Zero-Shot Learning for Materials Science Texts: Leveraging Duck Typing Principles

Xin Zhang[1], **Peiliang Zhang**[1,3], **Jingling Yuan**[1,2*], **Lin Li**[1]

[1]School of Computer Science and Artificial Intelligence, Wuhan University of Technology, China
[2]Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, China
[3]Department of Library and Information Science, Yonsei University, Korea
{xinz, zhangpl109, yjl, cathylilin}@whut.edu.cn

Abstract

Materials science text mining (MSTM), involving tasks like property extraction and synthesis action retrieval, is pivotal for advancing research by deriving critical insights from scientific literature. Descriptors, serving as essential task labels, often vary in meaning depending on researchers' usage purposes across different mining tasks. (e.g., '`Material`' can refer to both synthesis components and participants in fuel cell experiment). This meaning difference makes it difficult for existing methods, fine-tuned to specific task, to handle the same descriptors in other tasks. To overcome above limitation, we propose MatDuck, a simple and effective approach for Zero-Shot MSTM by evoking material knowledge within Large Language Models (LLMs). Specifically, inspired by the *Duck Typing* principles in programming languages, we present a ClassDefinition-Style Descriptor generation method that evokes task-specific characteristics to address usage variation. Subsequently, we introduce code-style in-context learning for zero-shot tasks, reframing them into code to leverage LLMs' proficiency in code understanding. Extensive experiments on eight benchmark datasets demonstrate that MatDuck, as a plug-and-play approach, significantly improves the Zero-Shot MSTM performance of LLMs by an average of 11.3% across seven tasks.

Introduction

Material Science Text Mining (MSTM) focuses on extracting valuable elements using predefined descriptors from materials science-related texts, which are more challenging than general texts due to their specialized terms, complex structures, and precise information requirements. (Gupta et al. 2022; Song, Miret, and Liu 2023; Song et al. 2023). Key MSTM tasks include named entity recognition, relation extraction, and sentence classification, among others. By applying these, researchers can efficiently organize and interpret vast amounts of material science data, including research papers, patents, and experimental reports (Tshitoyan et al. 2019; Suvarna et al. 2023; Sun et al. 2024; Dagdelen et al. 2024). This paradigm facilitates the identification of critical patterns and relationships in material properties, performance characteristics, and practical applications, driving faster innovation and deployment of new materials.

*Corresponding author.

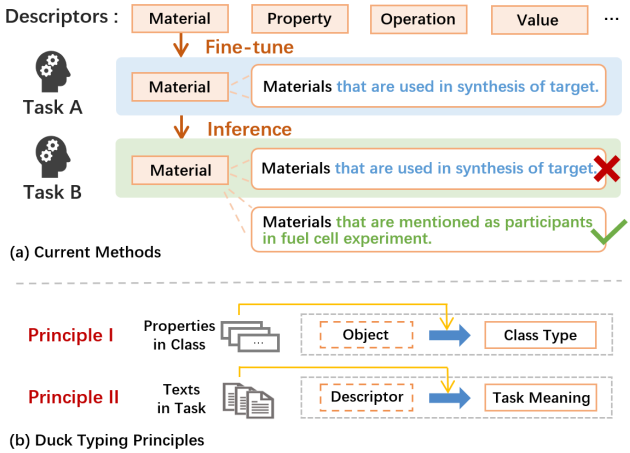

Figure 1: Illustration of meaning differences due to varying usage purposes, and two proposed *Duck Typing* principles.

Descriptors are abstract representations of material properties, structures, or behaviors, enabling data-driven analysis in materials science domain (Isayev et al. 2017; Wyrzykowska et al. 2022; Kabylda et al. 2023). In text mining, they often function as task labels. Researchers typically define descriptors based on task objectives, construct datasets, and fine-tune models accordingly. As a result, the meaning of descriptors can vary depending on their specific requirements in different tasks (Song, Miret, and Liu 2023; Gupta et al. 2024). For instance, the descriptor '`Material`' in materials science can refer to both the materials used in synthesis of target and those mentioned as participants in fuel cell experiment. As shown in Figure 1 (a), when encountering new tasks with the same descriptor but different purposes, variations in usage pose a challenge to the generalizability of specifically trained models. Retraining or fine-tuning often requires high effort and computational cost (Zhao et al. 2024; Xia et al. 2024), making it unaffordable for many material researchers, especially those with no background in natural language processing.

Given this challenge, from a broader research perspective, the meaning difference highlights the urgent need for paradigms that reduce fine-tuning on task-specific labeled data. In this context, Zero-Shot MSTM aims to exploit the embedded material knowledge and powerful generalisabil-

ity of LLMs (Brown et al. 2020; Kojima et al. 2022) to provide greater flexibility and applicability across various tasks without fine-tuning on any task data, enabling researchers to focus on scientific exploration rather than model adaptations. Indeed, leveraging the material knowledge embedded in rapidly evolving LLMs may be a more practical approach than substantial investments in fine-tuning.

To achieve this goal, inspired by the *Duck Typing* principles in programming languages, which emphasizes focusing on an object's behavior rather than its explicit type (e.g., if an object 'looks like a duck, walks like a duck, and quacks like a duck', then it should be treated as a duck), two principles for MSTM are naturally derived: **Principle I**: If the type of an object is unknown, it can be identified by its characteristic properties. **Principle II**: If the meaning of a descriptor is unclear, it can be inferred from its task-related text, as shown in Figure 1 (b). Based on these, we propose leveraging descriptor characteristic properties to enhance the identification of material text objects, and anchoring the meanings of descriptors with task-specific texts, thereby fully utilizing existing LLMs to achieve zero-shot.

