Ontology Guided Large Language Model Pipeline for Structured Information Extraction from Battery Cell Datasheets

Slimane Arbaoui*, Ali Ayadi, Ahmed Samet Tedjani Mesbahi, Romuald Boné

Université de Strasbourg, INSA Strasbourg, ICube laboratory UMR 7357
and CNRS, Strasbourg, 67000, France
prénom.nom@insa-strasbourg.fr

Résumé. L'extraction automatique d'informations à partir de fiches techniques de batteries est essentielle pour la modélisation précise et la gestion des performances des cellules électrochimiques. Ces fiches, souvent au format PDF non structuré, contiennent des informations critiques mais difficiles à exploiter automatiquement de manière fiable. Dans cet article, nous proposons une chaîne complète d'extraction basée sur une ontologie de domaine et des modèles de langage (LLMs). Notre approche utilise un pipeline dual-LLM combinant raisonnement étape par étape (Chain-of-Thought) et exemples à faible nombre (few-shot) pour extraire des triplets structurés à partir des datasheets. Une étape de validation par LLM garantit la conformité des triplets à l'ontologie et élimine les doublons ou les erreurs. Nous évaluons cinq modèles LLM sur plusieurs fiches techniques et validons manuellement les triples extraits. Les performances montrent que les modèles récents extraient des informations plus cohérentes et précises, confirmant l'efficacité d'une approche ontologique pour transformer des données hétérogènes en connaissances interrogeables.

1 Introduction

Batteries are becoming increasingly valuable as the world transitions toward the electrification of the transportation sector. Modern electric vehicles (EVs) rely on large battery packs composed of multiple modules, each containing hundreds or even thousands of individual battery cells. These cells form the core energy-storage units of any EV. Because most electrochemical characterization, aging studies, and performance validation tests are performed at the cell level, a significant amount of research focuses on developing accurate battery models, ranging from physics-based models Doyle et al. (1993); Santhanagopalan et White (2006) to data-driven and machine-learning-based models Singh et al. (2021); Attia (2020), using cycling data collected from individual cells.

A key challenge in cycling battery cells is obtaining accurate and complete information about the cell specifications, which is essential to ensure proper testing and avoid misuse. Critical parameters include the maximum continuous charging current, operating temperature range, cutoff charging voltage, and cutoff discharging voltage, among others. However, this

information is often difficult to locate or may be ambiguously presented. Most cells come with datasheets, typically in PDF format, but these documents may not always be available in English, especially for cells from major Chinese manufacturers such as EVE Energy, Sunwoda, Gotion High-Tech, or SVOLT Diao et al. (2021). Studies have shown that datasheets from different manufacturers can be inconsistent, incomplete, or difficult to interpret, which makes extracting operational limits and critical parameters challenging Diao et al. (2021); Hassini et al. (2023); Ali et al. (2024).

In recent years, artificial intelligence, especially in Natural Language Processing (NLP), has greatly improved the capabilities of large language models (LLMs). These models can now perform complex tasks such as reading comprehension, reasoning, and information extraction with high accuracy Brown et al. (2020); OpenAI et al. (2024). This makes it easier to extract knowledge from unstructured data, such as PDF datasheets, research papers, or manufacturer documents.

In battery research, LLMs can help read datasheets, understand specifications, and identify important parameters. Using LLMs, we can automatically get key information about battery cells, which is essential for accurate testing and modeling. However, LLMs can sometimes produce incorrect answers, a problem known as hallucination.

