iTextbooks 2026 · 7th Workshop on Intelligent Textbooks · AIED 2026

Automated Textbook Auditing with Multi-Agent LLM Systems

Ciprian Cristescu · Adrian-Marius Dumitran · Angela-Liliana Dumitran · Gabriel Ștefan

University of Bucharest · Dimitrie Cantemir Christian University

WHY NOW

Romania is rewriting every textbook at once

The first structural curriculum reform in 20 years

rolls out from grade 9 in 2026–2027, across all subjects and all four high-school years.

New, curriculum-conforming textbooks are due from 2026

— every discipline, every grade, multiple competing titles per subject, all needing approval on a compressed timeline.

The review burden

~1,000

new titles on the order of

100,000+

pages for editors and the Ministry to review — by hand

Volume is an order-of-magnitude estimate; the reform and 2026 textbook timeline are the Ministry's own.

THE PROBLEM

Grammar checkers can't read the content

What they miss

- a misplaced stream operator in a C++ example
- a sorting algorithm defined as a stack
- a historical date that contradicts the record
- a definition that is simply wrong

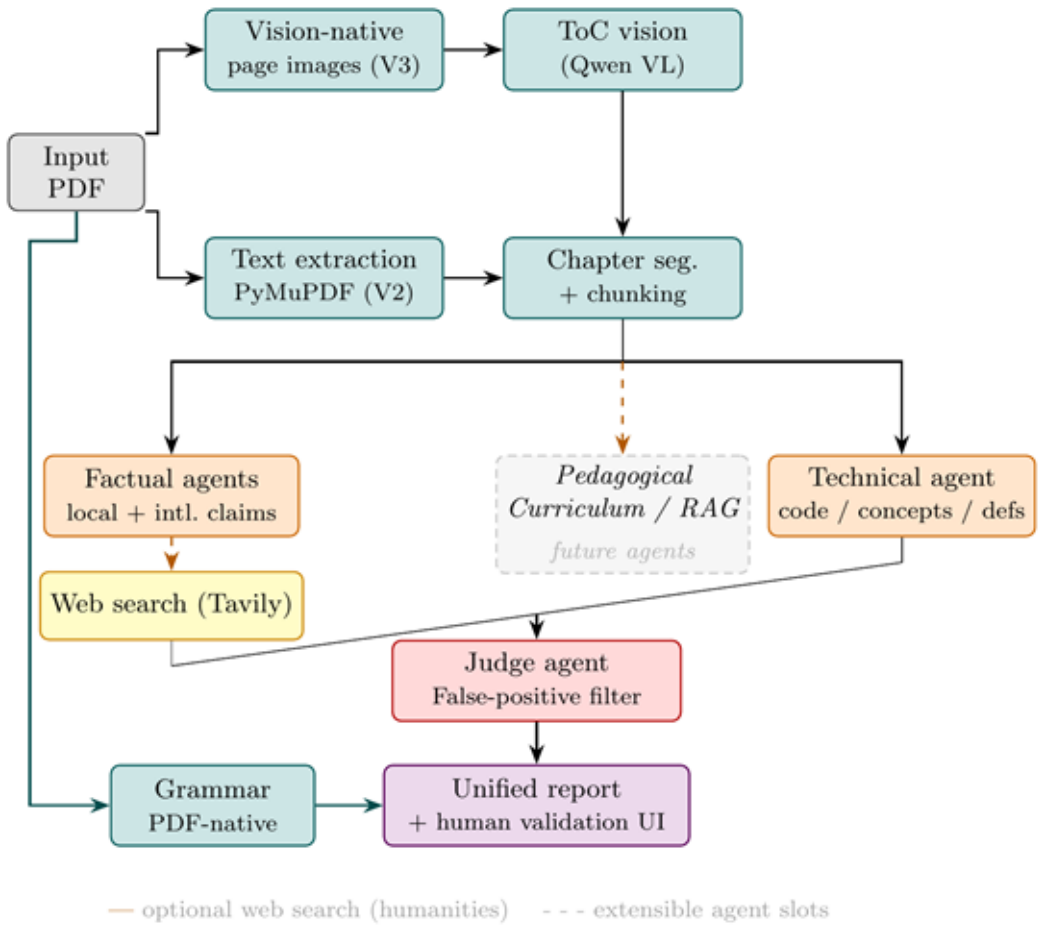
Why it matters

Textbooks are read by thousands of students as ground truth.

Auditing for factual, technical, and conceptual correctness at once is a task no general-purpose tool addresses.

THE SYSTEM

One pipeline, ingestion to human review



How to read it

Ingest — vision-native pages or PyMuPDF text; chapters cut from the ToC.

Two agents in parallel — Factual (web search for humanities) and Technical (code, concepts, definitions).

Judge — filters false positives before anything is shown.

Human validation — every finding reviewed before any edit.

Dashed = extensible slots for future agents; no core change.

Retargets to a new subject by swapping one expert-authored prompt — the taxonomy and what NOT to flag — with no architectural change.

WHAT IT ACTUALLY FOUND

Real errors in a 283-page Romanian CS textbook

Every example below shipped in a published, in-use textbook.

