






# LABKAG at LLMs4OL 2025 Tasks A and C: Context-Rich Prompting for Ontology Construction

Xinyi Zhao<sup>1,\*</sup>, Kevin Drake<sup>2</sup>, Caroline Watanabe<sup>2</sup>, Yuya Sasaki<sup>1</sup>, and  
Hidetaka Hando<sup>2</sup>

<sup>1</sup>Laboro.AI Inc., Tokyo, Japan

<sup>2</sup>CAGLA Inc., Aichi, Japan

\*Correspondence: Xinyi Zhao, [zhao@laboro.ai](mailto:zhao@laboro.ai)

**Abstract.** This paper presents LABKAG’s submission to the LLMs4OL 2025 Challenge, focusing on ontology construction from domain-specific text using large language models (LLMs). Our core methodology prioritizes prompt design over fine-tuning or external knowledge, demonstrating its effectiveness in generating structured knowledge. For Task A (Text2Onto: extracting ontological terms and types), we utilized a locally deployed Qwen3-8B model, while for Task C (Taxonomy Discovery: identifying taxonomic hierarchies), we evaluated the performance of GPT-4o-mini and Gemini 2.5 Pro. Our experiments consistently show that incorporating in-domain examples and providing richer context within prompts significantly enhances performance. These results confirm that well-engineered prompts enable LLMs to effectively extract entities and their hierarchical relationships, offering a lightweight, adaptable, and generalizable approach to structured knowledge extraction.

**Keywords:** Ontology Learning, Large Language Model, Prompt Engineering, In-Context Learning, Entity Extraction, Hierarchical Text Classification

## 1. Introduction

Ontology learning plays a crucial role in knowledge representation, particularly as a precursor to knowledge graph construction. It involves automatically extracting domain-specific entities, relations, and hierarchies from unstructured text to build a structured schema that serves as the conceptual backbone of a knowledge graph. In recent years, Large Language Models (LLMs) have shown great potential in supporting this process, thanks to their ability to extract and generalize knowledge from text without requiring additional training.

Our team, LABKAG, is particularly interested in exploring best practices for applying LLMs to knowledge graph construction and related tasks. While the LLMs4OL 2025 Challenge [1] is framed around ontology learning, its tasks align closely with essential steps in the knowledge graph construction pipeline. We participated in Tasks A and C, both of which simulate key components of extracting structured knowledge from arbitrary text, regardless of domain.

Task A focuses on identifying ontological terms and types from unstructured documents. The objective is to extract domain-specific entities and categorize them appropriately as terms or types. This process is critical in real-world scenarios where structured schemas are needed but do not exist yet, for example, in internal enterprise knowledge bases, or industry-specific documentation. Automatically identifying the foundational vocabulary of a domain helps bootstrap ontologies that can later support the search and query systems.

Task C, in contrast, centers on discovering the taxonomic structure between entities. The goal is to identify is-a relationships among entities, enabling the construction of hierarchies of domain knowledge. These hierarchies are essential for supporting reasoning and inference in graph-driven applications, such as semantic search, product recommendation, and compliance checking. In practice, having a taxonomic structure helps transform flat lists of entities into rich, navigable structures that reflect how concepts are related, improving the utility and interpretability of downstream knowledge systems.

Our goal is to evaluate how effectively LLMs can perform ontology learning tasks under restricted but commonly encountered conditions. We focused on compact models that require minimal computational resources and no additional training. We designed our systems based on two core hypotheses: (1) LLMs are capable of extracting relevant information from limited context, and (2) LLMs can generalize well from a small number of carefully selected examples.

## **2. Related Work**

### **2.1 In-Context Learning and Example Selection**

The in-context learning capability of LLMs, notably exemplified by GPT-3 [2], allows the language models to adapt to new domains and adhere to specific output formats by learning from examples provided within the prompt, without requiring parameter updates. This feature is particularly valuable for ontology construction, a domain-specific task that frequently requires both nuanced knowledge understanding and strictly structured output.