In this study, we propose MatDuck, a simple and effective approach for Zero-Shot MSTM, which leverages *Duck Typing* principles to evoke the material knowledge within LLMs. Specifically, following Principle I, we begin by setting the descriptor as the class name and then use LLMs to generate multiple potential class definitions, fully evoking the relevant embedded knowledge. Then, to anchor the most suitable class definitions for the task, we employ the task texts to obtain the fitness degree of each class definition based on Principle II, and retain the top one as the ClassDefinition-Style Descriptor. Finally, we strategically reframe the MSTM tasks into a code format to better leverage LLMs' code understanding, particularly their object-oriented class identification capability, enabling zero-shot learning. Overall, compared with existing methods, our approach can be adapted to different tasks without requiring fine-tuning on task-specific data.

Our contributions can be summarized as follows:

- We propose MatDuck, a user-friendly choice for researchers in the materials domain, which avoids the complexities of data annotation and model fine-tuning while enabling effective MSTM, even without requiring external materials knowledge.
- We achieve zero-shot learning by leveraging *Duck Typing* principles to evoke embedded material knowledge and code understanding capabilities in LLMs, with adaptability to diverse LLMs and tasks.
- Extensive experiments show that MatDuck boosts LLMs' zero-shot performance by 11.3% on average across seven MSTM tasks and eight benchmark datasets, outperforming the best commercial LLMs by using LlaMA3-8B as the backbone.

Related Work

Materials Science Text Mining As a widely studied task, traditional MSTM approaches relied on rule-based systems and manual ontologies (Tshitoyan et al. 2019). The rise of machine learning and natural language processing has led to the adoption of supervised learning, using BERT-based models, such as MatBERT (Walker et al. 2021), MatSciBERT (Gupta et al. 2022), BatteryBERT (Huang and Cole 2022), and MatSci-NLP (Song, Miret, and Liu 2023), to learn vector representations based on labeled data. Recently, with the outstanding text performance of LLMs, researchers have turned their attention to the materials domain LLMs (Song et al. 2023; Dagdelen et al. 2024). However, these methods heavily depend on task-specific data, resulting in limited generalisability. Meanwhile, the trend towards zero-shot learning has been highlighted as particularly beneficial in materials science. To achieve this, LLaMA models were adopted and fine-tuned on the generic material data. (Song et al. 2023). Despite performance gains, this method still heavily depends on extensive fine-tuning, limiting its efficiency compared to the rapid iteration of LLMs. Thus, it is necessary to further explore plug-and-play approaches that do not require fine-tuning but fully utilize the material knowledge embedded within LLMs.

Zero-Shot with LLMs Zero-shot learning with LLMs has shown promising results, particularly in text-related tasks. In this approach, LLMs are adopted to perform tasks without fine-tuning task-specific datasets. Research has shown that LLMs like GPT-3 can execute zero-shot text classification by using natural language prompts. (Brown et al. 2020). Further research emphasizes the critical role of In-Context Learning (ICL), where prompt engineering plays a crucial role in guiding LLMs to effectively utilize their pre-trained knowledge. (Min et al. 2022; Liu et al. 2023; Lin et al. 2023; Wang et al. 2023a). Chain-of-Thought prompting further enhances reasoning by structuring tasks into intermediate steps (Wei et al. 2022; Kojima et al. 2022; Wang et al. 2023b). These methods significantly boost LLM performance in zero-shot scenarios, highlighting the potential of ICL. Meanwhile, there is increasing interest in integrating code style into LLMs applications for text tasks (Li et al. 2024; Sainz et al. 2023; Wang, Li, and Ji 2023), using programming constructs to structure prompts, thereby leveraging the model's code ability to handle complex textual data. However, fine-tuning is still required. Unlike previously mentioned methods, to the best of our knowledge, we are the first to explore task-agnostic ICL for Zero-Shot MSTM, without requiring any fine-tuning or external materials knowledge.

Method

MatDcuk is designed to achieve Zero-Shot MSTM by enhancing ICL using *Duck Typing* principles. As shown in Figure 2, it encompasses three primary parts: Class Definition Generation, ClassDefinition-Style Descriptor Adaptation, and Code-Style Task Inference.

Problem Formulation

Similar to the works of (Song, Miret, and Liu 2023; Song et al. 2023), we formalize the mining tasks as classification tasks, as materials science places a high emphasis on data accuracy, and classification offers greater precision compared