To address this, we propose a method for extracting datasheet information from commercial cells using a dual-LLM approach combined with two prompting techniques. We use a "Chain of Thought" (CoT) method to guide the LLM step by step on how to extract the needed information, providing a few example prompts (few-shot prompting) to help the model understand how to interpret the data. Studies have shown that CoT prompting, even in a zero-shot or few-shot setting, can significantly improve reasoning and information extraction performance without any model fine-tuning Kojima et al. (2023); Liang et al. (2023); Lei et al. (2025). To further reduce hallucinations, we use an ontology that describes all the information that needs to be collected from the datasheet. This ontology guides the model so that it only generates triples that follow the defined structure. Thus, our contributions are three-fold :

1. We develop a dual-LLM pipeline for extracting critical information from commercial battery datasheets. This system leverages both CoT and few-shot prompting techniques to guide the models step by step, enabling accurate extraction without requiring fine-tuning.
2. We introduce an ontology-driven approach to reduce hallucinations and ensure that the extracted information adheres to a structured format.
3. We demonstrate that our approach can handle heterogeneous, multi-language datasheets, including those from major Chinese manufacturers, providing a scalable and automated solution for parsing unstructured datasheet data to support battery testing and modeling.

2 Related Work

Recent years have witnessed a growing interest in using LLMs to automate the construction of knowledge graphs (KGs) from unstructured text. Surveys and reviews highlight that LLM-based approaches often outperform traditional NLP pipelines on tasks such as entity and relation extraction, event extraction, and knowledge graph augmentation, in some cases even

reducing the need for labeled data and manual annotation Ibrahim Nourhan (2024); D'Souza et Mihindukulasooriya (2024). For instance, the method iText2KG demonstrates a zero-shot, topic-independent pipeline capable of incrementally building KGs from scientific papers, webpages, or CVs, with no post-processing required Lairgi et al. (2024). Similarly, KG-GPT shows that LLMs can perform structured reasoning on KGs by combining sentence segmentation, graph retrieval, and inference, enabling tasks like fact verification and KG-based question answering without fine-tuning on each domain Kim et al. (2023). In more domain-specific applications, such as materials science and manufacturing, recent works have successfully extracted structured information (from tables or free text) using LLMs to build KGs, demonstrating applicability even on technical, domain-heavy documents Dreger et al. (2025); Ivanisenko et al. (2024).

However, these LLM-based KG construction methods also surface nontrivial challenges. A recurring issue is hallucination : LLMs may generate plausible but incorrect relations or entities when the input is ambiguous or sparse Ivanisenko et al. (2024); Ibrahim Nourhan (2024). Some works report inconsistent entity/relation resolution and semantic duplication when relying solely on LLM outputs Lairgi et al. (2024). Moreover, domain-specific and technical documents such as datasheets or scientific publications often contain specialized terminology and structured data (tables, units, parameters), which complicates automatic extraction. As a result, hybrid methods combining LLM-based extraction with ontology guided validation or structured post-processing are increasingly seen as promising, a direction that aligns with our proposed approach.

3 Methodology

We propose an approach that leverages two LLMs performing complementary tasks. The first LLM extracts triples from chunks of text obtained from PDF datasheets, which are collected by scraping the websites of major battery manufacturers. The second LLM validates the output of the first model, checking both the format and the compliance with the ontology concepts and relationships.

Our approach can be divided into three main components :

— Ontology creation, where we define all relevant concepts and relationships that can appear in a datasheet.
— Construction of the first LLM prompt, which uses the extracted PDF text, few-shot examples, and CoT techniques to guide the extraction process
— Construction of the validation LLM prompt, which acts as a "judge" to verify the extracted triples and ensure they conform to the defined ontology.

3.1 Battery Cell Ontology

The battery cell ontology is designed to systematically represent the information contained in battery cell datasheets. It captures electrical, thermal, mechanical, safety, and performance characteristics of individual battery cells, facilitating structured knowledge extraction and analysis. Inspired by the Battinfo ontology Bat, we propose the ontology shown in Figure 1.