Consequently, prompt engineering has emerged as the key to optimal few-shot performance. As highlighted by Zhao et al. [3], LLM behavior is highly sensitive to various factors related to prompts, including the overall format, the quality and selection of examples, the distribution of labels within those examples, and even their presentation order. A comprehensive survey by Dong et al. [4] reviews key advancements in prompt design and example selection strategies. Notable approaches include leveraging LLMs to verbalize underlying task instructions for enhanced clarity [5], and selecting examples based on their semantic similarity to target test samples [6].

These insights from prior research on in-context learning and prompt engineering directly informed the design principles of our prompts for the ontology construction tasks addressed in this paper, particularly regarding the use of in-domain examples and richer contextual information.

### **2.2 LLM-Assisted Ontology Construction**

The automation of ontology construction has been a long-standing interest. Recent works by Giglou et al. [7] and Funk et al. [8] underscore the significant potential of LLMs in this domain, proposing paradigms for their application in ontology construction.

Beyond the overarching goal of extracting hierarchical structures, LLMs are also useful in providing the fundamental building blocks for ontologies through information extraction. For instance, as an initial phase within their Retrieval-Augmented Generation (RAG) pipeline, GraphRAG [9] showcases the utility of LLMs for extracting both entities and their interrelationships from unstructured text.

Collectively, these foundational works on LLM capabilities for both broad ontology assistance and fine-grained information extraction directly inspired how LLMs are used in our system for extracting entities and taxonomic relationships.

### **3. Task A: Text2Onto**

#### **3.1 Methods**

Task A consists of two subtasks: term extraction and type extraction. We adopted a two-step pipeline. First, we extracted all relevant entities from the input documents, without initially distinguishing between terms and types. Then, we classified each extracted entity as a term, a type, or both. This section details each component of the pipeline.

##### **3.1.1 Entity Extraction**

Entity extraction, as the first step of our method, focuses on identifying all potentially relevant entities from each document. This step does not distinguish between terms and types; instead, all extracted entities are passed to the subsequent classification step.

We explored two prompting strategies for this step. The first, simpler approach was inspired by GraphRAG's prompt for graph extraction from text chunks. While the original prompt targets both entities and relationships, we adapted it to focus solely on entity extraction. Our prompt, shown in Figure 1, consists of a task description and a single example that illustrates the expected output format. The expected response includes a list of entities, each followed by its entity type and description.

The second, more advanced strategy builds on top of the first by utilizing few-shot prompting, where examples are tailored to each test dataset. To select the most relevant examples, we incorporate a retrieval mechanism inspired by Retrieval-Augmented Generation (RAG). This involves selecting a small number of training documents that are semantically similar to most test documents. Given that the training data only provide entity names without associated descriptions, we first task the LLM with generating the complete structured outputs, including entity names, types, and descriptions. These retrieved documents with model-generated outputs are then inserted into the prompt as examples to better guide the model's behavior.

Post-processing in this step was kept minimal. We deduplicated entities that shared both the same type and description. Entities with identical names but different descriptions were retained in order to preserve context, which is important for the classification step that follows.

##### **3.1.2 Entity Classification**

After extracting candidate entities from the documents, the second step of our pipeline involves classifying each entity as a term, a type, or both. This classification establishes the foundation of the ontology: types are the entities that populate the schema, while terms represent domain-specific concepts or instances that fill in the structure.

```
You are assisting in the construction of a domain-specific ontology by
extracting meaningful entities from text. Each entity should represent a concept
relevant to the domain, including both general categories and specific subtypes.

Note that entities can overlap conceptually or hierarchically. Do not exclude
broader or more abstract terms just because more specific variants also appear.

Given a document, extract important domain terms (entities) and assign each one
a generalized type, such as Concept, Process, Unit, Method, Material, Organism,
Product, Ingredient, Category, etc. For each identified entity, extract the
following information:
    • entity_name: Name of the entity
    • entity_type: Type of the entity
    • entity_description: Comprehensive description of the entity's attributes
      and activities
Format each entity as
("entity"|||<entity_name>|||<entity_type>|||<entity_description>)
[EXAMPLE AND REAL DATA OMITTED]
```

**Figure 1.** Prompt used for entity extraction

We performed this classification using a second round of LLM prompting and explored several prompt design strategies, gradually increasing domain specificity and contextual richness. In all strategies, the model was explicitly instructed to assign a term or type label for each entity, based on its usage and meaning in context. During post-processing, entities that serve different functions across different contexts were labeled as both, reflecting their dual role in the ontology.

