Automated Textbook Auditing with Multi-Agent LLM Systems

Ciprian Cristescu[1], Adrian-Marius Dumitran[1], Angela-Liliana Dumitran[2] and Gabriel Ștefan[1]

[1]University of Bucharest, Bucharest, Romania
[2]Dimitrie Cantemir Christian University, Bucharest, Romania

Abstract

Ensuring the quality of educational materials requires more than standard proofreading: textbooks must be audited for factual accuracy, domain-specific technical correctness, and linguistic quality simultaneously — a task that general-purpose grammar checkers cannot address. We present **AI Textbook Auditor**, a modular multi-agent pipeline for automated quality assurance of educational materials across subject domains. The system accepts a textbook PDF and produces a structured, human-reviewable report via two analysis tracks: a **Factual and Technical Track** in which an ensemble of specialized LLM agents detects factual inaccuracies, code errors, incorrect definitions, and conceptual inconsistencies, augmented with web search for humanities domains; and a **Grammar Track** operating PDF-natively to preserve diacritical encoding. A **Judge Agent** filters false positives using domain-specific rules before presenting findings to a human reviewer. The pipeline supports two ingestion modes — vision-native page rendering and PyMuPDF text extraction — and is domain-adaptable via custom prompts encoding subject-specific error taxonomies. We demonstrate the system on two Romanian upper-secondary textbooks: a CS textbook (56 technical findings across seven categories, with an expert-validated precision of 62.5%) and a history and social sciences textbook (72 findings spanning factual errors, ideological bias, and grammar). The system is designed as a triage tool that reduces the manual effort of locating candidate issues, with human expert validation required before any editorial action.

1. Introduction

High-quality textbooks are foundational to effective teaching, yet ensuring their quality at scale remains an open challenge. General-purpose grammar checkers address only the linguistic dimension and are blind to the error types that matter most in technical and scientific textbooks: a misplaced stream operator in a C/C++ example, an extra mathematical symbol, a historical date that contradicts the established record, or a mountain's altitude stated incorrectly in a geography chapter.

We present **AI Textbook Auditor**, a modular multi-agent system for automated quality assurance of educational materials. An ensemble of **specialized LLM Agents** analyzes textbook content for factual inaccuracies and domain-specific technical errors, followed by a **Judge Agent** that filters false positives before presenting findings to a human reviewer. The system is domain-adaptable with the help of a domain expert via a custom prompt specifying the subject, error taxonomy, and negative constraints; for humanities textbooks it is augmented with web search for claim verification, while for technical subjects it relies on parametric model knowledge. We demonstrate the system on Romanian upper-secondary textbooks across two subject domains.

The main contributions of this work are: **(i)** a modular multi-agent auditing pipeline with domain adaptation via custom prompts; **(ii)** a Judge Agent for domain-aware false-positive filtering without additional human annotation in the detection loop; **(iii)** a unified architecture supporting web-search mode for humanities and parametric-knowledge mode for technical domains; **(iv)** preliminary evaluation on two Romanian upper-secondary textbooks spanning CS and humanities domains.

iTextbooks'26: Seventh Workshop on Intelligent Textbooks, June 28, 2026, Seoul, Republic of Korea
cristescuciprian.cc@gmail.com (C. Cristescu); marius.dumitran@unibuc.ro (A. Dumitran); dumitranangela@gmail.com (A. Dumitran); gabrielstefan04@gmail.com (G. Ștefan)
0009-0005-3547-5772 (A. Dumitran); 0009-0003-3590-9441 (A. Dumitran); 0009-0007-4193-7181 (G. Ștefan)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

2. Related Work

Automated educational content assessment has been studied primarily from the perspective of student-produced text: Grammar Error Correction [1, 2], automatic scoring [3, 4], and feedback generation [5]. Relatively little work targets the quality of published textbook content itself, and to our knowledge no prior system addresses the combined detection of factual inaccuracies and domain-specific technical errors across subject areas — CS textbooks being one concrete instance of a broader quality assurance challenge

Multi-agent LLM architectures have demonstrated that role specialization and structured inter-agent communication improve performance on complex reasoning tasks [6]. Our system applies this principle to textbook auditing, combining a domain-adaptable technical agent ensemble with a dedicated Judge Agent for false-positive filtering.