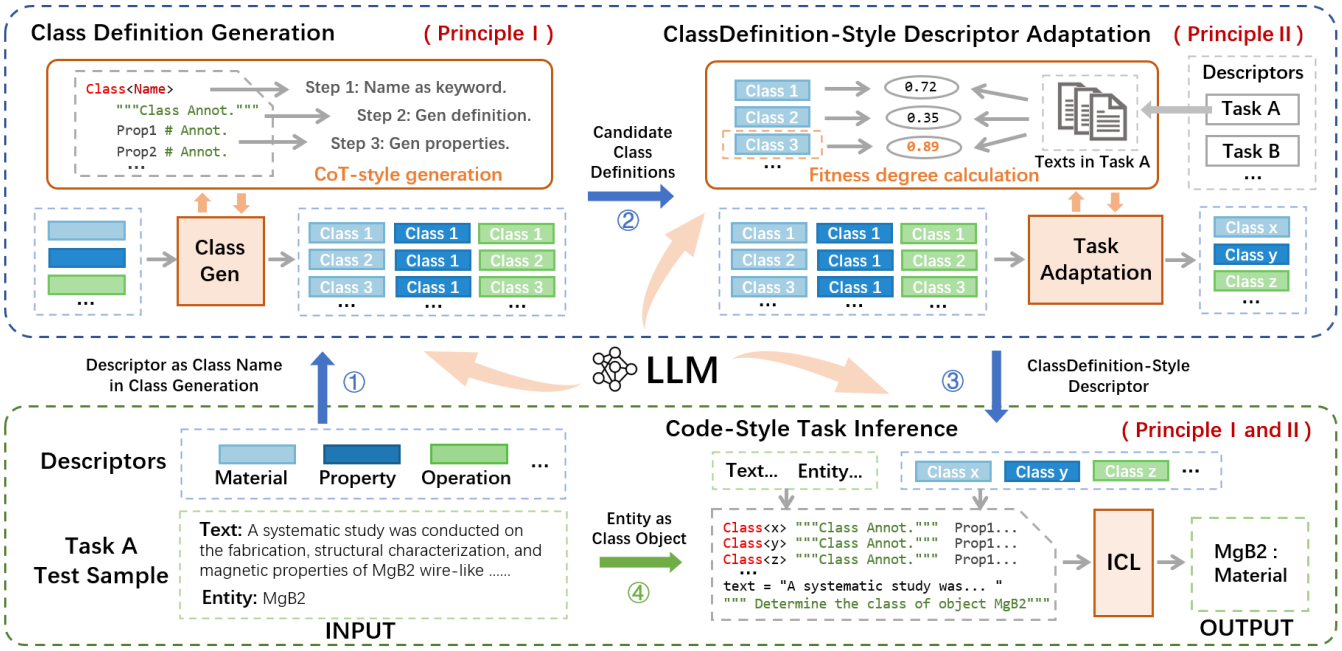

Figure 2: The framework of MatDuck, a task-agnostic ICL approach for Zero-Shot MSTM by leveraging *Duck Typing* principles (NER task as an example). MatDuck aims to fully leverage the materials knowledge and code capabilities within LLMs.

to other generative approaches. In this context, a simplified definition of the tasks, using Named Entity Recognition (NER) as an example, is as follows : Given an entity e mentioned within a contextual text T and a set of descriptors $D = \{d_1, d_2, ..., d_m\}$, the goal is to identify the most appropriate descriptor from D for the entity e based on the context provided in T. It is worth noting that in some tasks, such as sentence or paragraph classification, the x represents not an entity in the text but the contextual text T itself.

Class Definition Generation

Different usage purposes for the same descriptor lead to variations in its conveyed meaning, making it challenging to apply the descriptor name alone to new tasks. Inspired by the **Principle I**, we consider anchoring descriptor meanings in different tasks in the form of code class definitions.

However, current methods for constructing class definitions (Li et al. 2024; Sainz et al. 2023) often rely on combining templates with structured knowledge from external knowledge bases, which is unsuitable for materials science domain where access to accurate task knowledge is limited. In this paper, we use LLMs to construct class definitions, leveraging their embedded knowledge to overcome above limitation.

Specifically, instead of directly generating class definitions, we adopt the Chain-of-Thought (Wei et al. 2022) approach to produce more precise definitions. Given a set descriptors D, for each d, due to the diversity of descriptors in materials science, [**Step 1**] we use the descriptor name as a keyword in the prompts, and then [**Step 2**] generate a diverse set of definitions as class annotations $C^d = \{c_1^d, c_2^d, ..., c_h^d\}$ based on the multiple meanings of descriptor d :

$$C^d = \text{Generator}(\text{T}(d)) \tag{1}$$

where LLMs is selected as the Generator and $\text{T}(\cdot)$ denotes the instruction template. After that, For each possible meaning of the descriptor, [**Step 3**] descriptor d and class annotation $\text{c}_i^d \in C^d$ are adopted as the foundation for constructing descriptor-specific instruction to generate a set of properties and corresponding annotations $(P, C^p)_i = \{(p_1, c_1^p)_i, (p_2, c_2^p)_i, ..., (p_k, c_k^p)_i\}$:

$$(P, C^p)_i = \text{Generator}(\text{T}(d, \text{c}_i^d)) \tag{2}$$

Finally, the generated class annotation $(P, C^p)_i$ are used to construct the descriptor class definitions Γ^d based on predesigned templates $\text{Class}(\cdot)$:

$$\gamma_i = \text{Class}(d, \text{c}_i, (P, C^p)_i) \tag{3}$$

where $\gamma_i \in \Gamma$, Γ denotes the set of class definitions of d.

In this way, for each meaning of the descriptor d, we obtain the corresponding class definitions Γ, each with sufficient characteristic properties to anchor the exact meaning of the descriptor.

ClassDefinition-Style Descriptor Adaptation

To effectively mitigate the meaning gap between descriptors and specific tasks, we further propose a ClassDefinition-Style Descriptor adaptation method, which aims to identify the distribution of class objects within the task texts, thereby determining the most relevant class definitions as the ClassDefinition-Style Descriptors for the tasks.