As for the prompting strategy, we began with a simple few-shot prompt that included a generic task description and out-of-domain examples. The purpose of the examples was to illustrate the expected output format. For each inference, we input a single entity along with its description that was generated in the previous step. However, this generic prompt proved ineffective, as the definition of type and term can vary significantly across domains.

To address this, we designed a more tailored prompt with domain-specific task descriptions and in-domain examples. Definitions were rephrased to align with the domain, as shown in Figure 2. Examples were selected from the corresponding training set to provide clearer signals. While the input format remained compact and consisted only of the entity and its generated description, this strategy provided a more domain-relevant guidance for classification.

Finally, we experimented with a strategy using the full document text as input. In this setup, the model was given the original document along with the complete list of extracted entities and was instructed to classify all entities in a single pass. This approach retained the same domain-specific task description but provided the model with richer context, helping it better understand how each entity appears and functions within its original setting.

### 3.1.3 Term Expansion

In the Engineering subset, we observed consistently low recall in the term extraction subtask. Upon analyzing the training data, we noticed that many of the ground-truth terms do not appear verbatim in the documents. Many of these terms are prefixed units

[SHARED TASK DESCRIPTION OMITTED]

You are assisting in the construction of a domain-specific ontology in the field of engineering. Classify each entity accordingly, and use your understanding of engineering semantics, units, and measurement systems to guide your decisions.

Definitions:

- A specific term refers to a concrete value, expression, or unit used in engineering contexts. These are often numerical ranges (e.g., "0{5}"), quantified values (e.g., "54.3584 on the Kelvin scale"), or precise technical expressions (e.g., "British thermal unit (39 °F)", "centiwatt").
- A category type is a broader engineering concept or unit class that generalizes over multiple specific terms. For example, "energy unit" may include "kilojoule", "calorie", or "British thermal unit", while "temperature scale" may include "Kelvin" or "Celsius".

[EXAMPLE AND REAL DATA OMITTED]

**Figure 2.** Task Description for Engineering Subset

formed by attaching SI prefixes (e.g., milli-, giga-) to base units. While the documents often mention a few examples, the ground-truth term list typically includes the full range of prefixes, from “yocto-” to “yotta-”.

To address this, we introduced an additional term expansion step. We first provide the LLM with the full document and a list of extracted terms. The model identifies which terms are prefixed units. For each identified term, we generate variants by substituting the original prefix with all other prefixes from a predefined list, covering the complete spectrum.

### 3.2 Experimental Setup

All experiments were conducted using the Qwen3-8B model [10], which we deployed and ran locally, with no additional training or parameter updates. Inference was handled through a custom local pipeline, allowing full control over prompt formatting, batching, and output parsing without relying on external APIs.

The datasets used in this work were provided by the organizers of the LLMs4OL 2025 Challenge. Training data was not used to fine-tune the model, but served as a source for constructing example-based prompts and evaluating few-shot strategies. We concentrated our experiments on two subsets: scholarly and engineering. These subsets were selected due to their relatively small number of documents, which enabled faster iteration and easier result analysis during development.

### 3.3 Results and Analysis

We evaluated the performance on Task A by separately measuring the extraction quality of terms and types using precision, recall, and F1 score. We began by evaluating the intermediate output produced by the first step: entity extraction. At this stage, the system outputs a flat list of entities without classification. For evaluation purposes, we used this same list as the candidate set when assessing both term and type extraction quality, treating it as the system’s prediction for each category.

In Table 1, we compare the two prompting strategies we introduced in Section 3.1.1:

- **generic:** A one-shot generic prompt, which uses a general out-of-domain example to illustrate the expected output format.
- **in-domain:** A few-shot prompt tailored to each domain, which uses semantically similar training documents and LLM-generated examples.

