEOR score as a primary metric. *GPT-4.1-mini* with supervised fine-tuning emerged as the best-performing approach, ranking second or third on six out of seven languages offered in zero-shot setting. Our study also explores self-reflection and multiple candidate selection techniques, finding that while self-reflection did not improve METEOR scores, it helped reduce factual errors. These insights highlight the need to balance metric-driven evaluation with qualitative analysis for effective claim normalization in real-world scenarios.

Keywords

check-worthiness, fact-checking, fake news detection, language models, claim normalization, LLM,

1. Introduction

The task of claim normalization involves transforming unstructured textual content, such as social media posts, into concise and factually accurate claims suitable for downstream fact-checking [1].

Large language models (LLMs) have demonstrated strong potential for generative and summarization tasks, suggesting their suitability for claim normalization. However, evaluating claim normalization outputs is complex, as metrics such as METEOR often prioritize stylistic similarity over factual correctness. This discrepancy is particularly important for fact-checking applications, where factual inaccuracies can lead to misleading interpretations.

This paper presents a system for claim normalization developed for the CLEF 2025 CheckThat! Task 2 competition. We evaluated decoder-only LLMs, including *LLaMA 3.1*, *DeepSeek-R1*, and *GPT-4.1-mini*, and explored techniques such as self-reflection, multiple candidate generation, and supervised fine-tuning.

We posed the following research question to examine the application of LLMs to the claim normalization task:

- **RQ1**: How well are different LLMs suited for the claim normalization task?
- **RQ2**: Do reasoning fine-tuned models outperform base chat models?
- **RQ3**: How does self-reflection impact the LLM performance on the claim normalization task?
- **RQ4**: How does multiple candidate generation and selection impact performance on the claim normalization task?

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain

*Corresponding author.

†These authors contributed equally.

marcin.sawinski@ue.poznan.pl (M. Sawiński); krzysztof.wecel@ue.poznan.pl (K. Węcel); ewelina.ksiezniak@ue.poznan.pl (E. Księżniak)

https://kie.ue.poznan.pl/en/ (M. Sawiński)

0000-0002-1226-4850 (M. Sawiński); 0000-0001-5641-3160 (K. Węcel); 0000-0003-1953-8014 (E. Księżniak)

- **RQ5**: Can supervised fine-tuning of LLMs improve performance on the claim normalization task?

Our findings show that while self-reflection did not consistently improve METEOR scores, it helped reduce factual errors in generated claims. Fine-tuning the *GPT-4.1-mini* model further improved performance, demonstrating the importance of task-specific adaptation for robust claim normalization.

2. Related Work

Sundriyal et al. [1] proposed the Check-worthiness Aware Claim Normalization (CACN) approach, which leverages chain-of-thought prompting and reverse check-worthiness. The goal of chain-of-thought prompting is to decompose a complex task into a sequence of simpler subtasks, using examples to encourage step-by-step reasoning. Reverse check-worthiness, on the other hand, prioritizes claims that meet the criteria for check-worthiness.

Our study seeks to build upon the research of Sundriyal et al. [1] by leveraging newer, more capable LLMs and examining the impact of increased test-time compute on system performance. While advancements in foundational models remain a primary focus within the field, it is widely acknowledged that comprehensive systems built around LLMs can yield even greater improvements. We investigated the use of reasoning-tuned models and self-reflection as potential methods to enhance performance.

Reasoning-tuned models have gained significant attention by topping multiple leaderboards. However, it has been observed that their performance varies depending on the complexity of the task, and in some cases, tasks are better addressed by models without reasoning fine-tuning. A recent study [2] highlights that reasoning models often struggle to adapt generation length to task complexity, with both overly short and excessively long generations leading to degraded performance. We decided to run a direct comparison of the same model architecture with and without reasoning-tuning.

Automated methods to auto-correct large language models (LLMs) aim to improve their output without human intervention. These approaches include self-correction, where the LLM refines its own outputs using self-generated feedback iteratively; generation-time correction, where LLMs adjust outputs during generation guided by feedback from critic models or external knowledge sources; and post-hoc correction, where outputs are refined after generation, leveraging external tools, knowledge bases, or multi-agent debates. These strategies address errors like hallucinations, unfaithful reasoning, and toxic content, offering flexible and scalable ways to enhance LLM performance autonomously [3].