EXAMPLE 1 · CONTRADICTION

Binary search: both branches say “it exists”

LIMBAJUL PASCAL	LIMBAJUL C/C++

var x:array[1..20]of integer;
n,c:integer;
f:text;
……..
function cautare_binara(p:integer;u:integer):integer;
var poz:integer;
begin
poz:=(p+u) div 2;
if(c=x[poz]) then cautare_binara:=1
else
if(p<u) then
if(c<x[poz]) then
cautare_binara:=cautare_binara(p,poz-1)
else
cautare_binara:=cautare_binara(poz+1,u);
cautare_binara:=0;
end;
begin
……..
if(cautare_binara(1,n)<>0) then
writeln(' Valoarea ',c,' exista in sir')
else
writeln(' Valoarea ',c,' exista in sir');
……..
end.
	#include<fstream.h>
	ifstream f(“caut_bin.in”);
	int x[20],n,c;
	……..
	int cautare_binara(int p,int u)
	{int poz=(p+u)/2;
	if(c==x[poz])
	return 1;
	else
	if(p<u)
	if(k<x[poz])
	return cautare_binara(p,poz-1);
	else
	return cautare_binara(poz+1,u);
	return 0;
	}
	
	void main()
	{……..
	if(cautare_binara(1,n))
	cout<<” Valoarea “<<c<<” exista in sir”;
	else
	cout<<” Valoarea “<<c<<” nu exista in sir”;
	……..
	}

The bug

Pascal: the `else` branch also prints “Valoarea … exista in sir.” It should say it does **not** exist.

C++ bonus: the recursive call tests `k` — a variable that was never declared (should be `c`).

EXAMPLE 2 · ALGORITHMIC

An input check that can never trigger

(ultimul cat este 0).

LIMBAJUL PASCAL	LIMBAJUL C/C++

uses crt;
var n,b:integer
 	
procedure baza(n:integer; b:integer);
var r:integer;
begin
r:=n mod b;
if (n>=b)
baza(n div b,b);
write(r);
end;
 	
begin
clrscr;
write(' n = ');readln(n);
repeat
write(' baza = ');readln(b);
until(b<2 and b>10);
write(n,'→', baza(n,b);
end.
	#include<iostream.h>
	#include<conio.h>
	
	void baza(int n, int b)
	{ int r=n%b;
	if (n>=b)
	baza(n/b,b);
	cout<<r;
	}
	
	void main()
	{ int n, b;
	clrscr();
	cout<<" n = ";cin>>n;
	do
	{ cout<<" baza = ";cin>>b;
	}while((b<2) && (b>10));
	cout<<n<< "→"<<baza(n,b);
	}

The bug

The base must be 2–9. Both columns validate with

```
b < 2  &&  b > 10
```

No number is both `< 2` and `> 10`. The condition is always false — so the validation does nothing. **In Pascal the loop never ends; in C++ it runs once.**

EXAMPLE 3 · WRONG DEFINITION

A definition that works only by accident

What the textbook says

“A perfect number is one whose prime divisors sum to their product.”

$$6 = 1 + 2 + 3 = 2 \times 3$$

…and the example checks out, so nobody questions it.

Why it’s wrong

A perfect number equals the sum of its **proper divisors**, not its prime ones.

The prime divisors of 6 are {2, 3}. **“1 + 2 + 3” isn’t a sum of prime divisors at all** — the example holds only by coincidence. The definition conflates proper divisors with prime divisors.

EXAMPLE 4 · CROSS-TABLE REASONING

It re-derived the book's own rule — and caught a mislabel

Two vertical sections (Tabelul 4)

	Col 5	Col 6
values	69 · 30 · 39 · 42	73 · 19 · 20 · 50
shape	down→up	down→up
labeled	"uneven terrain"	"Valley, min 19"

Identical structure — but only one is called a valley.

What the model did

1 pulled the definition of "Valley" from a different table (Tabelul 6)

2 applied it to column 5's data — one descent, then a climb: a valley, min 30

3 noticed column 6 has the same shape and IS labeled a valley

4 concluded column 5 is mislabeled — and verified against the page image

No linter does this. *It's internal-consistency reasoning over the book's own logic.*

SAME PIPELINE, DIFFERENT SUBJECT

It transfers to the humanities

CS textbook (technical mode)

- 56 findings across 7 categories
- syntax, contradictions, algorithmic & conceptual errors
- parametric knowledge — no web search needed

History textbook (web-search mode)

- 72 findings: 49 grammar, 14 bias/nuance, 5 factual, 4 perspective
- Nobel laureates “F.G. Bauting and J. Meheod” (Banting & Macleod)
- claim contradicting historiographical consensus, flagged with evidence

DOES IT ACTUALLY WORK?

Expert-validated, and honestly so

62.5%

of 56 CS findings confirmed correct by a domain expert

18 / 18

Syntax findings correct — the reliable end

14.3%

hard false-positive rate — why the Judge + human stages exist

Model confidence separated right from wrong only weakly (0.62 vs 0.43) — a threshold alone won't do; you need the Judge and a human.

HONEST LIMITS

What we don’t claim yet

No recall measured

We report precision on the CS track only — we don’t yet know what the system misses.

Judge is double-edged

It cuts false positives but can discard real errors, and rejects aren’t shown to the reviewer.

Two books, two domains

One CS, one humanities. Generalisation to maths, science, and other languages is open.

Triage, not authority

A tool to locate candidate issues — every finding needs an expert before any edit.

TAKEAWAY

LLMs catch what linters and spell-checkers structurally can’t.

A domain-adaptable, judge-filtered, human-in-the-loop pipeline that surfaces real, content-level errors in published textbooks — across subjects.

Adrian-Marius Dumitran · **marius.dumitran@unibuc.ro**