The central entity of the ontology is *BatteryCell*, which aggregates all relevant information about a single electrochemical cell, including its manufacturer, model, electrochemistry,

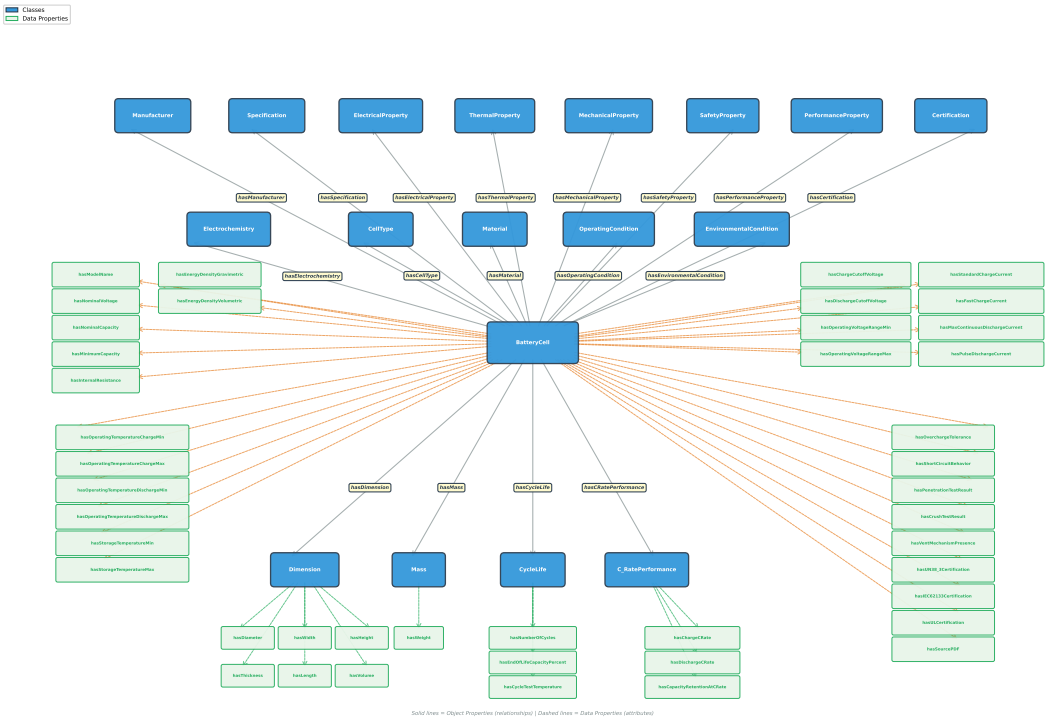

FIG. 1 – *The proposed battery cell ontology.*

mechanical format, and materials. *Manufacturer* links each cell to the producing company, enabling traceability and metadata-based filtering. *Electrochemistry* defines the chemical system (e.g., lithium nickel manganese cobalt oxides , lithium iron phosphate, lithium cobalt oxide) and strongly influences voltage, energy density, cycle life, and safety. *CellType* describes the mechanical format (cylindrical, prismatic, pouch). This choice affects thermal and mechanical behavior.

Specification acts as a grouping concept for all datasheet parameters. It includes *ElectricalProperty* (voltage, current, internal resistance), *ThermalProperty* (operating and storage temperature ranges, thermal limits), *MechanicalProperty* (dimensions, mass, casing), *SafetyProperty* (certifications, short-circuit or crush behavior), and *PerformanceProperty* (cycle life, C-rate performance). *ChargeSpecification* and *DischargeSpecification* define allowable charging and discharging conditions, critical for safe operation and degradation modeling. This structured design enables automated extraction, validation, and integration of cell-level knowledge into a comprehensive knowledge graph.

3.2 LLM Triple Extraction Using Chain-of-Thought and Few-Shot Prompting

To extract structured knowledge from battery datasheets, we use a combination of CoT reasoning and few-shot examples. Each prompt provides the model with a concise ontology summary, a text chunk from the PDF datasheet, and illustrative examples of the desired output. This allows the LLM to generate meaningful triples without requiring the full ontology file. As including the full ontology in OWL format would likely consume too many tokens, we instead provide a concise summary that captures all necessary concepts and their relationships through object and data properties :

— **Classes :** BatteryCell, Manufacturer, Specification, ElectricalProperty, ThermalProperty ...
— **Object Properties :** hasManufacturer (BatteryCell → Manufacturer), hasElectricalProperty (BatteryCell → ElectricalProperty)...
— **Data Properties :** hasNominalVoltage (BatteryCell → float), hasNominalCapacity (BatteryCell → float)...