**Table 1.** Task A: Performance After Entity Extraction

Subtask	Prompt	Recall (%)	Precision (%)	F1 Score (%)
<b>A1.2-Scholarly</b>	generic	46.88	23.08	30.93
<b>A1.2-Scholarly</b>	in-domain	<b>65.63</b>	<b>34.43</b>	<b>45.16</b>
<b>A1.3-Engineering</b>	generic	45.70	74.63	56.69
<b>A1.3-Engineering</b>	in-domain	<b>46.07</b>	<b>79.50</b>	<b>58.33</b>
<b>A2.2-Scholarly</b>	generic	<b>90.00</b>	41.54	56.84
<b>A2.2-Scholarly</b>	in-domain	<b>90.00</b>	<b>44.26</b>	<b>59.34</b>
<b>A2.3-Engineering</b>	generic	58.33	6.27	11.32
<b>A2.3-Engineering</b>	in-domain	<b>69.44</b>	<b>7.89</b>	<b>14.16</b>

The goal of the entity extraction step is to retrieve as many domain-relevant entities as possible, making recall the primary focus at this stage. As shown in Table 1, "In-Domain" prompts, which use few-shot, in-domain examples, consistently outperform the generic one-shot prompt across all subsets and evaluation metrics. This demonstrates that incorporating domain-specific examples into the prompt is effective for improving the coverage of target entities. The improvement is particularly significant in the A1.2-Scholarly and A2.3-Engineering subtasks, where recall increases from 46.88% to 65.63% and from 58.33% to 69.44%, respectively. These results suggest that these two subtasks may require more in-domain context or specialized vocabulary, and therefore benefit more from tailored prompting.

We then evaluated the final output after applying entity classification. All experiments in this stage were continued using the entity lists generated by "In-Domain" prompts from the previous step, which had demonstrated higher recall and overall performance.

Table 2 compares prompting methods that vary in the richness of entity context, as introduced in Section 3.1.2. For the A1.3 subtask, we additionally report results incorporating the term expansion step described in Section 3.1.3.:

- **description:** A few-shot in-domain prompt, given only the entity and its generated description.
- **full doc:** A few-shot in-domain prompt, given the entity and the original document as context.
- **full doc + term expansion:** Same as "Full Doc," with an additional term expansion step applied after classification.

The goal of the entity classification step is to improve precision while maintaining high recall. As shown in Table 2, prompts with full document context consistently outperform the other in most subtasks. The gains are especially notable in A1.2-Scholarly, where F1 scores increase by over 30%. This suggests that access to richer contextual information allows the model to make more accurate classification decisions. In A1.3-Engineering, the term expansion step significantly boosts recall, leading to a higher overall performance; however, this comes at the cost of reduced precision.

**Table 2.** Task A: Performance After Entity Classification

Subtask	Method	Recall (%)	Precision (%)	F1 Score (%)
<b>A1.2-Scholarly</b>	description	15.63	50.00	23.81
<b>A1.2-Scholarly</b>	full doc	<b>65.62</b>	<b>75.00</b>	<b>70.00</b>
<b>A1.3-Engineering</b>	description	43.88	90.23	59.04
<b>A1.3-Engineering</b>	full doc	43.88	93.02	59.63
<b>A1.3-Engineering</b>	full doc + term expansion	<b>73.86</b>	<b>60.66</b>	<b>66.61</b>
<b>A2.2-Scholarly</b>	description	70.00	41.18	51.85
<b>A2.2-Scholarly</b>	full doc	<b>90.00</b>	<b>77.14</b>	<b>83.08</b>
<b>A2.3-Engineering</b>	description	44.44	25.40	32.32
<b>A2.3-Engineering</b>	full doc	<b>63.89</b>	<b>37.10</b>	<b>46.94</b>

## 4. Task C: Taxonomy Discovery

### 4.1 Methods

Task C focuses on identifying hierarchical relationships within flat lists of domain-specific terms, facilitating the construction of is-a taxonomic structures. The goal is to identify patterns in which one term represents a broader category (parent) and another represents a more specific instance or subcategory (child). Our approach relies on prompt engineering with LLMs, using carefully crafted prompts with explicit instructions and examples to guide the extraction of parent-child relationships.