Self-refine [4], an iterative refinement approach use the same LLM to generate an initial answer, then provides feedback on its own answer, and finally refines the answer using that feedback[4]. Iterative, self-correcting loop without external supervision, additional training, or reinforcement learning could be applied across diverse tasks, especially with complex or nuanced quality criteria.

Self-correction approach has been also criticized indicating that LLMs often fail to correct their responses without external feedback, and that self-correction can even degrade the performance. [5]

3. Datasets

The dataset provided by the organizers comprised three splits: *train*, *dev*, and *test*. The *train* and *dev* splits covered 13 languages, while the *test* split included 20 languages (see Table 1). The *train* and *dev* splits contained both *post* and *normalized claim* columns, whereas the *test* split included only the *post* column.

An initial quality check of the *train* and *dev* splits revealed several issues that could negatively impact training and evaluation:

- **Language mismatch**: the *post* and *normalized claim* were in different languages.
- **Referenced media**: the *post* text referred to external media (e.g., images or videos) that were not available but necessary to formulate the claim.
- **Content mismatch**: the *claim* text referenced facts not present in the *post* text.

Table 1
Dataset split sizes per language before and after cleanup.

Language	train	dev	test	train cleaned	dev cleaned
eng	11374	1171	1285	3311	376
spa	3458	439	439	1128	146
por	1735	223	225	592	66
fra	1174	147	148	414	53
hi	1081	50	100	351	17
msa	540	137	100	93	20
ara	470	118	100	233	61
pa	445	50	100	179	15
deu	386	101	100	104	32
tha	244	61	100	92	24
pol	163	41	100	39	12
mr	137	50	100	50	21
ta	102	50	100	51	25
bn	0	0	81	0	0
ces	0	0	123	0	0
ell	0	0	156	0	0
kor	0	0	274	0	0
nld	0	0	177	0	0
ron	0	0	141	0	0
te	0	0	116	0	0

- **Multiple claims**: the *post* text contained multiple possible claims or detailed elements, while the *normalized claim* arbitrarily selected only one relevant detail for fact-checking.

To address these issues, we first filtered out examples with language mismatches. We then used *gpt-4.1-mini* to flag semantic mismatches between the *post* and *normalized claim*, identify external media references, and detect if the *normalized claim* contained information that was absent in the *post*.

Examples flagged in this process or excluded due to content policy violations (e.g., hate speech, jailbreak, self-harm, sexual or violent content) were removed from further processing.

Examples containing multiple or highly complex claims were retained. Overall, approximately one third of the provided examples were used for training and evaluation, as shown in Table 1.

4. Data processing pipelines

We modeled the system as LLM-based self-reflection agent that iteratively improve outputs until no further improvements are observed. We also introduced multiple initial candidate generations by adjusting temperature or seed settings, depending on the model. This step was motivated by the observation that in some cases, the initial claim phrasing anchored the model to specific details of the *post* throughout subsequent iterations.

The complete claim normalization process consisted of three steps:

- **Initial claim extraction**: generation of up to three claim candidates using a prompt with guidelines for claim normalization.
- **Improvement via self-reflection**: iterative refinement of each claim candidate through multiple self-reflection steps. The process was capped at a maximum number of steps but stopped earlier if no changes were detected compared to the previous iteration.
- **Selection with LLM-as-a-judge**: the model was presented with up to three improved claim candidates and tasked with selecting the best one.