One-Shot Example :

```
Chunk Text: "The 18650 lithium-ion cell
has a nominal voltage of 3.6V and a nominal capacity of 3000mAh."
Triple:
{"Used_text": "The 18650 lithium-ion cell has a nominal voltage of
    3.6V
  and a nominal capacity of 3000mAh",
  "head": "18650 LithiumIonCell",
  "head_type": "BatteryCell",
  "relation": "hasNominalVoltage",
  "tail": "3.6",
  "tail_type": "float"}
```

Prompt Structure :

```
You are a battery datasheet information extraction model.
Your task: extract ONLY technical datasheet information.
RULES:
1. Process text sentence by sentence.
2. Map each entity to a class from the ontology summary.
3. Use only valid relations (object/data properties).
4. Normalize entity names to canonical identifiers.
5. Translate any non-English word to English.
6. Return ONLY JSON triples aligned with the ontology.
Ontology dictionary (classes and relations):
{ontology_summary}
Few-shot examples:
{examples_str}
Text to analyze:
""" {chunk_text} """

Return ONLY a JSON array of valid datasheet triples.
```

3.3 LLM as Judge : Validation Model

The second LLM in our pipeline acts as a **validator** for the triples generated by the first extraction model. Its primary role is to ensure that all output triples :

- Respect the ontology constraints (head type, tail type, and relation must exist in the ontology),
- Accurately reflect information present in the text chunk,
- Follow the required output format, including keys : `Used_text`, `head (Subject)`, `head_type`, `relation (Predicate)`, `tail (Object)`, and `tail_type`.

The validator ensures that only meaningful and correct triples describing battery datasheet information are retained. Specifically, it checks for :

- Electrical properties : voltage, capacity, charge/discharge limits, internal resistance
- Mechanical properties : dimensions, mass
- Materials and components : electrodes, electrolytes, separators
- Cell type and chemistry

Prompt Structure :

```
You are an expert validator for battery datasheet information. Your
    task is to strictly validate candidate triples.

RULES:

1. Keep only triples that describe technical datasheet information.
2. Verify that each entity exists in the text chunk.
3. Verify ontology alignment: head_type, tail_type, and relation must
     exist in ontology.
4. Ignore all non-battery content (authors, references, journals, etc
    .).

Output ONLY a JSON array of objects with keys:
"Used_text", "head", "head_type", "relation", "tail", "tail_type"

Candidate triples:
{candidates}

Text chunk:
""" {chunk_text} """

Ontology dictionary:
{ontology_summary}

Return ONLY the JSON array of validated datasheet triples.
```

4 Experiments

We used multiple PDF datasheets collected by scraping popular manufacturer websites (e.g., energy.panasonic...) and fed them into our pipeline to generate triples. We evaluated

five lightweight models to demonstrate that our approach can provide good results even with smaller models. The tested models are :

— `Qwen/Qwen2-7B-Instruct`
— `mistralai/Mistral-7B-Instruct-v0.3`
— `meta-llama/Llama-3.1-8B-Instruct`
— `Qwen/Qwen3-8B`
— `openai/gpt-oss-20b`

These models are commonly employed in related works. In our approach, we use the same LLM for both extraction and validation tasks, ensuring that any differences in model performance do not influence the evaluation results. The overall pipeline for extracting structured triples from battery datasheets is summarized in Algorithm 1, where $maxnewtokens$ was set to 1000 and $chunksize$ to 500.