To address challenges that arose in larger subsets, we introduced two pre-processing strategies to augment the base method: (1) length-based chunking, which splits long lists of terms into shorter, fixed-length segments; and (2) semantic-based grouping, which organizes terms into coherent clusters based on meaning. The following sections describe the motivation and implementation of three resulting pipeline variants.

#### 4.1.1 Single-Step IsA Relationship Extraction

Single-Step IsA Relationship Extraction serves as a straightforward baseline with minimal pre-processing. In this approach, LLMs are explicitly instructed to identify taxonomic (IsA) relationships directly from flat, unstructured lists of domain-specific terms, and to return the output in JSON format. To guide this process, we constructed a fixed prompt with clear instructions and illustrative examples, as shown in Figure 3.

```
You are given a list of terms.
Your task is to identify hierarchical relationships between terms where one is a
more general "parent" and the other is a more specific "child".

Analyze EVERY term and identify SPECIFIC "is-a" relationships.

[INSTRUCTIONS AND REAL DATA OMITTED]
```

**Figure 3.** Example Prompt for Single-Step Extraction

While initially chosen for its simplicity, this method provided a foundation upon which we built more specialized components to address challenges specific to each test subset. We observed relatively low recall, particularly in larger subsets, likely due to input length

limitations and the inefficiency of processing long, unordered lists. To address this, we experimented with several chunking strategies, dividing term lists into manageable batches while preserving context.

#### 4.1.2 Chunking + IsA Extraction

The single-step approach breaks down when applied to long lists of terms, often resulting in inconsistencies and missed IsA relationships due to token limits and the model's inability to maintain context over vast inputs. To address this, we adopted a chunking strategy comprising two conceptual steps:

- **Chunking and Relationship Generation:** The term list is divided into smaller, more manageable chunks, typically ranging from 300 to 500 terms per chunk. Each of these chunks is then independently processed by the LLM using the prompt as the single-step method. Subtask-specific examples are included to provide consistent guidance within each chunk.
- **Aggregation and Normalization:** Within each chunk, parent-child pairs are extracted in a similar fashion to the single-step method, but restricted to the terms present in that specific chunk. The resulting pairs from each chunk are aggregated and normalized during post-processing to form the final relationship set.

While this chunking approach enables efficient scaling of IsA extraction, it introduces certain trade-offs. Because relationships are extracted only within individual chunks rather than across the entire dataset, if a parent and child term appear in separate chunks, the relationship may be missed. This local-only view can lead to reduced recall and an increase in false positives. Our initial experiments confirmed this, showing that recall remained relatively low despite improved handling of longer inputs.

#### 4.1.3 Category Regrouping + Chunking + IsA Extraction

Category Regrouping aims to refine domain-specific terms into semantically coherent groups (e.g., Health & Medical, Travel & Transportation) before applying IsA relationship extraction. This step is motivated by two primary observations: (1) simple length-based chunking often fails to capture valid parent-child relationships when related terms are distributed across chunks; and (2) the source term lists frequently include irrelevant or loosely related terms that are out-of-domain and unlikely to form meaningful taxonomic structures. By category regrouping, we improve the quality of subsequent IsA extraction in two ways: irrelevant terms are filtered out, and intra-group relationships become more coherent, enabling more precise and focused extraction.

We implemented this regrouping step using an additional round of LLM prompting. As shown in Figure 4, the LLM is instructed to classify terms into a predefined set of high-level categories. This serves both to organize the input space and to suggest potential parent terms for the next stage. While this step initially led to lower recall, iterative refinement of the prompt design led to improvements in both recall and precision.



You are a categorization assistant.  
 Classify the following scientific terms into high-level parent categories. Use only from the following standardized parent categories:

[INSTRUCTIONS AND REAL DATA OMITTED]

**Figure 4.** Prompt for Category Regrouping

## 4.2 Experimental Setup

We conducted our experiments using two LLMs: OpenAI's GPT-4o-Mini [11] and Google's Gemini 2.5 Pro [12]. Both were queried with the same input terms to allow for a fair comparison. To ensure deterministic and comparable outputs, we fixed the generation parameters as follows: temperature was set to 0.1, top-p to 3, and top-k to 1.