Code error detection via static analysis and compiler-based tools works well for compilable code but cannot handle the incomplete fragments, pseudocode hybrids, and illustrative snippets typical of textbook content. LLMs have shown strong performance on code understanding and error detection tasks [7, 8, 9], making them well-suited for this setting.

Web-augmented fact-checking with LLMs has been explored for claim verification tasks [10, 11]. Our system integrates optional web search for humanities domains where parametric model knowledge is insufficient, while relying on parametric knowledge alone for technical subjects where language standards and algorithmic theory provide a stable verification basis.

3. System Architecture

The pipeline accepts a textbook PDF and produces a structured, human-reviewable report of candidate errors. Figure 1 provides an overview of the end-to-end architecture. All components are implemented as a Streamlit application with exportable CSV and Excel output.

3.1. Ingestion and Document Segmentation

The system supports two ingestion modes: **vision-native** (V3), where pages are rendered to images for a multimodal LLM, avoiding the diacritical encoding artifacts common in Romanian publisher PDFs; and **text-extraction** (V2), using PyMuPDF with Unicode NFC normalization for lower cost on large documents. Both share the same downstream agents and output schema. Document structure is induced from the table of contents via a vision step extracting *(chapter title, start page)* pairs, editable before analysis; ToC failure falls back to fixed-size page blocks. Chapter text exceeding the context window is split with a sliding-window chunker (9 000 characters, 700-character overlap) to avoid truncating code examples.

3.2. Agent Ensemble

The analytical core comprises two specialized agents operating in parallel on each chapter chunk.

Factual agent. The Factual Agent extracts verifiable claims from the text and checks them for accuracy. For **humanities domains** (history, geography, literature) it is augmented with web search via Tavily, issuing targeted queries per claim and grounding verdicts in retrieved evidence. For **technical and exact-science domains** it relies on parametric model knowledge, which is sufficient for verifying code correctness, standard compliance, and algorithmic claims without retrieval overhead.

Technical agent. The Technical Agent supports two prompt modes: a **generic** cross-domain mode that detects factual, mathematical, scientific, conceptual, contradiction, and code/logic errors without a fixed taxonomy, and a **custom** mode in which a domain-specific error taxonomy is encoded directly in the prompt. For computer science textbooks the custom mode operates against an eight-category

