Electronic Supplementary Information (ESI)

Optimizing Data Extraction from Materials Science Literature: A Study of Tools Using Large Language Models

Wenkai Ning,[a] Musen Li,*[abc] Jeffrey R. Reimers,*[ac] and Rika Kobayashi*[d]

Contents

[a.] *Department of Physics, International Centre of Quantum and Molecular Structures, Shanghai University, Shanghai 200444, China*
[b.] *School of Materials Science and Engineering & Materials Genome Institute, Shanghai University, Shanghai 200444, China*
[c.] *University of Technology Sydney, School of Mathematical and Physical Sciences, Ultimo, New South Wales 2007, Australia*
[d.] *Supercomputer Facility, Australian National University, Canberra, ACT 2601, Australia*
†Email: musenli@shu.edu.cn, Jeffrey.reimers@uts.edu.au, rika.kobayashi@anu.edu.au

S1. Model and prompt details (Section 2.4)

This section provides information about the models and prompts used in this study to facilitate the reproducibility of the results and a clearer understanding of the tool's operation.

Table 4 in the main text lists the models used by ChatExtract and LangChain, while Table S1 presents more detailed version information for these models. The specific deployment procedures can be found on GitHub.[1]

Table S1. Model versions and acquisition URLs of the local LLMs used in this work.

Model	Version	URL
Nomic-embed-text	0a109f422b47	https://ollama.com/library/nomic-embed-text:latest
Bge-m3	790764642607	https://ollama.com/library/bge-m3:latest
Llama2:13b	d475bf4c50bc	https://ollama.com/library/llama2:13b
Llama3.1:70b	dfc9cf0b496aea479874ddce703154f07d22ec3d	https://huggingface.co/bartowski/Llama-3.1-Nemotron-70B-Instruct-HF-GGUF
Qwen2.5:14b	7cdf5a0187d5	https://ollama.com/library/qwen2.5:14b

The three figures in this section show the prompts used in ChatExtract (Fig. S1 and S2), LangChain, and Kimi (Fig. S3), taken from the extraction code available on GitHub.[1]

As a prompt engineering tool, ChatExtract employs multiple prompts and dynamically adjusts the subsequent prompt based on the previous output. We adjusted the prompts based on the original work[2] to adapt to our unified post-processing.

For the detailed workflow, please refer to the original study;[2] only key information is summarized as follows. Each sentence is processed sequentially. The first prompt, "*classif_q*" (Fig. S1), is used to determine whether a sentence contains a bandgap. Sentences identified as containing a bandgap are collected as "positive sentences". As shown in Fig. S2, each positive sentence is then examined to determine whether it includes multiple bandgap values. For sentences containing only one data entry, the LLMs are prompted to extract *Material*, *Value*, and *Unit* to form a complete data record. If a sentence contains multiple bandgap entries, the LLMs are prompted to list all data in a table and verify each field of every record before compiling the final results.

```
classif_q = f'Answer only "Yes" or "No" without any explanation. Based on the following text, is there a value of **{PROPERTY}** mentioned in it?\n\n'

ifmulti_q = f'Answer "Yes" or "No" only. Does the following text mention more than one value of **{PROPERTY}**?\n\n'

single_q = [
    f'Give the number only without units, do not use a full sentence. If the value is not present in the text, type "None". What is the value of the **{PROPERTY}** in the following text?\n\n',
    f'Give the unit only, do not use a full sentence. If the unit is not present in the text, type "None". What is the unit of the **{PROPERTY}** in the following text?\n\n',
    f'Give the name of the material only, do not use a full sentence. If the name of the material is not present in the text, type "None". What is the material for which the **{PROPERTY}** is given in the following text?\n\n',
]
```