Specifically, for descriptor d, given a generated set of class definitions $\{\gamma_1, \gamma_2, ..., \gamma_h\}$ and a set of texts $\{T_1, T_2, ..., T_r\}$

Algorithm 1: ClassDefinition-Style Descriptors Generation

Input: Descriptors of TaskA $D = \{d_1, d_2, \ldots, d_m\}$, Texts of TaskA $T = \{t_1, t_2, \ldots, t_r\}$
Output: ClassDefinition-Style Descriptors of TaskA D^c

$D^c \leftarrow \{\}$
for each descriptor $d \in D$ **do**
  Generate multiple class annotations C^d
  for each class annotation $c_i^d \in C^d$ **do**
    Generate multiple property&annotations $(P, C^p)_i$
    Define class γ_i consisting of d, c_i^d, and $(P, C^p)_i$
    Initialize object counter $Obj[\gamma_i] \leftarrow 0$
    Form a set $S_i = \{(d, c_i^d)\} \cup \{(P, C^p)_i\}$
    for each task text $t \in T$ **do**
      Count objects of elements in S_i within t and update $Obj[\gamma_i]$
    end for
    Obtaining the fitness degree $\Phi_{S_i} \leftarrow sum(Obj[\gamma_i])$
  end for
  Select the class definition $\gamma_c \leftarrow \gamma_i \arg\max \Phi_{S_i}$
  $D^c \leftarrow D^c \cup \{\gamma_c\}$
end for
return D^c

randomly selected from the texts of current task. We aim to follow **Principle II** to calculate the distribution of instance relationships between class definitions and potential objects, determining their fitness to specific tasks:

$$\boldsymbol{S}_i = \{(d, c_i^d)\} \cup \{(P, C^p)_i\} \tag{4}$$

$$\Phi_{S_i} = \sum_{j=1}^{k} \sum_{s \in S_i} \text{Count}(\text{Instance}(s, T_j)) \tag{5}$$

where Instance(a, b) denotes the set of instances of class a that may be present in b, which is achieved using LLMs' class identification capabilities through prompts. Then, the class definition with the best fitness degree Φ is selected as the ClassDefinition-Style Descriptor γ:

$$\gamma = \gamma_{i^*} \quad \text{where} \quad i^* = \arg\max_i \Phi_{S_i} \tag{6}$$

The pseudo-code for complete ClassDefinition-Style Descriptors generation procedure is presented in Algorithm 1.

Code-Style Task Inference

After obtaining task-specific ClassDefinition-Style Descriptors, our goal is to effectively introduce them to enhance ICL for zero-shot learning.

To achieve this, we introduce a Code-Style Task Inference method that enhances ICL by leveraging the object-oriented class identification in code understanding capability of LLMs (Yang et al. 2024). This method integrates task constraints and descriptor class definitions into a code-style format for instance-based judgment. Specifically, the Python programming language is employed to represent the MSTM tasks. We wrap the object to be recognized e, the background text T, and the set of descriptors $\{\gamma_1, \gamma_2, ..., \gamma_m\}$ into a code style prompt and output with a class type.

Natural Language Style ICL（NER）

```
This is a material science literature text:
"A systematic study was conducted on the fabrication, structural
characterization, and magnetic properties of MgB2 ..... "

Please tell me the entity type of "MgB2" in the literature text,
all entity types are in Options. Options: Material, Element,...
```

Code Style ICL（NER）

```
class Entity :
    def __init__(self, name: str):
        self.name = name

# The following entity classes describe the entity types
class Material(Entity):
    """ A substance that is used to create a specific structure
or morphology, such as ... """
    def __init__(self, name: str):
        super().__init__(name=name)
        self.density  # The mass per unit volume of the ...
        self.size  # The dimensions or scale of the material...
        self.shape  # The geometric form or morphology of the ...
        ...
class Element(Entity):
...

# This is a material science literature text
text = " A systematic study was conducted on the fabrication,
structural characterization, and magnetic properties of MgB2... "
"""
Based on above classes, please determine which class the object
"MgB2" in the above text belongs to.
"""
```

Figure 3: The Code-Style ICL (NER task as an example).

Task	#Samples	#Descriptors	Misuse Rate
NER	112,191	41	0.439
RC	25,674	19	0.316
EAE	6,566	7	-
PC	1,500	1	-
SAR	5,547	8	-
SC	9,466	1	-
SF	8,253	18	0.333
Total	169,197	95	0.316

Table 1: The detailed statistics of datasets and the misuse rate of descriptors.

As shown in Figure 3, with the NER task as an example, we define the descriptor as a sub-class of task class, and represent the task constraints through parameters. The input background text is then assigned as a string to the variable "$Text$" and specified with an annotation. The task goal is described as instance class type judgment in the form of code annotation.

Query Execution Complexity Analysis

In materials science text mining, large volumes of data (including documents and reports) often need to be processed. Although some zero-shot methods do not require fine-tuning, they still necessitate numerous LLM queries for a single instance (Zhuang et al. 2024; Chen et al. 2024), which can significantly impact execution time and computational costs. Therefore, we will analyze the complexity of query execution in MatDuck.