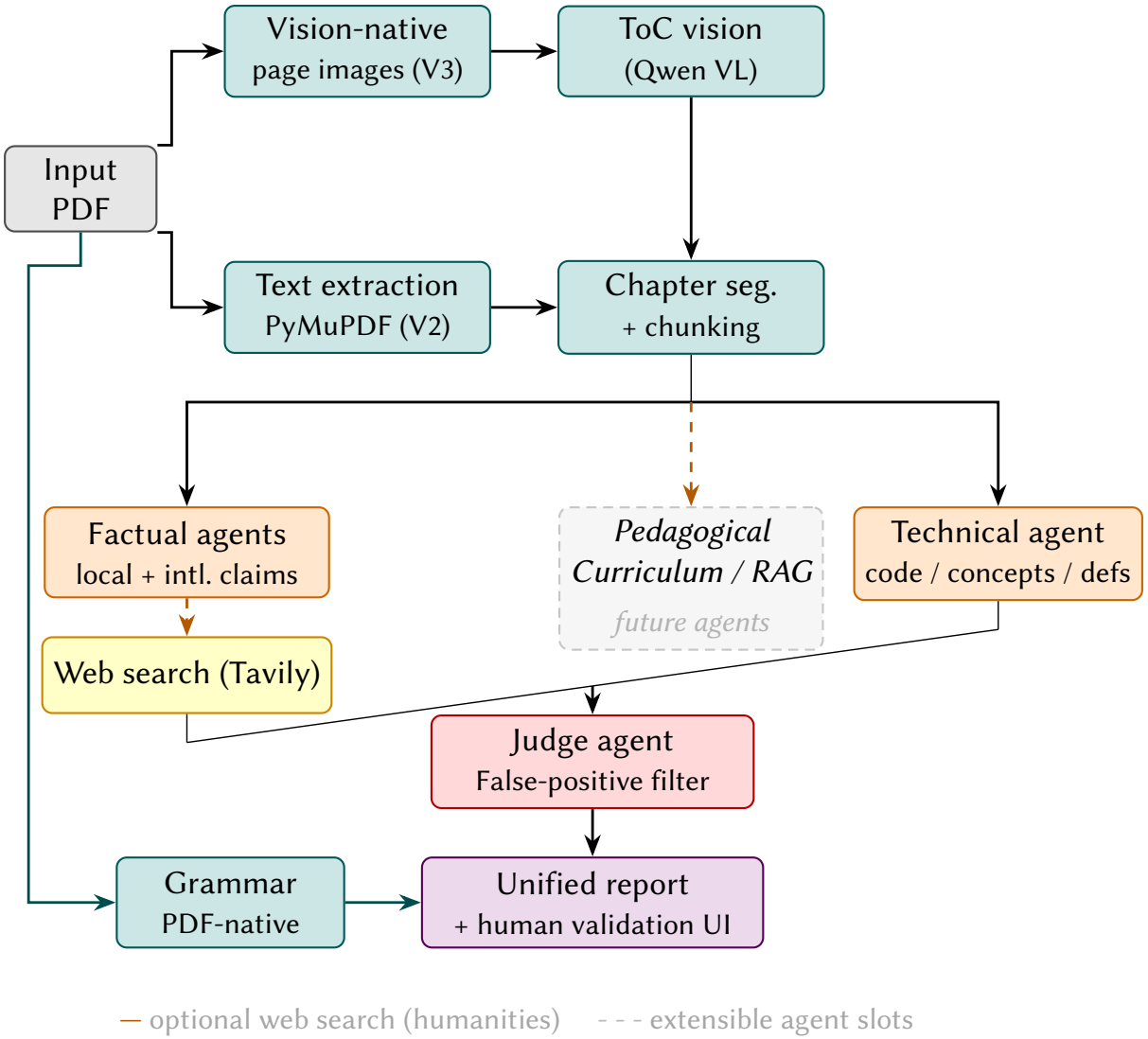

Figure 1: Overview of the auditing pipeline.

Table 1
Technical error taxonomy for CS textbooks, encoded as domain-specific prompt constraints.

Category	Description / example
Syntax	Invalid C/C++ syntax: `int:x;`, `Typedef struct`, missing semicolons
Code	Logic errors: and instead of &&, inconsistent loop variable, literal in condition
Pseudocode	Digit 0 as letter O, wrong assignment operator, logic contradicts surrounding explanation
Example	Code or pseudocode output contradicts the described result
Concept	False language behavior claims: "the first array element is `T[1]`" in C/C++
Definition	Wrong data structure or algorithm definition: stack described as FIFO, incorrect complexity
Standard	`void main()` or `#include <iostream.h>` presented without qualification as current standard
Portability	Compiler-specific patterns presented as universally valid without caveat

taxonomy (Table 1) derived from empirical analysis of Romanian CS textbooks, covering invalid syntax, logic errors, pseudocode mistakes, incorrect concept definitions, non-standard constructs, and portability issues; the taxonomy is replaceable for other technical domains. Either mode is selectable per run.

Both agents follow a consistent prompt engineering pattern: role specialization, explicit negative constraints (what not to report), strict JSON schema forcing, and an empty-list instruction to suppress

hallucinated findings when no clear error is present.

Extensibility. The architecture accommodates additional specialized agents without modifying the core pipeline — candidates include Pedagogical, Bias, Curriculum Alignment, and RAG-based verification agents (shown as dashed slots in Figure 1).