Algorithm 1 Ontology Extraction Pipeline from PDF

Require: pdf_file, $ontology\_file$, $model\_name$, $chunk\_size$, max_new_tokens
Ensure: Extracted triplets $Triplets$

$model \leftarrow$ LOAD_MODEL_LLM($model\_name$)
$ontology\_summary \leftarrow$ SUMMARIZE_ONTOLOGY_FOR_PROMPT($ontology\_file$)
$chunks \leftarrow$ CHUNK_PDF_TEXT(pdf_file, $chunk\_size$)
$Triplets \leftarrow []$
for $chunk$ **in** $chunks$ **do**
    $prompt \leftarrow$ EXTRACT_TRIPLETS_COT_WITH2SHOTS_PROMPT($chunk$, $ontology\_summary$)
    $output \leftarrow$ ASK_LLM($prompt$, $model$, max_new_tokens)
    $validation\_prompt \leftarrow$ BUILD_VALIDATION_PROMPT($output$, $chunk$, $ontology\_summary$)
    $validated\_output \leftarrow$ ASK_LLM($validation\_prompt$, $model$, max_new_tokens)
    $Blocks \leftarrow$ PARSE_JSON($validated\_output$) ▷ Empty list if parsing fails
    $Triplets \leftarrow Triplets + Blocks$
end for
return $Triplets$

To evaluate the performance of our approach, we perform two types of validation. First, we assess the quality of the ontology using Competency Questions (CQs). This evaluation checks whether the ontology contains all necessary elements, such as classes and properties, required to construct SPARQL queries that can answer the CQs correctly.

Second, we evaluate the quality of the information extracted by the LLM models. This includes measuring metrics such as the total number of triples extracted and the number of valid triples per PDF. For this evaluation, we use a human sampling method, which, although not fast, provides a reliable and thorough assessment of the extraction quality.

4.1 Results

We first evaluated the quality of the proposed ontology using 24 competency questions provided by battery experts. These CQs are designed to verify whether the ontology contains

all necessary classes, properties, and relationships to answer practical queries about battery cells. Table 1 summarizes the questions along with their answerability and explanations on which ontology elements are used to answer each question.

Q#	Competency Question	Answerable ?	Explanation
1	What is the manufacturer of a specific battery cell ?	Yes	Use `hasManufacturer` linking `BatteryCell` to `Manufacturer`.
2	What is the model name of a battery cell ?	Yes	Use `hasModelName` of `BatteryCell`.
3	What is the nominal voltage and capacity of a battery cell ?	Yes	Use `hasNominalVoltage` and `hasNominalCapacity`.
4	What is the internal resistance of a battery cell ?	Yes	Use `hasInternalResistance`.
5	What is the charge and discharge cutoff voltage ?	Yes	Use `hasChargeCutoffVoltage` and `hasDischargeCutoffVoltage`.
6	What is the operating voltage range ?	Yes	Use `hasOperatingVoltageRangeMin` and `hasOperatingVoltageRangeMax`.
7	What is the standard and fast charge current ?	Yes	Use `hasStandardChargeCurrent` and `hasFastChargeCurrent`.
8	What is the max continuous and pulse discharge current ?	Yes	Use `hasMaxContinuous DischargeCurrent` and `hasPulseDischargeCurrent`.
9	What are the operating temperatures for charge and discharge ?	Yes	Use `hasOperatingTemperature ChargeMin/Max` and `hasOperatingTemperature DischargeMin/Max`.
10	What are the storage temperature limits ?	Yes	Use `hasStorageTemperatureMin` and `hasStorageTemperatureMax`.
11	What are the physical dimensions of a battery cell ?	Yes	Use `Dimension` class with `hasDiameter`, `hasWidth`, `hasHeight`, `hasThickness`, `hasLength`, `hasVolume`.
12	What is the weight of a battery cell ?	Yes	Use `Mass` class with `hasWeight`.
13	What materials are used in the cell ?	Yes	Use `hasMaterial`, `hasCathodeMaterial`, `hasAnodeMaterial`.
14	What is the electrochemistry of the cell ?	Yes	Use `hasElectrochemistry`.
15	What type of cell is it ?	Yes	Use `hasCellType`.
16	What is the cycle life of the cell ?	Yes	Use `CycleLife` class with `hasNumberOfCycles`, `hasEndOfLifeCapacityPercent`, `hasCycleTestTemperature`.
17	What is the C-rate performance of the cell ?	Yes	Use `C_RatePerformance` class with `hasChargeCRate`, `hasDischargeCRate`, `hasCapacityRetentionAtCRate`.