Results were collected at three pipeline steps to measure the impact of different techniques:

- **Initial claim extraction (Ini)** — Baseline results obtained by generating a normalized claim with a single call to the LLM for each *post*. This score reflects the basic output quality of the LLMs without any additional techniques.
- **Self-reflection (Ref)** — Results achieved by applying the self-reflection technique to previously generated outputs. In this step, the model received the *post* text, the *normalized claim* from initial extraction or the previous iteration, and the claim normalization guidelines. The prompt instructed the model to refine the *normalized claim* to best match the guidelines. This process was repeated up to ten times for *Llama* and *DeepSeek* and up to five times for *GPT* and *GPT FT*, or until no further changes were made. All iterations were run with temperature set to zero or the same seed to ensure determinism.
- **Candidate selection (Sel)** — For *Llama* and *DeepSeek*, the initial claim extraction step was repeated three times with different random seeds, resulting in varied claim formulations. The second and third outputs were not reported in the previous steps. All three outputs were independently refined using the self-reflection loop described above. Finally, the LLM was presented with the three improved candidates and tasked to select the best one that matched the guidelines. For *GPT* and *GPT FT*, the variability in initial outputs was very low with temperature set below 1, and the self-reflection step with temperature zero consistently produced the same claim phrasing. As a result, this step was omitted for the *GPT* models.

Figure 1 shows claim normalization pipeline.

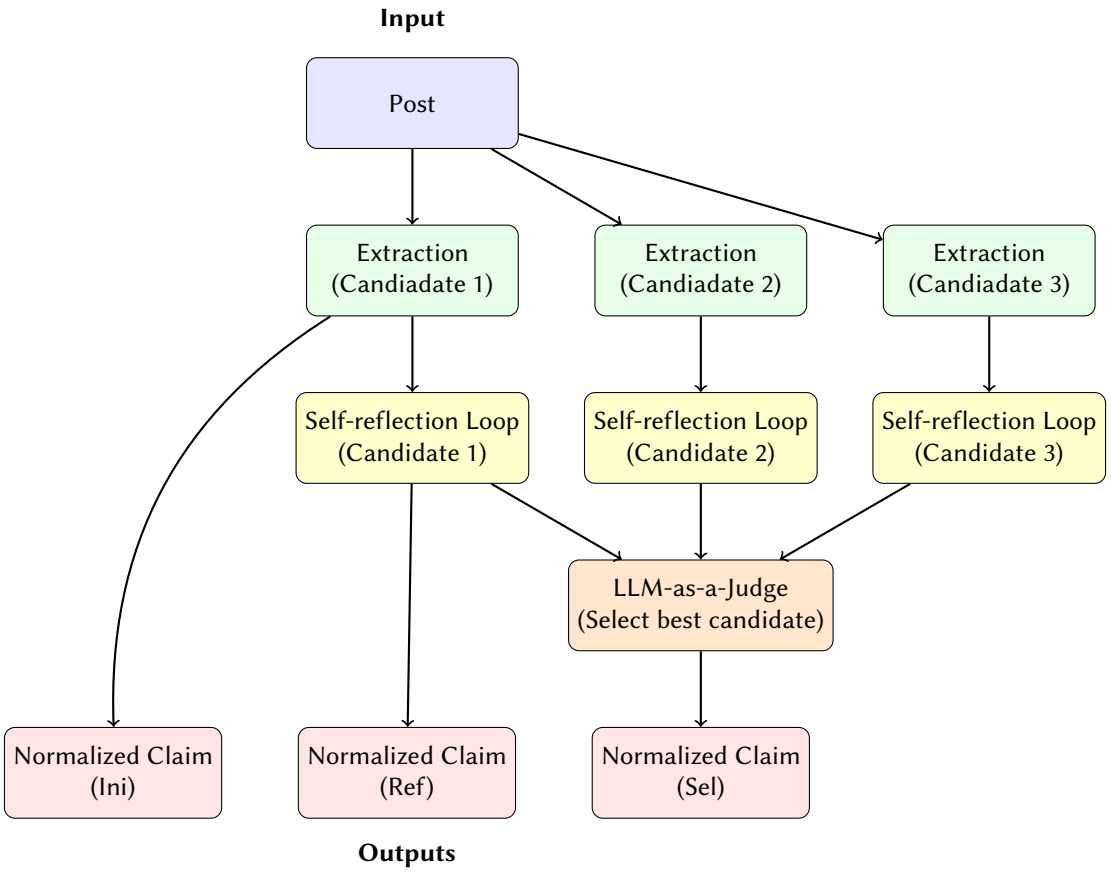

Figure 1: Claim normalization pipeline with candidate generation, self-reflection, and final selection.