3.3. Judge Agent

Raw agent outputs exhibit a non-trivial false-positive rate from two systematic sources: encoding artifacts in extracted text (corrupted diacritical characters misread as content errors) and over-flagging of valid domain conventions (e.g. 1-based indexing in Pascal contexts, standard Romanian CS terminology). The Judge Agent performs a second-pass LLM call over aggregated findings, processing them in batches and returning a binary verdict (`validat: true/false`) with a brief justification. Its prompt encodes explicit false-positive and true-positive criteria derived from observed failure modes, requiring no additional human annotation in the detection loop.

3.4. Domain Adaptation via Custom Prompts

The agent ensemble is fully parameterized by a **custom prompt** specifying the subject domain, error taxonomy, and negative constraints for the textbook under review. Authoring this prompt requires a **domain expert**: the quality of the audit depends directly on a subject specialist encoding the relevant error categories and the conventions that must *not* be flagged, since a mis-specified taxonomy is the main source of false positives. Given such a prompt, switching domains — e.g. from CS to history — requires only replacing the taxonomy and enabling web search, while the pipeline, Judge, and interface remain unchanged.

3.5. Grammar Module

A parallel grammar module, authored with a specialist in Romanian orthography, checks linguistic compliance independently of the factual track; its findings bypass the Judge Agent and default to unvalidated pending review. As the norms live in the prompt rather than the architecture, the module is adaptable to other languages. Its full design and evaluation are a separate contribution beyond this paper's scope; we note it here as a parallel track (Figure 1).

3.6. Human Validation Interface

All findings are consolidated into a unified report (`validated`, `type`, `chapter`, `fragment`, `suggestion`, `explanation`) presented as an editable data grid. Reviewers toggle a `validated` checkbox per finding and export validated results as CSV or Excel.

4. Preliminary Evaluation

We report illustrative findings from two pilot runs on Romanian upper-secondary textbooks from different subject domains, demonstrating the system across both implemented agent configurations.

4.1. CS Domain: Upper-Secondary School Textbook

The technical agent was run, in its generic cross-domain prompt mode, on a 283-page Romanian CS upper-secondary textbook, producing 56 candidate findings across seven categories, as shown in Figure 2. Representative confirmed findings include:

- **Syntax:** `for (l=0; i<5; i++)` — the loop uses an undeclared variable `i`, so it never advances and does not compile (the parallel Pascal column is correct).