Fig. S1. Basic prompt set of ChatExtract. This includes prompts for filtering irrelevant sentences, identifying sentences containing multiple data entries, and extracting data from positive sentences.

```
tab_q = f'Use only data present in the text. If data is not present in the text, type "None". Summarize the values of **{PROPERTY}** in the following text in a form of a table consisting of: Material, Value, Unit. Ensure that the "Value" and "Unit" are separated into different columns.\n\n'

tabfollowup_q = [
    [
        'There is a possibility that the data you extracted is incorrect. Answer "Yes" or "No" only. Be very strict. Is ',
        " the ",
        f" material for which the value of **{PROPERTY}** is given in the following text? Make sure it is a real material.\n\n",
    ],
    [
        'There is a possibility that the data you extracted is incorrect. Answer "Yes" or "No" only. Be very strict. Is ',
        f" the value of the **{PROPERTY}** for the ",
        " material in the following text?\n\n",
    ],
    [
        'There is a possibility that the data you extracted is incorrect. Answer "Yes" or "No" only. Be very strict. Is ',
        " the unit of the ",
        f" value of **{PROPERTY}** in the following text?\n\n",
    ],
]
```

Fig. S2. Additional prompt set of ChatExtract. This includes prompts for extracting data from sentences containing multiple entries into a table and for verifying each record in the table.

Figure S3 shows the prompts used by LangChain and Kimi, which are designed to output data in a tabular format. The *retrieval_query* is a reference used in the RAG retrieval phase to find the target chunk (the *Top-k* similar chunks will be used as contextual content (CONTEXT in Fig. S3) for the RAG generation phase to obtain extraction results), and it is also used in the generation phase as a QUESTION (Fig. S3). In the generation stage, the retrieved contextual information is inserted into the "{context}" placeholder, and *retrieval_query* is treated similarly. For Kimi, the ending *CONTEXT* and *QUESTION* components are removed, and the contextual information is provided by the uploaded PDF files.

```
prompt = """
You are an expert information extraction algorithm.
Extract all the band gap values in the CONTEXT given blow.
Output the band gap values in the form of a markdown table,
including: Material (name of the material), Value (band gap
value), Unit (unit of value).
Do not explain, only output the table in markdown format.
The output is strictly in the following format.
| Material | Value | Unit |
|----------|-------|------|
| ... | ... | eV |
| ... | ... | meV |
If no band gap values mentioned in the article, the following
table is acceptable:
| Material | Value | Unit |
|----------|-------|------|
| None | None | None |
---
CONTEXT: {context}
---
QUESTION: {retrieval_query}
Answer in markdown table:
"""

retrieval_query = "What are the materials' name and their band
gap values?"
```

Fig. S3. Prompts used by LangChain and Kimi, designed to produce outputs in tabular format.

S2. Kimi implementation (Section 2.4)

We extracted data by manually uploading PDF files one by one to its official webpage and using the prompt consistent with LangChain, using a small batch of data to test the potential gap between the local model and the web interface LLM.

The difference between the Kimi web interface and our LangChain deployment is that in addition to the basic input, inference, and output, the web interface includes additional processing such as

hidden system instructions, prompt optimization, context management, and filtering logic to assist human interaction and data security, as well as memory management which is less relevant in this work. Even with the same base LLM, its web interface, application programming interface (API), and open-source models may yield different results due to the aforementioned differences, as well as variations in deployment frameworks and minor model versions. Providing a strictly versioned local model is relatively stable for result reproduction, which is also one of the reasons we use local models instead of APIs.[3]

The GPT-4 used in the ChatExtract original work[2] is similar to Kimi in its implementation, and in addition to the stability reasons mentioned above, replacing it with local models can better reflect the effectiveness of Prompt Engineering technology itself rather than the model.

S3. Matching mechanism (Sections 2.5)

Regarding the determination of whether a data entry extracted by an AI tool is correct (i.e., a True Positive, TP, or a False Positive, FP), we employed a strict matching mechanism.

First, at the fundamental level of comparison: for *Material*, both human-annotated and AI-extracted materials were converted to lowercase, with leading, trailing, and intermediate spaces removed before comparison. For *Value* and *Unit*, all units were normalized to "eV", and the values were required to be exactly identical. In addition, the corresponding paper DOI must also match for the entry to be counted as a correct match (TP).