Model	Named Entity Recognition	Relation Extraction	Event Argument Extraction	Paragraph Classification	Synthesis Action Retrieval	Sentence Classification	Slot Filling	Overall All Tasks
Fine-tuning Performance								
MatSciBERT	0.707	0.791	0.436	0.719	0.692	0.914	0.436	0.671
(Gupta et al. 2022)	0.470	0.507	0.251	0.623	0.484	0.660	0.194	0.456
MatBERT	0.875	0.804	0.451	0.756	0.717	0.909	0.548	0.722
(Walker et al. 2021)	0.630	0.513	0.288	0.691	0.594	0.614	0.273	0.517
Zero-Shot Performance								
Chat-GPT	0.444	0.572	0.451	0.626	0.672	0.384	0.360	0.501
(OpenAI 2022)	0.309	0.525	0.421	0.626	0.561	0.354	0.274	0.439
LLaMA3-8B	0.405	0.576	0.499	0.692	0.722	0.495	0.409	0.543
(Dubey et al. 2024)	0.306	0.552	0.465	0.691	0.598	0.432	0.310	0.480
GPT-4	0.497	0.549	0.516	0.708	0.745	0.354	0.529	0.557
(OpenAI 2023)	0.404	0.526	0.478	0.706	0.641	0.334	0.398	0.511
Claude3.5	0.556	0.613	0.522	0.624	0.817	0.406	0.551	0.584
(Anthropic 2024)	0.480	0.606	0.487	0.624	0.734	0.370	0.402	0.533
Honybee-7B	0.267	0.245	0.290	0.490	0.688	0.490	0.393	0.409
(Song et al. 2023)	0.190	0.178	0.189	0.343	0.342	0.365	0.289	0.271
Honybee-13B	0.429	0.412	0.481	0.611	0.801	0.589	0.578	0.557
(Song et al. 2023)	0.372	0.346	0.378	0.467	0.429	0.503	0.423	0.417
MatDuck	0.470	0.611	0.528	0.748	0.785	0.853	0.411	0.629
(LLaMA3-8B)	0.355	0.520	0.488	0.735	0.762	0.573	0.330	0.538
MatDuck	0.564	0.579	0.538	0.768	0.851	0.817	0.516	0.661
(GPT4)	0.472	0.503	0.497	0.755	0.828	0.596	0.401	0.579
MatDuck	0.627	0.639	0.544	0.758	0.874	0.885	0.554	0.697
(Claude3.5)	0.512	0.587	0.508	0.757	0.859	0.576	0.437	0.605

Table 2: Zero-shot and fine-tuning evaluation results for MSTM tasks. We present macro-F1 (top) and micro-F1 (bottom) scores, with the best, second-best and third-best zero-shot performing approachs highlighted.

Preparation Phase: The query number during the definition process of m anchored descriptor classes (as described in the Eqs. 1-6):

$$Q^{pred} = m(h(1+k) + hr(1+k)) \quad (7)$$

Interfence Phases: The query number during the inference process for n samples to be classified.

$$Q^{intfer} = n \quad (8)$$

Total Count: The total query number for n samples:

$$Q^{total} = Q^{pred} + Q^{intfer} = n + mh(1+r)(1+k) \quad (9)$$

It can be observed that total query number is mainly related to the sample size n. The complexity of the query execution is considered acceptable.

Experiments

In this section, we conduct extensive experiments to address the following questions to evaluate the effectiveness of the proposed MatDuck: **RQ1**: How does MatDuck perform on Zero-Shot MSTM? Compared to existing zero-shot methods and fine-tuning methods? **RQ2**: How effective is the ClassDefinition-Style Descriptor in capturing task-specific characteristics (**Principle I**)? **RQ3**: How effective is the task adaptation in selecting ClassDefinition-Style Descriptors for specific tasks (**Principle II**)? **RQ4**: How effective is code-style ICL compared to natural language for zero-shot performance improvement?

Dataset

To evaluate the performance of our proposed MatDuck for Zero-Shot MSTM, we conduct experiments on eight public materials science text benchmark datasets (Song, Miret, and Liu 2023), where the texts included are all materials science domain literature. Tasks include Named Entity Recognition (NER), Relation Extraction (RE), Event Attribute Extraction (EAE), Paragraph Classification (PC), Synthesis Action Retrieval (SAR), Sentence Classification SC), and Slot Filling (SF). Further details about the tasks and datasets can be found in the work of (Song, Miret, and Liu 2023).

Additionally, we calculated the descriptor misuse rate, defined as the proportion of the same descriptors applied across multiple research contexts (tasks or datasets). As shown in Table 3, the overall misuse rate reaches **31.6%** and **43.9%** in the NER task. This misuse rate highlights the necessity of anchoring the meanings of descriptors.

Baselines and Evaluation Metrics

To validate the competitiveness, we compare the proposed MatDuck variants with the following baselines:

API-accessible LLMs, such as GPT4 (OpenAI 2023), GPT3.5 (OpenAI 2022), and Claude3.5 (Anthropic 2024), represent the state-of-the-art in general-purpose models. Due to the limited availability of open-source options, we conducted our evaluation through API access.