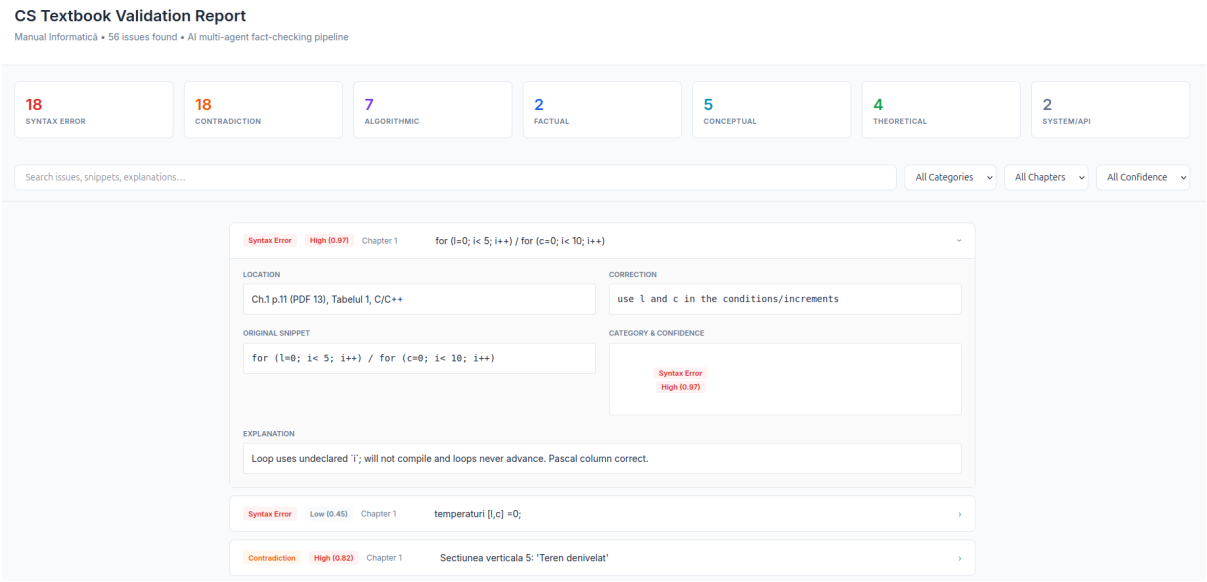

Figure 2: HTML validation report for the CS upper-secondary textbook. The statistics bar summarizes the technical findings by category. The expanded card shows a confirmed Syntax error with original fragment, suggested correction, and explanation.

- **Contradiction:** in a binary-search example both output branches report that the searched value *exists in the array*; the `else` branch should state that it does not.
- **Algorithmic:** a base-conversion routine validates its input with the impossible condition `b<2 && b>10`, which can never hold, defeating the check (a non-terminating loop in Pascal, a single useless pass in C++).
- **Theoretical:** a perfect number is defined as one whose *prime* divisors sum to their product; the correct definition is the sum of *proper* divisors ($6 = 1 + 2 + 3$), and the example holds only by coincidence.

Expert validation. To move beyond raw finding counts, a domain expert manually reviewed all 56 technical findings, labeling each as *correct*, *debatable* (context-dependent or stylistic), or *incorrect* (a clear false positive): 35 findings (62.5%) were confirmed correct, 13 (23.2%) debatable, and 8 (14.3%) incorrect, giving a strict precision of 62.5% and a hard false-positive rate of 14.3%. Reliability varied sharply by category: all 18 Syntax findings were correct, whereas contradiction-type findings were the noisiest (8 correct, 5 debatable, 5 incorrect). The model's self-reported confidence separated correct from incorrect findings only weakly (mean 0.62 vs. 0.43, with substantial overlap), indicating that a confidence threshold alone is insufficient and motivating the separate Judge and human-validation stages.

4.2. Humanities Domain: History and Social Sciences Textbook

The factual and bias agent ensemble (V2 configuration, with Tavily web search) was run on a Romanian upper-secondary history and social sciences textbook (A364), producing 72 candidate findings: 49 grammar, 14 bias/nuance, 4 international perspective, and 5 factual errors. Representative findings include:

- **Factual:** The names of Nobel Prize laureates Frederick Banting and John Macleod appear as "F.G. Bauting and J. Meheod"; the Exxon Valdez oil tanker is rendered as "Exon Vald"; "Three Mile Island" appears as "Three Miles Island". These require world knowledge unavailable to a standard grammar checker, motivating the LLM-based approach.

- **International perspective:** The textbook's claim that the Yalta Declaration had no role in Soviet policy toward Romania was flagged as contradicting international historiographical consensus, with archival evidence cited by the agent.
- **Bias/Nuance:** Gender-stereotyped language in two passages — one attributing "the masculine concept of freedom" to men, another framing childcare as an exclusively female dilemma — flagged for reformulation.

4.3. Discussion

The two pilot runs illustrate how domain adaptation via custom prompts yields qualitatively different but complementary audit profiles. On the 283-page CS textbook (V3, vision-native), the technical agent produced 56 reviewed findings, dominated by Syntax (18), Contradiction (18), and Algorithmic (7) errors, with an expert-validated precision of 62.5%. On the 131-page history and social sciences textbook (V2, text extraction with web search), the factual and bias agents produced 72 findings: 49 grammar, 14 bias/nuance, 4 international perspective, and 5 factual errors. The CS run surfaces verifiable technical errors in code and definitions, while the humanities run surfaces factual inaccuracies, historiographical bias, and linguistic errors that co-occur in published educational content. All findings are candidate issues pending human expert validation.