The challenging part lies in the *Material* expressions. When a material includes a description of its physical or chemical state corresponding to its chemical formula, we required both the formula and the state to be extracted. The state descriptions appear in parentheses after the chemical formula, such as in paper:[4] "Silicene on Ar(111) (without vdW interactions and without SOC)". However, since each model describes such states differently, and we did not provide additional prompts to standardize the format or examples (which may lead to worse results), we manually judged certain *Materials*. If the meaning expressed was consistent with the actual state (for instance, "ArSi (without vdW, without SOC)"), we also considered it a correct match and added this *Material* to the matching list for subsequent automated matching. Another example is from paper,[5] which reports three bandgap values corresponding to three states of Ag_3AuSe_2. Although nearly all LangChain variants correctly extracted the three *Value* entries, they only extracted the *Material* as "Ag_3AuSe_2" without the corresponding state descriptions; therefore, all were regarded as FPs.

The *Value* field is even more complex. As described in Table 3 in the main text under *Value Class*, we imposed strict requirements on numerical representation. For example, in paper,[6] the value corresponding to "white-p" is reported as "~1.4". Most methods extracted "1.4" instead, but these were all considered FPs. Another subtle case appears in paper,[7] where the *Material* "InI" is associated with two values, "2" (in a table) and "~2" (in the main text). The correct "2" appears in a table that is relatively difficult to extract, while the AI tools incompletely extracted "2" from the text (missing the tilde). However, since a correct "2" also exists, these entries—technically FPs—were treated as TPs. In addition, some implicit information requires extremely high language

comprehension ability from the model; for instance, in paper,[8] the term "gapless" denotes a bandgap value of 0, but none of the tools successfully extracted this.

In many cases, the AI tools were not incapable of extracting the exact values we required, but rather lacked appropriate extraction rules due to limitations in the prompt. Our implementations follow those of previous workers only. Considering that adding more complex prompts might affect extraction performance, and that our goal is to construct a database that balances accuracy and completeness, we regard this matching mechanism as reasonable at the current stage.

S4. Error Class (Section 2.5)

The hierarchical diagram in Fig. S4 illustrates the classification framework for Error Classes used to evaluate extraction results. We first distinguish whether a Material or Value exists in the source document. If neither is found, the case is defined as Class 1 (Hallucination). When both entities exist, the classification proceeds by identifying whether the Material or Value is incorrect. Incorrect materials are further divided according to the sufficiency of their descriptions, resulting in Class 2 (Insufficient Material) and Class 3 (Wrong Material). Errors in extracted values are categorized based on the sufficiency and correctness of their descriptions and associated units, leading to Class 4 (Inaccurate Value), Class 5 (Wrong Value), and Class 6 (Wrong Unit). Class 7 (Wrong Pairing) is used when two (or more) Materials and Values are present in the paper but their assignment is mismatched, e.g., if the text "The bandgap of ZnO is 3.4 eV. The bandgap of silicon is 1.1 eV." led to the extraction of the ZnO bandgap as 1.1 eV and the silicon bandgap as 3.4 eV.

It is worth noting that because Material, Value, and Unit are evaluated independently, multiple Error Classes may occur simultaneously. More specifically, Classes 2 and 3 for Material, Classes 4 and 5 for Value, and Class 6 for Unit may appear together in a single instance. To ensure consistency in labeling and interpretation, a priority scheme was established, allowing a maximum of two Error Classes to be assigned to any given data point. The priority order is defined as: Hallucination (Class 1) > Wrong Pairing (Class 7) > Material (Class 2/3) > Value (Class 4/5) > Unit (Class 6).

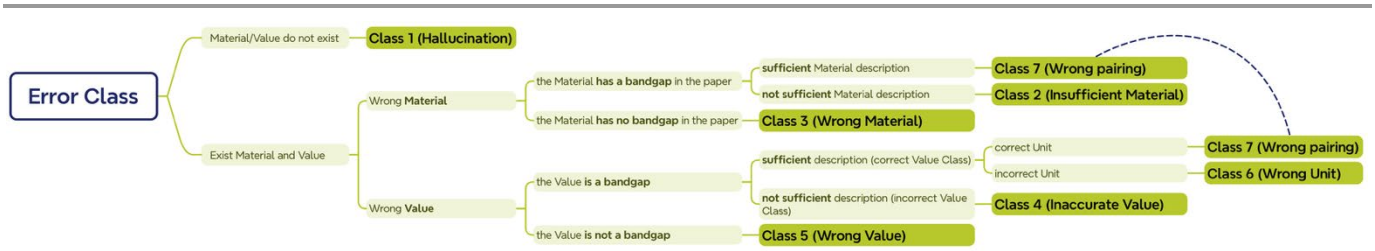

Fig. S4. Decision tree diagram about Error Class.