18	Does the cell have safety certifications like UN38.3, IEC62133, UL ?	Yes	Use `hasUN38_3Certification`, `hasIEC62133Certification`, `hasULCertification`.
19	What are the operating and environmental conditions ?	Yes	Use `hasOperatingCondition` and `hasEnvironmentalCondition`.
20	Where can I find the source datasheet PDF ?	Yes	Use `hasSourcePDF`.
21	Can I query all cells from a specific manufacturer ?	Yes	Use `hasManufacturer` to filter `BatteryCell` instances.
22	Can I query cells with specific electrochemistry or material combinations ?	Yes	Use `hasElectrochemistry`, `hasMaterial`, `hasCathodeMaterial`, `hasAnodeMaterial`.
23	Can I find all cells within specific voltage, capacity, or C-rate ranges ?	Yes	Use numeric data properties like `hasNominalVoltage`, `hasNominalCapacity`, and `C_RatePerformance` properties.
24	Can I find cells with a specific certification and safety behavior ?	Yes	Combine certification and safety properties.

TAB. 1 – Competency Questions used to evaluate the battery cell ontology.

We evaluated the performance of five different LLM models on extracting triples from five PDF datasheets (Randomly selected). All models processed each PDF, and the number of extracted triples, as well as validated triples, was manually inspected and verified. Table 2 summarizes the results. It is evident from the results that there are significant differences in

Model	**p1.pdf**		**p2.pdf**		**p3.pdf**		**p4.pdf**		**p5.pdf**	
	Triples	Valid	Triples	Valid	Triples	Valid	Triples	Valid	Triples	Valid
Qwen2-7B-Instruct	34	19	117	30	70	13	62	25	70	17
Mistral-7B	7	1	41	6	26	1	13	0	18	0
Llama-3.1-8B	26	5	120	20	76	16	68	13	66	13
Qwen3-8B	56	29	157	20	81	23	83	31	69	23
gpt-oss-20b	12	7	46	7	21	5	19	2	20	3

TAB. 2 – *Results of extracted and validated triples per PDF for each model.*

both the number of extracted triples and the number of validated triples across the models. It is important to note that we applied a strict validation process : any triple containing incomplete sentences, mischaracterizations, or errors was excluded. Additionally, redundant triples were not counted ; only unique and correctly formed triples were considered in the final validation. This rigorous filtering ensures that the reported numbers accurately reflect the quality and correctness of the extracted knowledge.

5 Conclusion

In this work, we presented a comprehensive ontology-driven pipeline for extracting structured information from unstructured battery datasheets using large language models. By combining a dual-LLM approach with Chain-of-Thought reasoning and few-shot prompting, we provide an effective method for extracting structured knowledge from battery datasheets. Experimental results on multiple PDF datasheets demonstrate significant differences in model performance, highlighting the importance of both the extraction strategy and validation step. Recent models such as Qwen3-8B and Qwen2-7B-Instruct consistently produced more validated triples, confirming that the combination of modern LLMs and ontology guidance is effective for technical knowledge extraction.

Future work will focus on developing an automated method to evaluate the outputs of LLM models, reducing the reliance on manual validation and enabling large-scale, reproducible assessment of extraction quality.

Code Availability

The source code used in this approach is available on GitHub at the following repository : BatIE-OGLLM.

Acknowledgment

This research received partial funding from the French National Research Agency (ANR) under the project "ANR-22-CE92-0007-02". Additionally, support was provided by the European Union through the Horizon Europe program and the innovation program under "GAP-101103667".

Références

Battinfo. `https://github.com/BIG-MAP/BattINFO`. Accessed 26-September-2025.