Limitations and future work. We report a first precision estimate on the CS technical track, but the evaluation remains preliminary: we do not yet measure recall, have no validation of the humanities or grammar tracks, and report no Judge Agent ablation. Because the Judge trades false positives against recall (it can discard genuine errors) and rejected findings are not currently surfaced to the reviewer, such losses are invisible. The factual track relies on parametric LLM knowledge for technical domains and web search for humanities domains; neither is infallible, and both can reproduce dominant narratives or curriculum-specific conventions, producing false positives the Judge does not fully eliminate. The demonstration covers two Romanian textbooks, one per domain; generalization to other languages and subjects (mathematics, natural sciences), richer inter-agent interaction, and multi-textbook evaluation remain open. Latency is substantial given sequential per-chapter LLM calls, suiting offline batch auditing rather than real-time use.

5. Conclusions.

We have presented **AI Textbook Auditor**, a modular multi-agent pipeline for automated quality assurance of educational materials. The system addresses a gap in educational NLP by combining factual and technical error detection with a domain-adaptable prompt mechanism and a Judge Agent for false-positive filtering. Pilot runs on a Romanian CS upper-secondary textbook (56 expert-validated technical findings, 62.5% precision) and a history and social sciences textbook (72 findings spanning factual errors, bias, and grammar) demonstrate the system's ability to surface candidate issues across qualitatively different domains. A grammar analysis module provides a complementary linguistic track. Future work includes formal recall evaluation, a Judge Agent ablation, and extension to other STEM subjects and additional agent types (curriculum alignment, RAG-based standard verification).

6. Ethical considerations.

The system is designed as an assistive tool for human editors, not an autonomous decision-making system. All findings require expert validation before any editorial action is taken; the `validated` field and the interactive interface are specifically designed to keep a human reviewer in the loop. The textbooks analyzed are publicly available through the Romanian national curriculum; no student data or unpublished content is processed. candidate findings, not confirmed errors — acting on unreviewed findings risks introducing incorrect editorial changes to otherwise correct content.

Acknowledgments

We would like to thank Together AI for providing the API credits used for model evaluation. Additionally, the authors acknowledge the early contribution of Victor Albu to the design of the system.

Declaration on Generative AI

During the preparation of this work, the authors used Claude (Anthropic), ChatGPT (OpenAI), and Gemini (Google) in order to: Grammar and spelling check, Paraphrase and reword, and Improve writing style. After using these tools and services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, C. Bryant, The CoNLL-2014 shared task on grammatical error correction, in: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 1–14. URL: https://aclanthology.org/W14-1701/. doi:10.3115/v1/W14-1701.

[2] C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, T. Briscoe, Grammatical error correction: A survey of the state of the art, volume 49, MIT Press, Cambridge, MA, 2023, pp. 643–701. URL: https://aclanthology.org/2023.cl-3.4/. doi:10.1162/coli_a_00478.

[3] S. Burrows, I. Gurevych, B. Stein, The eras and trends of automatic short answer grading, International Journal of Artificial Intelligence in Education 25 (2015) 60–117. doi:10.1007/s40593-014-0026-8.

[4] W. A. Mansour, S. Albatarni, S. Eltanbouly, T. Elsayed, Can large language models automatically score proficiency of written essays?, in: Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 2777–2786. URL: https://aclanthology.org/2024.lrec-main.247/.

[5] E. Kochmar, D. Vu, R. Belfer, V. Gupta, I. Serban, J. Pineau, Automated Personalized Feedback Improves Learning Gains in An Intelligent Tutoring System, 2020, pp. 140–146. doi:10.1007/978-3-030-52240-7_26.

[6] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, J. Schmidhuber, Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL: https://arxiv.org/abs/2308.00352. arXiv:2308.00352.

[7] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, et al., Evaluating large language models trained on code (2021). URL: https://arxiv.org/abs/2107.03374. arXiv:2107.03374.

[8] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez, et al., Code llama: Open foundation models for code (2024). URL: https://arxiv.org/abs/2308.12950. arXiv:2308.12950.

[9] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. K. Li, F. Luo, Y. Xiong, W. Liang, Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL: https://arxiv.org/abs/2401.14196. arXiv:2401.14196.

[10] J. Thorne, A. Vlachos, C. Christodoulopoulos, A. Mittal, FEVER: a large-scale dataset for fact extraction and VERification, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 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.

[11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020.