S5. Differences between arXiv and publisher presentations (Section 3.1)

There are some differences between the publisher[9] and arXiv[10] versions of the same papers. Figure S5 shows the difference on an example paper. The two subfigures have the same layout, with the left side being the PDF page, the upper right side showing an enlarged section of the PDF containing the bandgap value, and the lower right side displaying the parsed plain text.

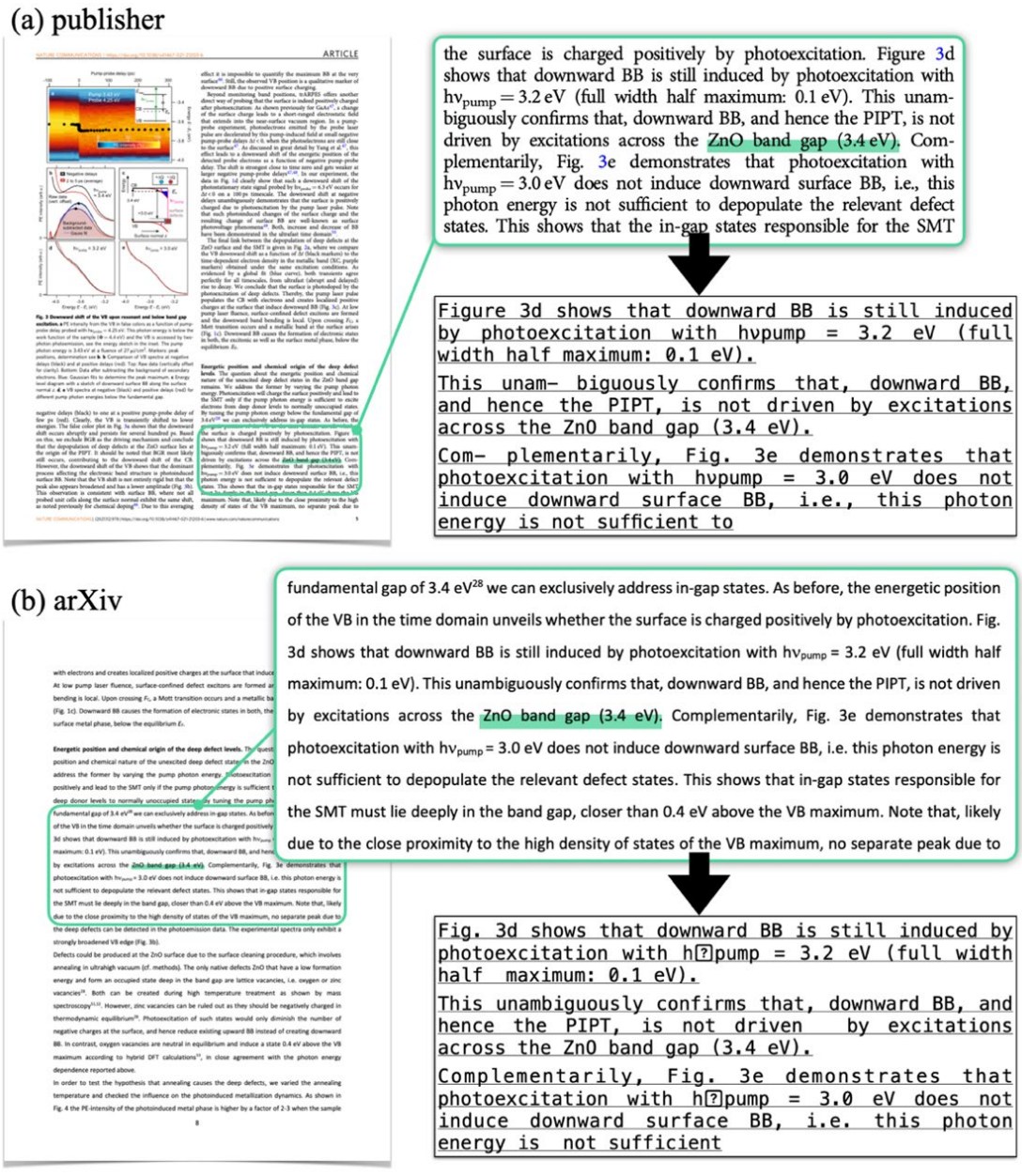

Fig. S5. Illustration of the differences between the publisher and the arXiv version, regarding the PDF layout and the plain text created after parsing. The publisher text[9] is reproduced, with only the highlighting boxes added, using its Creative Commins license.[11] The arXiv text[10] is reproduced, with only the highlighting boxes added, using its arXiv non-exclusive license.[12]

Regarding the position of the page in the full text, the publisher's is located on page 5 amongst 8 pages, whereas arXiv's is on page 8 out of 19 pages. Regarding the position of the target data on the page, the publisher's layout is double-column with a smaller line spacing, resulting in a higher information density; arXiv's layout is single-column with larger line spacing, containing less information on a single page. Regarding the parsed text, however, they are basically consistent (except for line breaks caused by the different column layouts and differences in symbol parsing possibly due to different fonts).

S6. Context analysis (Section 3.2)

We studied the mechanism of RAG in the information extraction process to analyze the context, taking the paper[9] highlighted in Fig. S5 as an example. From this data, the human-extracted information is that ZnO has a bandgap of 3.4 eV. Results for this target bandgap obtained using the six LangChain tools are listed in Table S2.

Table S2. Results of the contextual analysis research (partly). For more detailed information (such as the word count of the original content and chunk statistics after chunking), please refer to GitHub (Table S2).[1]

round	*sentence_count*	*sentence_index*	*chunk_count*	LC_{11}	LC_{12}	LC_{13}	LC_{21}	LC_{22}	LC_{23}
0	11	174-184	2	✓	✓	✓	✓	✓	✓
1	21	169-189	4	✓	✓	✓	✓	✓	✓
2	31	164-194	6	✓	✓	✓	✓	✓	✓
…	…	…	…	…	…	…	…	…	…
31	321	19-339	51	✓	✓	✓	✓	✓	✓