Model	Named Entity Recognition	Relation Extraction	Event Argument Extraction	Paragraph Classification	Synthesis Action Retrieval	Sentence Classification	Slot Filling	Overall All Tasks
MatDuck	**0.627**	**0.639**	**0.544**	**0.758**	**0.874**	**0.885**	**0.554**	**0.697**
(Claude3.5)	**0.512**	**0.587**	**0.508**	**0.757**	**0.849**	**0.576**	**0.437**	**0.605**
w/o	0.565	0.622	0.526	0.592	0.804	0.396	0.517	0.575
Property	0.412	0.599	0.481	0.592	0.723	0.365	0.385	0.508
w/o	0.593	0.619	0.502	0.632	0.826	0.591	0.526	0.613
Task-Adapt	0.481	0.565	0.459	0.630	0.729	0.529	0.391	0.541
w/o	0.608	0.583	0.510	0.688	0.848	0.656	0.535	0.633
Code-Style	0.489	0.529	0.477	0.687	0.751	0.539	0.419	0.556

Table 3: Zero-shot evaluation results of MatDuck in ablation experiments.

Open-source LLMs, such as LLaMA3 (Dubey et al. 2024), represent the competitive open-source LLMs. Considering the performance improvement validation and resource constraints, we will concentrate on the LLaMA3-8B.

Material-domain LLMs, including HoneyBee-7B and HoneyBee-13B (Song et al. 2023), are state-of-the-art billion-parameter models in MSTM, achieving zero-shot performance improvements through fine-tuning on materials science instruction data.

Fine-tuned BERTs, such as MatBERT (Walker et al. 2021), MatSciBERT (Gupta et al. 2022), are BERT (Devlin 2018) variants pre-trained on materials science task-specific labeled data, achieving outstanding performance based on the supervised information.

Evaluation Metrics: We evaluate the performance by using two widely adopted metrics: Micro-F1 and Macro-F1, consistent with existing work (Song, Miret, and Liu 2023; Song et al. 2023).

Implementation Details

As MatDuck is a plug-and-play method, we evaluated its performance using high-performance API-accessible models such as GPT-4 and Claude3.5, as well as open-source models like LLaMA3-8B as backbone models. Specifically, the GPT-4 version used is GPT-4-Turbo, Claude3.5 is Claude3.5 Sonnet, and LLaMA3-8B was evaluated on two NVIDIA RTX- 4090 GPUs. The generators in the MatDuck variants are the foundational LLMs on which they are based. In class definition generation and adaptation, we set the number of properties to 5, and in the class definition selection, the number of random task texts selected is 10% of the total texts. For some datasets, fewer texts may be needed to select suitable ClassDefinition-Style Descriptors, and adjustments can be made based on the task dataset's distribution. Our experimental setup and baseline results for the BERT series and Honeybee series methods follow the (Song, Miret, and Liu 2023; Song et al. 2023). The source code is available at https://github.com/xinzcode/MatDuck.

Experimental Results

Comparison with Zero-Shot Baselines (RQ1) As shown in the results in Table 2, existing general-purpose LLMs demonstrate objective performance in Zero-Shot MSTM tasks, which can be attributed to the extensive material domain text included in their pre-training datasets and their notable generalization capabilities in text tasks. Among these, Claude3.5 exhibits the best overall performance across multiple tasks. Additionally, although LLaMA3-8B performs slightly lower than Claude3.5 and GPT4, its advantages as an open-source model make this trade-off acceptable. HonnyBee, as the method fine-tuned for materials science based on the original LLaMA-7B, shows limited advantages in zero-shot performance compared to the rapidly evolving LLMs. This indicates that leveraging the material knowledge embedded in contemporary LLMs might be a more effective strategy compared to extensive investments in domain-specific fine-tuning. Our proposed approach, utilizing Claude3.5, GPT4, and LLaMA3-8B, significantly outperforms both existing general-purpose LLMs and material-domain LLMs in zero-shot performance, achieving improvements of **8.6%, 10.4%**, **11.3%**, and **14.0**%, respectively. This finding effectively validates the effectiveness of MatDuck, which enhances Zero-Shot MSTM by applying *Duck Typing* principles to characterize descriptors and improve ICL. Additionally, it is worth noting that the MatDuck-enhanced LLaMA3-8B outperforms the top general-purpose models, GPT-4 and Claude3.5, by **7.2%** and **4.5%**. This makes it a viable option for researchers with open-source or low-cost requirements.

Comparison with Fine-Tuned Baselines (RQ1) Compared to the zero-shot approach, the fine-tuned BERTs achieve considerable performance on certain tasks, with **0.914** F1-score on the SC task, making BERT-based models a viable choice when task labeled data is available. Furthermore, MatDuck demonstrates competitive performance as a zero-shot method compared to these approaches. With Claude3.5 as its backbone, MatDuck achieves **0.697** F1-score, surpassing the fine-tuned method MatSciBERT. This reflects that MatDuck, when combined with general-purpose LLMs, can reach a level comparable to supervised learning and highlights the promising potential of leveraging material knowledge within LLMs for Zero-Shot MSTM.

Ablation Study To analyze the contributions of different components within the proposed MatDuck, we conducted ablation experiments with the following variants, using Claude3.5 as the backbone:

- **w/o Properties (RQ2)**: It means that descriptor class def-

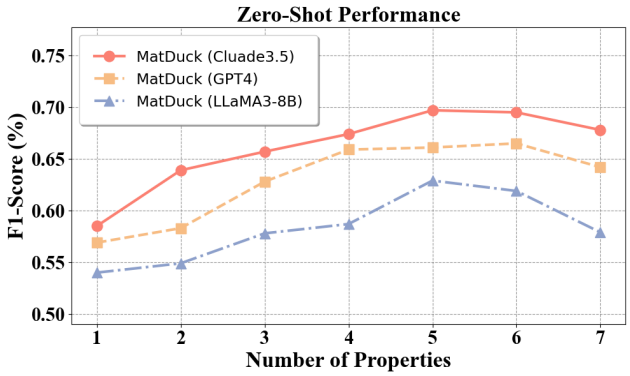

Figure 4: Performance with different number of properties.