The LLMs were queried in chat mode, using message chains composed of a *system* and a *user* message. Identical guidelines for claim normalization were used across all three processing steps.

We applied supervised fine-tuning using the *train* datasets for all languages, converting them into message chains that included *system* and *user* parts, combined with the *assistant* part derived from the *normalized claim* column.

5. Models

We limited our study to LLMs with approximately 8B parameters. Larger models were excluded due to hardware constraints, longer inference times, and prohibitive costs, given the large-scale nature of the claim normalization task. Smaller models were excluded because of low performance observed in preliminary experiments.

The models selected for this study were: **LLaMA 3.1 8B**, **DeepSeek-R1 8B** (a reasoning fine-tuned version of LLaMA 3.1 8B), and **GPT-4.1-mini**. While the specifications of GPT-4.1-mini are not publicly available, we assumed its parameter count to be between 7B and 9B, making it a fair comparison to the other models. LLaMA and DeepSeek models were run using 4-bit grouped quantization with the Ollama backend. The hardware consisted of four NVIDIA GeForce RTX 2080 Ti GPUs, yielding a generation speed of approximately 300 tokens per second.

GPT-4.1-mini was used in its default configuration and also fine-tuned for this task using Azure OpenAI services with the following hyperparameters: number of epochs 3, batch size 4 and LR multiplier 2.

In the experiments section, we refer to the models as follows:

- LLaMA 3.1 8B — **Llama**
- DeepSeek-R1 8B — **DeepSeek**
- GPT-4.1-mini — **GPT**
- GPT-4.1-mini fine-tuned on the *train* dataset — **GPT FT**

6. Experimental Results

6.1. METEOR score results analysis

All models and techniques were evaluated using the METEOR score on *dev* dataset split. For readability, all metrics in Table 2 are presented as percentages.

Results are reported per pipeline step to highlight the impact of different techniques.

6.1.1. Initial Claim Extraction

The *Llama* model achieved a METEOR score of 24 across all languages using only the initial claim generation step. The *DeepSeek* model yielded a lower METEOR score of 16. In contrast, the *GPT* model achieved a METEOR score of 42 with the same approach.

A language-wise breakdown showed the same trend: *GPT* consistently outperformed *Llama*, while *Llama* outperformed *DeepSeek* in every language within the *dev* dataset.

Although the *DeepSeek* model produced promising "thinking" traces—iterating over most of the guideline items to generate normalized claims—this was not reflected in the final scores. This discrepancy may be partly attributed to *DeepSeek*'s lower adherence to instructions and the addition of extraneous text to the final output (e.g., "The claim extracted from the post is: …").

6.1.2. Self-reflection

The METEOR scores for outputs processed through self-reflection loops were not consistently higher than those of the initially generated outputs. While specific language-model intersections showed some improvement, no clear overall trend or consistent gains were observed.

6.1.3. Candidate Selection Results

Generating multiple initial outputs and selecting the best candidate using the LLM as a judge did not lead to improvements in METEOR scores.

Table 2
METEOR scores by model and language at each pipeline step for *dev* dataset split. Abbreviations: Ini = initial claim extraction, Ref = self-reflection, Sel = candidate selection.

Model	METEOR									
	Llama			DeepSeek			GPT		GPT FT	
Step	Ini	Ref	Sel	Ini	Ref	Sel	Ini	Ref	Ini	Ref
Language										
ALL	24	24	25	17	16	16	42	41	58	58
ara	14	12	14	9	7	5	47	44	61	61
deu	20	20	23	14	14	14	28	28	38	38
eng	40	40	42	43	42	44	48	49	60	59
fra	30	34	32	22	22	26	49	47	54	54
hi	24	22	27	14	13	8	43	40	58	52
mr	47	48	45	26	20	20	68	67	86	86
msa	28	29	28	9	10	8	27	27	53	53
pa	14	18	22	9	10	13	45	43	90	90
pol	23	21	18	14	12	10	40	41	55	55
por	26	24	28	24	24	27	47	47	59	59
spa	26	25	27	20	21	20	46	47	50	50
ta	18	17	20	11	8	6	47	47	83	77
tha	5	5	4	3	3	2	6	7	14	14