32	331	14-344	53	✓	✓	✓	✓	✓	✓
33	341	9-349	55	✗	✗	✗	✓	✓	✓
34	351	4-354	56	✗	✗	✗	✓	✓	✓
35	359	1-359	58	✓	✓	✓	✓	✓	✓
36	364	1-364	59	✓	✓	✓	✓	✓	✓

As described in Section 3.2 of the main text, different subsections of the plane text are sent to the AI tools for data analysis, starting with the text highlighted in Fig. S5; this is called data extraction *round* zero. Columns *sentence_count* and *sentence_index* indicate the number of sentences and the numbering of those sentences, respectively (the center sentence is the 179th sentence). Each row in Table S2 describes the results of subsequent extraction rounds, with each new round embodying an expansion of the text sent for analysis. For each round, the number of chunks obtained after chunking the expanded text is listed as *chunk_count*. The last six columns in the table indicate, for LC_{11}-LC_{23}, whether the target bandgap data was extracted (successful extractions are indicated by "✓", and vice versa by "✗").

Amongst the last six columns (LC_{11}-LC_{23}), the first three (LC_{1x}, pertaining to the Nomic-embed-text Embedding Model) show obvious differences in extraction results compared to the last three (LC_{2x}, Bge-m3 Embedding Model). The internal results of the two parts are basically consistent, indicating that in this case, the Embedding Model is the main factor affecting the extraction results. Additionally, we observed that as the amount of contextual content increases in each round, there

is a phenomenon where the extraction of information goes from success to failure, and then back to success. This is counter-intuitive, but is the crux of the matter. In this regard, we conducted further research.

To investigate the reason behind the observed transition from success to failure to success with increasing content, we conducted a more detailed process analysis for LC_{11} in *rounds* 32–35 (results for LC_{1x} were identical, so LC_{11} is used as a representative example). We examined the content of each chunk after chunking and identified the chunk indices containing the target information. After the retrieval step, we recorded the *Top-k* retrieved chunks and their similarity scores with respect to the "retrieval query". Finally, we obtained the extraction results from the LLM, as summarized in Table S3.