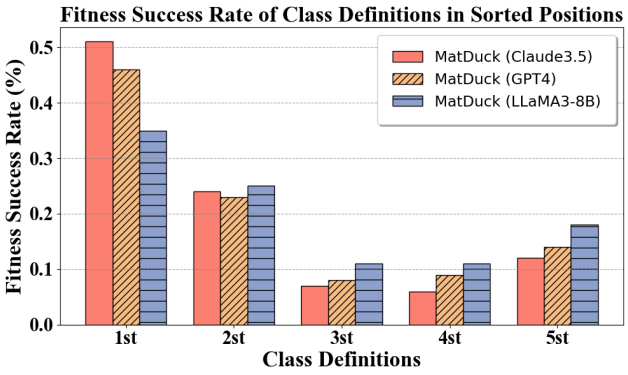

Figure 5: True fitness success rate of sorted class definitions.

initions do not incorporate task-specific characteristic information, such as properties and annotations.

- **w/o Task-Adapt (RQ3)**: It means that generating ClassDefinition-Style Descriptor directly for tasks, without task adaptation.
- **w/o Code-Style (RQ4)**: It means employing natural language for tasks rather than converting them into code format within ICL.

The results of the ablation experiments are shown in Table 3. Performance drops significantly to 0.575 when task-specific properties and annotations are excluded from descriptor class definitions, falling below the original LLM baseline. This decline likely occurs because, without these characteristics, class definitions include only descriptor names and structures, making descriptor-related information less prominent in the context, leading to reduced performance. This highlights the importance of descriptor characteristics in class definitions, consistent with Principle I. Additionally, when task adaptation is not employed, performance also declines significantly. This may be attributed to the gap between directly generated descriptor class definitions and the intended task, validating the necessity of Principle II. While using natural language style ICL, the lack of knowledge aggregation from class definitions and the limited use of LLM code understanding capabilities restrict performance improvements, though it still benefits from incorporating task-specific knowledge.

Performance with Different Number of Properties (RQ2) To further assess the effectiveness of the proposed ClassDefinition-Style Descriptors in MatDuck, we will evaluate how varying the number of generated class definition properties affects performance. This approach is based on the intuition that more properties enhance the anchoring effect of class definitions on material knowledge, as outlined in Principle I. As shown in Figure 4, we evaluate the performance of MatDuck with different backbones across varying numbers of properties. The results indicate a positive correlation between the number of properties and performance across different backbones. Generally, performance improves as the number of properties increases, with the peak observed at 4, 5, and 6 properties. However, as the number of properties continues to rise, the benefits start to diminish. This decline might be due to an overabundance of properties, which can reduce the relative importance of descriptor-specific information and lead to negative effects from overly long contexts. Overall, the impact of property number on performance validates the importance of Principle I for Zero-Shot MSTM, which emphasizes the role of properties in enhancing the identification of descriptors.

Effectiveness Analysis of Task Adaptation (RQ3) To analyze the proposed task adaptation method, which aims to select the best-fit ClassDefinition-Style Descriptors for specific tasks by leveraging Principle II, we analyzed the fitness success rate for class definitions with different fitness degrees. This rate represents the probability that a class definition at each position in the ranking results is the true optimal class definition. In this way, the effectiveness of task adaptation can be validated. The results presented in Figure 5 show that, in most cases, the optimal choices frequently appear in the top two positions of the ranking results based on the task adaptation method. This validates the effectiveness of Principle II in determining descriptor meanings according to the task texts, also highlights the necessity of addressing potential meaning differences of descriptors caused by usage purpose variation.

Conclusion

In this paper, we propose a simple and effective approach, MatDuck, for Zero-Shot MSTM by leveraging *Duck Typing* principles. MatDuck aims to enhance in-context learning by leveraging material knowledge and code capabilities within LLMs under the guidance of two *Duck Typing* principles. As a plug-and-play solution, MatDuck can be applied to various rapidly evolving LLMs without adjustments or fine-tuning, making it user-friendly for materials researchers without an NLP background. Extensive experiments across seven datasets and eight datasets validate the Zero-Shot performance of MatDuck, showing an average improvement of 11.3% without requiring fine-tuning or external materials knowledge bases. In future work, we will further explore material knowledge and capabilities within LLMs to advance the field of materials science.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.62472332, No.62276196), the "Open Bidding for Selecting the Best Candidates" Project of Wuhan East Lake High-Tech Development Zone (No.2024KJB322), and the Hubei Provincial International Science and Technology Cooperation Project (No.2024EHA031).

References

Anthropic. 2024. Claude 3.5 Sonnet. https://www.anthropic.com/claude/sonnet. Accessed: 2024-08-15.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, X.; Chen, X.; He, B.; Wen, T.; and Sun, L. 2024. Analyze, generate and refine: Query expansion with LLMs for zero-shot open-domain QA. In *Findings of the Association for Computational Linguistics ACL 2024*, 11908–11922.

Dagdelen, J.; Dunn, A.; Lee, S.; Walker, N.; Rosen, A. S.; Ceder, G.; Persson, K. A.; and Jain, A. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1): 1418.

Devlin, J. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783v2*.