For each run, we provided a carefully constructed prompt that included few-shot examples and detailed rule-based instructions. The models returned structured outputs representing parent-child relationships, which were then converted into JSON format for analysis.

## 4.3 Results and Analysis

Our experimental results are summarized across several tables. Table 3 presents the best performance achieved for each subtask, detailing the specific model, prompt, and preprocessing strategies employed. This table highlights the overall effectiveness of our best configurations.

For subtask C2, Table 4 compares the impact of using 1-shot versus few-shot prompts. In these specific experiments, regrouping and chunking preprocessing steps were not applied. This set of experiments clearly indicates that providing a greater number of in-context examples generally contributes to improved performance, particularly in simpler experimental settings.

Lastly, Table 5 details the influence of preprocessing strategies, namely regrouping and chunking, on subtasks C2 and C5. The application of the regrouping step, designed to provide a more organized input structure, consistently led to performance improvements compared to experiments without this step, underscoring its contribution to improved output quality.

**Table 3.** Best Performance on Task C

Subtask	Model	Prompt	Regroup	Chunk	Recall (%)	Precision (%)	F1 (%)
<b>C2-MatOnto</b>	Gemini 2.5 Pro	1-shot	✓	✓	47.09	49.71	48.36
<b>C5-SchemaOrg</b>	Gemini 2.5 Pro	1-shot	✓	✓	71.38	59.68	65.01
<b>C7-FoodOn</b>	GPT-4o-mini	1-shot	✓	✓	2.00	2.32	2.15
<b>C8-PO</b>	Gemini 2.5 Pro	1-shot	✓	✓	5.70	2.60	3.57
<b>C9-Blind</b>	Gemini 2.5 Pro	1-shot	✓	✓	3.49	7.95	4.85

**Table 4.** Task C: Comparison of 1-shot and few-shot prompting

Subtask	Model	Prompt	Recall (%)	Precision (%)	F1 (%)
<b>C2-MatOnto</b>	GPT-4o-mini	1-shot	14.96	37.50	21.39
<b>C2-MatOnto</b>	GPT-4o-mini	few-shot	<b>19.67</b>	<b>50.00</b>	<b>28.23</b>

**Table 5.** Task C: Effect of regrouping and chunking

Subtask	Model	Regroup	Chunk	Recall (%)	Precision (%)	F1 (%)
<b>C2-MatOnto</b>	Gemini 2.5 Pro	✗	✓	40.72	48.20	44.14
<b>C2-MatOnto</b>	Gemini 2.5 Pro	✓	✓	<b>47.09</b>	<b>49.71</b>	<b>48.36</b>
<b>C5-SchemaOrg</b>	Gemini 2.5 Pro	✗	✓	45.31	39.20	45.32
<b>C5-SchemaOrg</b>	Gemini 2.5 Pro	✓	✓	<b>71.38</b>	<b>59.68</b>	<b>65.01</b>

## 5. Conclusion and Future Work

This paper presented LABKAG’s participation in the LLMs4OL 2025 Challenge, investigating the efficacy of LLMs for constructing ontologies from domain-specific text. Our work demonstrates that carefully crafted prompt design, without relying on fine-tuning or external knowledge, serves as a powerful and highly effective strategy for structured knowledge acquisition.

Our experiments, utilizing models including Qwen3-8B, GPT-4o-mini, and Gemini 2.5 Pro, consistently showed that prompts incorporating in-domain examples and richer context significantly enhance performance. Conversely, the inclusion of noise, whether in the examples or input, invariably degraded performance. These findings underscore the critical importance of carefully selecting prompting strategies according to the data characteristics, especially in real-world applications.