Ali, H., H. A. Khan, et M. Pecht (2024). Evaluation of manufacturer's low-temperature lithium-ion battery datasheet specifications. *Journal of Energy Storage 91*, 112063, doi: https://doi.org/10.1016/j.est.2024.112063.

Attia, Peter M. Grover, A. J. N. S. K. A. M. T. M. L. Y.-H. C. M. H. C. B. P. N. Y. Z. H. P. K. A. M. H. S. J. B. R. D. E. S. C. W. C. (2020). Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature 578*, 397–402.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, et D. Amodei (2020). Language models are few-shot learners.

Diao, W., C. Kulkarni, et M. Pecht (2021). Development of an informative lithium-ion battery datasheet. *Energies 14*(17).

Doyle, M., T. F. Fuller, et J. Newman (1993). Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *Journal of the Electrochemical Society 140*(6), 1526–1533.

Dreger, M., K. Malek, et M. Eikerling (2025). Large language models for knowledge graph extraction from tables in materials science. *Digital Discovery 4*(5), 1221–1231.

D'Souza, J. et N. Mihindukulasooriya (2024). The state of the art large language models for knowledge graph construction from text : Techniques, tools and challenges. The Knowledge Graph Conference.

Hassini, M., E. Redondo-Iglesias, et P. Venet (2023). Lithium–ion battery data : From production to prediction. *Batteries 9*(7).

Ibrahim Nourhan, Aboulela Samar, I. A. K. R. (2024). A survey on augmenting knowledge graphs (kgs) with large language models (llms) : models, evaluation metrics, benchmarks, and challenges. *Discover Artificial Intelligence 4*.

Ivanisenko, T. V., P. S. Demenkov, et V. A. Ivanisenko (2024). An accurate and efficient approach to knowledge extraction from scientific publications using structured ontology models, graph neural networks, and large language models. *International Journal of Molecular Sciences 25*(21).

Kim, J., Y. Kwon, Y. Jo, et E. Choi (2023). Kg-gpt : A general framework for reasoning on knowledge graphs using large language models.

Kojima, T., S. S. Gu, M. Reid, Y. Matsuo, et Y. Iwasawa (2023). Large language models are zero-shot reasoners.

Lairgi, Y., L. Moncla, R. Cazabet, K. Benabdeslem, et P. Cléau (2024). itext2kg : Incremental knowledge graphs construction using large language models.

Lei, I. T., Z. Zhu, H. Yu, Y. Yao, et Z. Deng (2025). Hint of pseudo code (hopc) : Zero-shot step by step pseudo code reasoning prompting.

Liang, Y., J. Wang, H. Zhu, L. Wang, W. Qian, et Y. Lan (2023). Prompting large language models with chain-of-thought for few-shot knowledge base question generation. In H. Bouamor, J. Pino, et K. Bali (Eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, pp. 4329–4343. Association for Computational Linguistics.

OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin,

D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, et B. Zoph (2024). Gpt-4 technical report.

Santhanagopalan, S. et R. E. White (2006). Online estimation of the state of charge of a lithium ion cell. *Journal of Power Sources 161*(2), 1346–1355.

Singh, A., C. Feltner, J. Peck, et K. I. Kuhn (2021). Data driven prediction of battery cycle life before capacity degradation.

Summary

Automatic information extraction from battery datasheets is critical for accurate modeling and performance management of electrochemical cells. These datasheets, often in unstructured PDF format, contain essential information that is difficult to exploit automatically. This paper introduces a full extraction pipeline leveraging a domain ontology and large language models (LLMs). Our approach uses a dual-LLM pipeline combining Chain-of-Thought reasoning and few-shot examples to extract structured triples from datasheets. A subsequent LLM-based validation step ensures ontology compliance and removes duplicates or errors. We evaluate five LLMs on multiple datasheets and manually verify the extracted triples. Results indicate that recent models achieve higher accuracy and structural consistency, confirming the effectiveness of an ontology-driven approach to transform heterogeneous datasheet content into queryable knowledge.