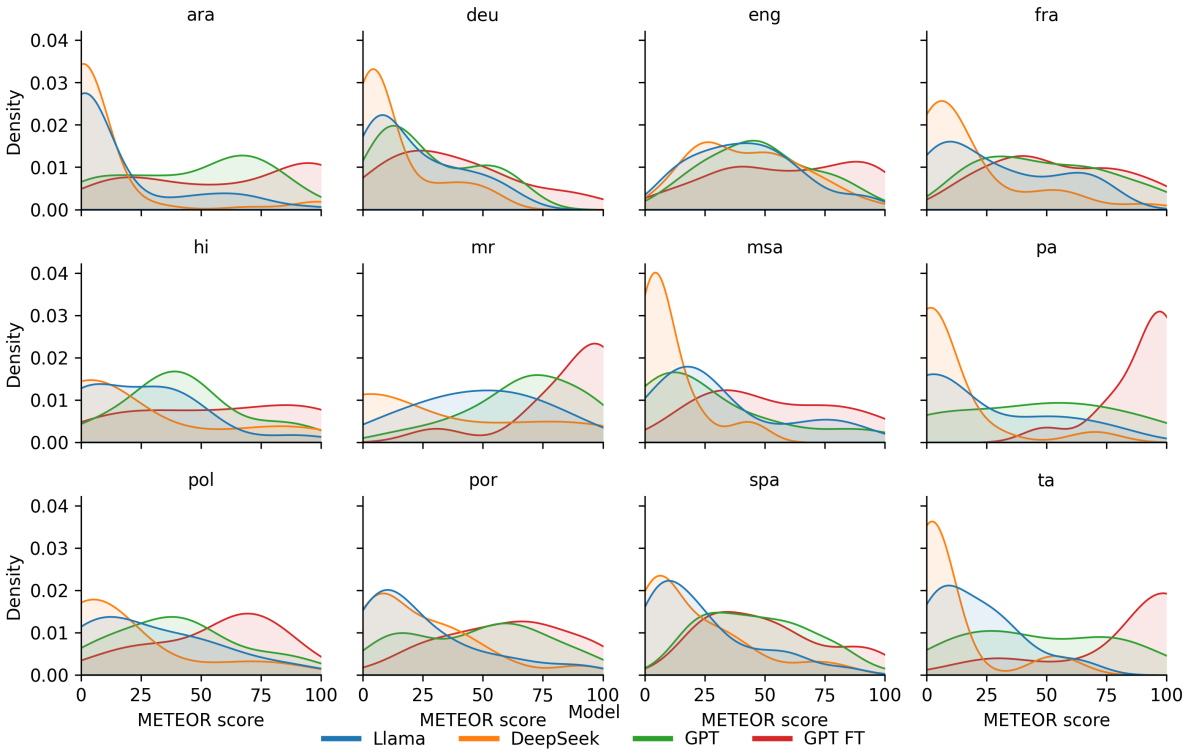

Figure 2: KDE plots of METEOR score calculated for initial claim extraction across languages and models for *dev* datasets split.

6.1.4. Supervised Fine-tuning

Fine-tuning the *GPT* model on the *train* dataset improved the METEOR score by 16 across all languages. This approach proved to be the most effective adaptation technique tested in the study.

Table 3
Illustration of impact of factual errors and phrasing on METEOR score.

Reference: Dandelion root is able to kill 98% of cancer cells within 48 hours	
Candidates:	METEOR
1.Dandelion is able to grow 98% of cancer cells within 4 hours	84
2.Dandelion is able to kill 1% of cancer cells within 4 hours"	84
3.Dandelion kills 98% of cancer cells within 2 days	63
4.Dandelion kills nearly all cancer cells in 2 days	25

6.2. Additional manual analysis

Scoring claim normalization results presents several challenges. First, the METEOR score does not adequately recognize semantically equivalent sentences and tends to prioritize stylistic features over factual accuracy. As shown in Table 3, incorrect claims 1 and 2—which reverse the meaning or alter numerical values—achieve high scores of 84, while correct claims 3 and 4, which use alternative phrasings, score lower at 63 and 25, respectively.