Table S3. Extraction process key intermediate information and result summary (conducting an in-depth study of the LC_{11} extraction results from rounds 32-35 in Table S2). For more detailed information (such as the content of each chunk obtained and the scores of the top-ranked chunks), please refer to GitHub (Table S3).[1]

round	*Top_k*	*chunks_count*	*correct_chunk_index*	*retrieved_chunk_index*	*extracted_correct_chunk_index*	Result
32	5	53	5, 28, 29	50, 0, 4, 21, 5	5	✓
33	5	55	6, 29, 30	51, 0, 5, 22, 1		
33	10	55	6, 29, 30	51, 0, 5, 22, 1, 6, 47, 2, 29, 25	6, 29	✓
34	5	56	7, 30, 31	52, 1, 6, 23, 2		
34	10	56	7, 30, 31	52, 1, 6, 23, 2, 7, 48, 3, 30, 26	7, 30	✓
35	5	58	8, 31, 32	8, 53, 1, 24, 7	8	✓

The focus of our analysis is on the two sets of chunk indices obtained during the process, i.e., the chunks identified as correct by humans after chunking (column *correct_chunk_index*) and the *Top-k* chunks after retrieval (column *retrieved_chu*nk_index). The overlapping indices in these two columns, which represents the correctly retrieved chunks, is presented in column *extracted_correct_chunk_index*.

In column *correct_chunk_index*, there are three correct chunk indices in all the rounds (32 to 35). Examination of the output files revealed that the contents of the three critical chunks remain identical; the change in numbering indicated in Table S3 occurs because new chunks are generated as the context length increases. The same pattern applies to the retrieved chunks after retrieval — the same chunk receives an index increment of one when the round number increases by one. In

round zero, only one value appears in *correct_chunk_index*, but by *rounds* 32-35 the target bandgap data (ZnO bandgap, 3.4 eV) appears twice in the paper, meaning that multiple *correct_chunk_indices* appear. Taking round 32 as an example, chunk 5 is outside the central sentence (which contains our target information), while chunks 28 and 29 both contain the central sentence itself (as chunks overlap by 200 tokens, as controlled by the *Chunk-overlap* hyperparameter). Ideally, we would expect the *retrieved_chunk_index* to include the two latter *correct_chunk_index* indices. The column *retrieved_chunk_index* lists the identified important chunks; its order is sorted by similarity in descending order, with the first one being the most similar.

The hyperparameter *Top_k* sets the total number of chunks to be retrieved. In Table S3, we consider values of *Top_k* = 5 and 10. An interesting observation emerged: amongst the four retrieval attempts with *Top-k* = 5, both successful extractions (rounds 32 and 35) drew data not from the central sentence we expected, but from a chunk with a smaller index in *retrieved_chunk_index*; in contrast, both failed extractions did not retrieve any of the three chunks containing the target information. Increasing *Top-k* to 10 resulted in the target chunks being retrieved multiple times. This confirms that the similarity between the target chunks and the retrieval query in RAG is crucial to the extraction result, and that once a target chunk is retrieved, the model is highly likely to extract the correct data.

A further analysis of the chunk indices and similarity scores (see the two JSON file on context analysis on GitHub)[1] reveals that the same chunk exhibits identical similarity scores with the retrieval query (for example, chunk 50 in round 32 and chunk 51 in round 33), which is consistent with expectations. Comparing *retrieved_chunk_index* between rounds 32 and 33, the 1st, 3rd, and 4th retrieved chunks are identical, while the 2nd and 5th differ. Examining their contents shows that correspondence between rounds begins from chunk 3 in round 32 and chunk 4 in round 33. Moreover, when moving from round 32 to 33, chunk 5 (which had enabled successful extraction in round 32) was replaced by chunk 1 in round 33, leading to extraction failure after increasing the context length. The similarity scores corroborate this finding. Therefore, we conclude that increasing the context length (i.e., higher round numbers) affects both the segmentation of certain chunks and their ranking order.