Gupta, A.; Mittal, D.; Goel, O.; and Jha, S. K. 2024. Natural language processing algorithms for domain-specific data extraction in material science: Reseractor. *Journal of Materials Science*, 1–17.

Gupta, T.; Zaki, M.; Krishnan, N. A.; and Mausam. 2022. MatSciBERT: A materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(1): 102.

Huang, S.; and Cole, J. M. 2022. BatteryBERT: A pretrained language model for battery database enhancement. *Journal of chemical information and modeling*, 62(24): 6365–6377.

Isayev, O.; Oses, C.; Toher, C.; Gossett, E.; Curtarolo, S.; and Tropsha, A. 2017. Universal fragment descriptors for predicting properties of inorganic crystals. *Nature communications*, 8(1): 15679.

Kabylda, A.; Vassilev-Galindo, V.; Chmiela, S.; Poltavsky, I.; and Tkatchenko, A. 2023. Efficient interatomic descriptors for accurate machine learning force fields of extended molecules. *Nature communications*, 14(1): 3562.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Li, Z.; Zeng, Y.; Zuo, Y.; Ren, W.; Liu, W.; Su, M.; Guo, Y.; Liu, Y.; Li, X.; Hu, Z.; et al. 2024. KnowCoder: Coding Structured Knowledge into LLMs for Universal Information Extraction. *arXiv preprint arXiv:2403.07969*.

Lin, H.; Yi, P.; Ma, J.; Jiang, H.; Luo, Z.; Shi, S.; and Liu, R. 2023. Zero-shot rumor detection with propagation structure via prompt learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 5213–5221.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35.

Min, S.; Lyu, X.; Holtzman, A.; Artetxe, M.; Lewis, M.; Hajishirzi, H.; and Zettlemoyer, L. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 11048–11064.

OpenAI. 2022. Introducing ChatGPT. https://openai.com/index/chatgpt. Accessed: 2024-08-15.

OpenAI. 2023. GPT-4: Technical report. https://www.openai.com/research/gpt-4. Accessed: 2024-08-15.

Sainz, O.; García-Ferrero, I.; Agerri, R.; de Lacalle, O. L.; Rigau, G.; and Agirre, E. 2023. Gollie: Annotation guidelines improve zero-shot information-extraction. *arXiv preprint arXiv:2310.03668*.

Song, Y.; Miret, S.; and Liu, B. 2023. MatSci-NLP: Evaluating Scientific Language Models on Materials Science Language Tasks Using Text-to-Schema Modeling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3621–3639.

Song, Y.; Miret, S.; Zhang, H.; and Liu, B. 2023. HoneyBee: Progressive Instruction Finetuning of Large Language Models for Materials Science. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 5724–5739.

Sun, L.; Han, Y.; Zhao, Z.; Ma, D.; Shen, Z.; Chen, B.; Chen, L.; and Yu, K. 2024. Scieval: A multi-level large language model evaluation benchmark for scientific research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19053–19061.

Suvarna, M.; Vaucher, A. C.; Mitchell, S.; Laino, T.; and Pérez-Ramírez, J. 2023. Language models and protocol standardization guidelines for accelerating synthesis planning in heterogeneous catalysis. *Nature Communications*, 14(1): 7964.

Tshitoyan, V.; Dagdelen, J.; Weston, L.; Dunn, A.; Rong, Z.; Kononova, O.; Persson, K. A.; Ceder, G.; and Jain, A. 2019. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763): 95–98.

Walker, N.; Trewartha, A.; Huo, H.; Lee, S.; Cruse, K.; Dagdelen, J.; Dunn, A.; Persson, K.; Ceder, G.; and Jain, A. 2021. The impact of domain-specific pre-training on named entity recognition tasks in materials science. *Available at SSRN 3950755*.

Wang, L.; Li, L.; Dai, D.; Chen, D.; Zhou, H.; Meng, F.; Zhou, J.; and Sun, X. 2023a. Label Words are Anchors: An

Information Flow Perspective for Understanding In-Context Learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 9840–9855.

Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R. K.-W.; and Lim, E.-P. 2023b. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2609–2634.

Wang, X.; Li, S.; and Ji, H. 2023. Code4Struct: Code Generation for Few-Shot Event Structure Prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3640–3663.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wyrzykowska, E.; Mikolajczyk, A.; Lynch, I.; Jeliazkova, N.; Kochev, N.; Sarimveis, H.; Doganis, P.; Karatzas, P.; Afantitis, A.; Melagraki, G.; et al. 2022. Representing and describing nanomaterials in predictive nanoinformatics. *Nature Nanotechnology*, 17(9): 924–932.

Xia, Y.; Kim, J.; Chen, Y.; Ye, H.; Kundu, S.; Talati, N.; et al. 2024. Understanding the Performance and Estimating the Cost of LLM Fine-Tuning. *arXiv preprint arXiv:2408.04693*.

Yang, K.; Liu, J.; Wu, J.; Yang, C.; Fung, Y. R.; Li, S.; Huang, Z.; Cao, X.; Wang, X.; Wang, Y.; et al. 2024. If llm is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *arXiv preprint arXiv:2401.00812*.

Zhao, A.; Huang, D.; Xu, Q.; Lin, M.; Liu, Y.-J.; and Huang, G. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19632–19642.

Zhuang, S.; Zhuang, H.; Koopman, B.; and Zuccon, G. 2024. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 38–47.