These results highlight prompt-driven LLM application as a lightweight, generalizable, and adaptable approach to acquiring structured knowledge. However, the reliance on domain-specific in-context data limits its direct applicability to truly "blind" test sets. Future work will therefore focus on adapting our approach to handle unseen domains, and will further explore more advanced techniques for prompt calibration and example selection.

## Data Availability Statement

This work is based solely on the datasets provided by the organizers of the LLMs4OL 2025 Challenge. These datasets are publicly available as part of the official challenge resources. We did not use any additional third-party or proprietary data.

## Underlying and Related Material

Our implementation code and prompt templates are publicly available on GitHub at: <https://github.com/laboro-public/LABKAG-LLMs4OL-2025>

## Author Contributions

**Xinyi Zhao:** Conceptualization, Data Curation, Project Administration, Investigation, Methodology, Visualization, Writing - Original Draft

**Kevin Drake:** Data curation, Investigation, Methodology, Writing – review & editing

**Caroline Watanabe:** Data curation, Investigation, Methodology, Writing – review & editing

**Yuya Sasaki:** Supervision

**Hidetaka Hando:** Supervision

## Competing Interests

The authors declare that they have no competing interests.

## References

- [1] H. Babaei Giglou, J. D’Souza, N. Mihindukulasooriya, and S. Auer, “Llms4ol 2025 overview: The 2nd large language models for ontology learning challenge”, *Open Conference Proceedings*, 2025.
- [2] T. B. Brown et al., *Language models are few-shot learners*, 2020. arXiv: [2005.14165 \[cs.CL\]](https://arxiv.org/abs/2005.14165). [Online]. Available: <https://arxiv.org/abs/2005.14165>.
- [3] T. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, “Calibrate before use: Improving few-shot performance of language models”, in *International Conference on Machine Learning*, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:231979430>.
- [4] Q. Dong et al., *A survey on in-context learning*, 2024. arXiv: [2301.00234 \[cs.CL\]](https://arxiv.org/abs/2301.00234). [Online]. Available: <https://arxiv.org/abs/2301.00234>.
- [5] O. Honovich, U. Shaham, S. R. Bowman, and O. Levy, “Instruction induction: From few examples to natural language task descriptions”, in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 1935–1952. DOI: [10.18653/v1/2023.acl-long.108](https://doi.org/10.18653/v1/2023.acl-long.108). [Online]. Available: <https://aclanthology.org/2023.acl-long.108/>.
- [6] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, “What makes good in-context examples for GPT-3?”, in *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, E. Agirre, M. Apidianaki, and I. Vulić, Eds., Dublin, Ireland and Online: Association for Computational Linguistics, May 2022, pp. 100–114. DOI: [10.18653/v1/2022.deelio-1.10](https://doi.org/10.18653/v1/2022.deelio-1.10). [Online]. Available: <https://aclanthology.org/2022.deelio-1.10/>.
- [7] H. Babaei Giglou, J. D’Souza, and S. Auer, “Llms4ol: Large language models for ontology learning”, in *The Semantic Web – ISWC 2023*, T. R. Payne et al., Eds., Cham: Springer Nature Switzerland, 2023, pp. 408–427, ISBN: 978-3-031-47240-4.
- [8] M. Funk, S. Hosemann, J. C. Jung, and C. Lutz, *Towards ontology construction with language models*, 2023. arXiv: [2309.09898 \[cs.AI\]](https://arxiv.org/abs/2309.09898).
- [9] D. Edge et al., *From local to global: A graph rag approach to query-focused summarization*, 2025. arXiv: [2404.16130 \[cs.CL\]](https://arxiv.org/abs/2404.16130). [Online]. Available: <https://arxiv.org/abs/2404.16130>.
- [10] Q. Team, *Qwen3 technical report*, 2025. arXiv: [2505.09388 \[cs.CL\]](https://arxiv.org/abs/2505.09388). [Online]. Available: <https://arxiv.org/abs/2505.09388>.
- [11] OpenAI, *GPT-4o-mini*, Large Language Model, accessed June 2025, 2024. [Online]. Available: <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>.
- [12] Google, *Gemini 2.5 Pro*, Large Language Model, accessed June 2025, 2025. [Online]. Available: <https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-pro>.