In summary, the retrieval query used in the RAG retrieval phase plays a crucial role in determining the extraction results. As the amount of text in one of the 37 processed sets increases, the resulting changes in chunk segmentation alter the structure of the vector database. Consequently, newly added irrelevant text may obtain higher similarity scores than the target chunk, potentially causing the loss of target data and its subsequent recovery as more text is incorporated. This conclusion, obtained here in response to the observed extraction of different results from papers presented in their publisher and arXiv formats, underpins inherent weaknesses in the RAG-based extraction tools.

S7. LangChain *Temperature* (Section 3.3.1)

To better select *Temperature* hyperparameters, we conducted multiple experiments and obtained four sets of results. The values corresponding to Fig. 4 in the main text are shown in Table S4. For more detailed data, please refer to GitHub (four Excel files about *Temperature*).[1]

Table S4. *F-scores* of the six LangChain variants under different *Temperature* hyperparameters settings.

Temperature	LC_{11}	LC_{12}	LC_{13}	LC_{21}	LC_{22}	LC_{23}
0	3	14	9	7	16	14
0.25	4	14	9	7	15	13
0.8	2	13	10	7	14	13
0.8	2	13	9	5	14	13

S8. *Chunk-size* and *Top-k* (Section 3.3.2)

In the discussion of Table S3, we tested the impact of *Top-k* on the extraction results. Here, we combined another factor that affects the context, *Chunk-size*, for further investigation. We combined two sets of *Chunk-size* (500 and 1000, corresponding to Chunk-overlap of 100 and 200, respectively) with two sets of *Top-k* (5 and 10), resulting in four sets of experimental results. The *F-score* is shown in Table S5, corresponding to Fig. 5 in the main text, with more detailed data available GitHub (four Excel files, one of which overlaps with the *Temperature* file).[1]

Table S5. *F-scores* obtained from 6 LangChain variants under different *Chunk-size* and *Top-k* hyperparameters settings.

Chunk-size	*Top-k*	LC_{11}	LC_{12}	LC_{13}	LC_{21}	LC_{22}	LC_{23}
500	5	3	7	3	3	8	5
500	10	2	8	7	3	11	9
1000	5	3	14	9	7	16	14
1000	10	6	15	14	2	12	11

S9. References

1. W. Ning, wenkaining/Bandgap-Extraction-Comparison, https://github.com/wenkaining/Bandgap-Extraction-Comparison, (accessed 2025-03-31 19:22:03).
2. M. P. Polak and D. Morgan, *Nature Communications*, 2024, **15**, 1569.
3. OpenAI, OpenAI Developer Community, https://community.openai.com, (accessed 2025-07-22 21:19:15).
4. S. Sattar, R. Hoffmann and U. Schwingenschlögl, *New Journal of Physics*, 2014, **16**, 065001.
5. M.-Á. Sánchez-Martínez, I. Robredo, A. Bidaurrazaga, A. Bergara, F. de Juan, A. G. Grushin and M. G. Vergniory, *Journal of Physics: Materials*, 2019, **3**, 014001.
6. J. A. Flores-Livas, A. Sanna, A. P. Drozdov, L. Boeri, G. Profeta, M. Eremets and S. Goedecker, *Physical Review Materials*, 2017, **1**, 024802.
7. Y.-T. Huang, S. R. Kavanagh, D. O. Scanlon, A. Walsh and R. L. Hoye, *Nanotechnology*, 2021, **32**, 132004.
8. X. Wang, L. Cheng, D. Zhu, Y. Wu, M. Chen, Y. Wang, D. Zhao, C. B. Boothroyd, Y. M. Lam and J. X. Zhu, *Advanced Materials*, 2018, **30**, 1802356.
9. L. Gierster, S. Vempati and J. Stähler, *Nature Communications*, 2021, **12**, 978.
10. L. Gierster, S. Vempati and J. Stähler, *arXiv preprint arXiv:2005.13424*, 2019, DOI: 10.48550/arXiv.2005.13424.
11. C. Commons, Deed - Attribution 4.0 International - Creative Commons, https://creativecommons.org/licenses/by/4.0/, (accessed 2025-10-23 13:50:32).
12. arXiv.org, arXiv.org - Non-exclusive license to distribute, https://arxiv.org/licenses/nonexclusive-distrib/1.0/license.html, (accessed 2025-10-23 13:49:15).