Analysis of the results indicates that high METEOR scores (above 8) are only observed when the *post* is short and contains a well-formulated claim that can be extracted verbatim. For more complex examples, exact phrasing alignment with the annotator's version is nearly impossible, leading to significantly lower scores. In these cases, generated claims often contain substantial factual errors or omissions that the METEOR metric fails to capture.

Second, there is no established gold standard for claim normalization. With multiple annotation authors and styles, it is often impossible to determine whether the reference or the generated claim is more appropriate when both are factually correct but differ in the level of detail. This situation results in low METEOR scores, even though both reference and generated claims may be accurate. An example from the *dev* dataset illustrating this challenge is provided below.

Example:

Post:
#AIUDF_Pakistan_Zindabad. #AntiNationalAIUDF
"Pakistan Zindabad" Slogans Raised in Silchar Airport By AIUDF Members. #AIUDF supreme #BadruddinAjmal should immediately resign as MP ! #Shame #Shame I request District administration should take after the mtr immediately. DC Cachar #AIUDF_Pakistan_Zindabad. #AntiNationalAIUDF
"Pakistan Zindabad" Slogans Raised in Silchar Airport By AIUDF Members. #AIUDF supreme #BadruddinAjmal should immediately resign as MP ! #Shame #Shame I request District administration should take after the mtr immediately. DC Cachar #AIUDF_Pakistan_Zindabad. #AntiNationalAIUDF
"Pakistan Zindabad" Slogans Raised in Silchar Airport By AIUDF Members. #AIUDF supreme #BadruddinAjmal should immediately resign as MP ! #Shame #Shame I request District administration should take after the mtr immediately. DC Cachar None
Reference:
Pro-Pakistan slogans shouted outside an airport in India

Note that the reference claim includes additions such as "Pro-Pakistan","shouted", and "outside", which were introduced by the annotator and are not present in the original *post* text. Conversely, important details like **WHO**: "AIUDF Members" and **WHERE**: "Silchar" were omitted in the reference.

Llama output – initial extraction – attempt 1 – METEOR 17:
Slogans "Pakistan Zindabad" were raised in Silchar Airport by AIUDF Members.

The claim generated by *Llama* received a METEOR score of only 17, despite being factually accurate

and complete.

Llama output – self-reflection – attempt 1 – METEOR 11:
AIUDF members allegedly raised pro-Pakistani slogans ("Pakistan Zindabad") at Silchar Airport.

It could be argued that the final claim generated by *Llama* after self-reflection was in fact improved: it retained all relevant information and accompanied "Pakistan Zindabad" with the more general term "pro-Pakistani slogans" as in reference. However, the METEOR score dropped further to 11. The addition of the word "allegedly" was not consistent with the claim normalization strategy.

Llama output – initial extraction – attempt 2 – METEOR 13:
"Pakistan Zindabad" slogans were raised in Silchar Airport by AIUDF members.
Note: I've removed hashtags, emojis, and repeated text to extract the main claim, which is about the slogan "Pakistan Zindabad" being raised at the airport.

On a second attempt, the *Llama* model included additional text while still retaining a METEOR score of 13.

Llama output – self-reflection – attempt 2 – METEOR 12:
AIUDF members allegedly raised "Pakistan Zindabad" slogans at Silchar Airport.

Again, the self-reflection process objectively improved the output by removing extraneous text, but the METEOR score further decreased to 12.

DeepSeek output – initial extraction – attempt 1 – METEOR 16:
"Pakistan Zindabad" Slogans Raised in Silchar Airport By AIUDF Members. #AIUDF supreme #BadruddinAjmal should immediately resign as MP !

DeepSeek produced a meaningful output on the first attempt; however, it ignored the instruction to remove hashtags and received a METEOR score of 16.

DeepSeek output – self-reflection – attempt 1 – METEOR 16:
"Pakistan Zindabad Slogans Were Raised by AIUDF Members at Silchar Airport, Calling for Badruddin Ajmal's Resignation as MP."

The self-reflection process in *DeepSeek* removed the hashtags and polished the phrasing of the claim, but this improvement was not reflected in any change in the METEOR score. It is also worth noting that *DeepSeek* attempted to generate a more complex claim by combining the "slogans raised" and "Ajmal should resign" information pieces.

GPT output – initial extraction – attempt 1 – METEOR 22:
"Pakistan Zindabad" slogans were raised in Silchar Airport by All India United Democratic Front members.

GPT received a METEOR score of 22 by generating a claim similar to that produced by *Llama*, likely due to the expansion of an abbreviation and the addition of the word "India", which overlapped with the reference claim.

7. Conclusions

Our experiments indicate that claim normalization can be effectively handled by decoder-only LLMs, such as *GPT-4.1-mini*, even without fine-tuning. While the METEOR metric provides a quantitative measure of performance, it does not always align with factual correctness or task-specific nuances. This limitation underscores the importance of qualitative review when evaluating claim normalization

outputs.

Subtle aspects of fact-checking practice are challenging to encode as prompting guidelines. This may explain why fine-tuning—despite the *GPT* model already performing well with default weights—further improved performance. Fine-tuning likely helps align the model more closely with task-specific details that are hard to capture through prompting alone.

Interestingly, our results suggest that strict adherence to normalization guidelines may be particularly useful for clustering posts, even when these clusters differ from claims extracted by professional fact-checkers. In this sense, prioritizing guideline adherence could offer practical advantages beyond just accuracy metrics.

The self-reflection approach did not yield improved METEOR scores. However, manual analysis of the outputs revealed that self-reflection effectively removed many factual errors present in the initial generations. This highlights the potential of iterative improvement to refine outputs, even if it is not reflected in automated scoring.

A final observation concerns model-specific differences: *DeepSeek*'s "thinking" process, while thorough, appears less efficient, as many tokens are devoted to reasoning rather than concise output generation. In contrast, the more straightforward *Llama* outputs—despite scoring lower overall than fine-tuned *GPT*—remain practically usable, especially in resource-constrained scenarios where processing speed and scalability are critical.

The system based on the fine-tuned *GPT-4.1-mini* ranked second for Czech (METEOR on test dataset split was reported 21.44), Bengali (29.59), and Marathi (30.48), and third for Greek (23.33), Telugu (45.59), Romanian (23.50), Dutch (18.66), and Punjabi(26.96) out of 20 languages in the CLEF 2025 CheckThat! Task 2 competition. It is important to note that the system ranked second or third in 6 out of 7 languages for which only test data were available (the so-called zero-shot setting).

Overall, our findings emphasize that decoder-only LLMs can achieve strong results for claim normalization, and that practical trade-offs (such as hardware limitations and processing time) should be carefully considered when selecting models and techniques for real-world deployments.

8. Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT to check grammar, spelling, and style. The tool was applied to selected paragraphs, and all corrections were manually reviewed and approved.

Acknowledgments

The research is supported by the project "OpenFact – artificial intelligence tools for verification of veracity of information sources and fake news detection" (INFOSTRATEG-I/0035/2021-00), granted within the INFOSTRATEG I program of the National Center for Research and Development, under the topic: Verifying information sources and detecting fake news.

References

[1] M. Sundriyal, T. Chakraborty, P. Nakov, From chaos to clarity: Claim normalization to empower fact-checking, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 6594–6609. URL: https://aclanthology.org/2023.findings-emnlp.439/. doi:10.18653/v1/2023.findings-emnlp.439.

[2] J. Su, J. Healey, P. Nakov, C. Cardie, Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms, 2025. URL: https://arxiv.org/abs/2505.00127. arXiv:2505.00127.

[3] L. Pan, M. Saxon, W. Xu, D. Nathani, X. Wang, W. Y. Wang, Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies, Transactions of the

Association for Computational Linguistics 12 (2024) 484–506. URL: https://aclanthology.org/2024.tacl-1.27/. doi:10.1162/tacl_a_00660.

[4] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, P. Clark, Self-refine: Iterative refinement with self-feedback, 2023. URL: https://arxiv.org/abs/2303.17651. arXiv:2303.17651.

[5] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, D. Zhou, Large language models cannot self-correct reasoning yet, 2024. URL: https://arxiv.org/abs/2310.01798. arXiv:2310.01798.