# SParK-Eval: Evaluating Structure-Aware Knowledge Acquisition in LLMs for Domain Adaptation to Industrial Records

**Ekant Muljibhai Amin, Yuta Koreeda, Yasuhiro Sogawa**
Research and Development Group, Hitachi, Ltd., Tokyo, Japan
{ekant.amin.mu, yuta.koreeda.pb, yasuhiro.sogawa.tp}@hitachi.com

## Abstract

Large Language Models (LLMs) often underperform in domain adaptation for industrial settings, where available corpora are limited and structurally diverse. These corpora frequently include non-natural formats such as tables, entity lists, or bullet-point instructions that hinder effective learning. To understand and improve domain-adaptive pretraining on such data, we introduce SParK-Eval (Structure-aware Parametric Knowledge Evaluation), a framework that constructs question–answer pairs from pretraining data and annotates each with its input structure (e.g., natural sentence, table, list). This enables fine-grained analysis of how input structure affects parametric knowledge acquisition during DAPT. Additionally, we propose a prompt-based input normalization method that converts diverse inputs into coherent natural sentences, providing a reference for isolating structural effects. Our experiments show that LLMs acquire substantially more knowledge from natural sentences than from their structurally non-standard counterparts. These findings underscore the importance of structure-aware evaluation in diagnosing learning challenges and guiding effective domain adaptation strategies.

## 1 Introduction

Large language models (LLMs) acquire general knowledge through pretraining on vast corpora. This pretraining typically relies on widely available general-domain sources—such as books, Wikipedia, news articles, and web data—which provide large volumes of high-quality natural language content (Gunasekar et al., 2023; Zhao et al., 2024; Longpre et al., 2024). While this enables broad linguistic and commonsense competence, LLMs often struggle when applied to specialized domains that feature unique terminology, structural patterns, and discourse styles not represented in the pretraining data.

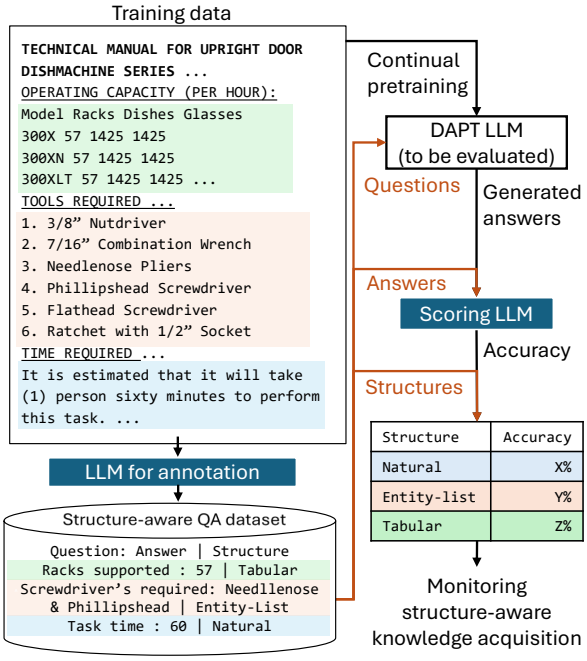


Figure 1: Overview of SParK-Eval, a framework for measuring structure-aware knowledge acquisition for evaluating LLM pretraining on limited, information-dense corpora with diverse structures. *Example shown uses publicly available dishwasher specifications*[1]*; actual experiments use proprietary technical records.*

To address this, Domain-Adaptive Pretraining (DAPT) is widely adopted to specialize LLMs for specific domains (Gururangan et al., 2020). However, DAPT corpora differ from general-domain data not only in domain specificity but also in *size* and *structure*. Broad domains such as law, biomedicine, finance, and mathematics typically provide large-scale resources—often billions of tokens—where redundancy ensures that knowledge is repeated and phrased in many natural-language variations. In contrast, *industrial and enterprise domains*—covering areas such as manufacturing,

[1] https://www.jacksonwws.com/wp-content/uploads/2016/11/7610-002-64-22E_300x-xn-xlt-xs.pdf



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 24403–24418
July 2-7, 2026 

energy, transportation, and IT operations—often offer much smaller corpora (only a few million tokens or less), because available text is largely restricted to highly specialized *technical records* (e.g., product manuals, software release notes, maintenance logs). These corpora are information-dense and frequently exhibit *diverse structures* such as bullet-point instructions, entity lists (e.g., supported product names), tables, or short imperative fragments.

Common data filtering practices, such as removing short or unpunctuated sentences (Raffel et al., 2020; Rae et al., 2022), may help in general corpora but can be detrimental in niche settings, where every sentence may carry important information. Hence, learning from structurally diverse data becomes crucial for effective DAPT.

Moreover, while recent works have explored data transformation techniques—such as rephrasing structured inputs into natural language (Cheng et al., 2024; Maini et al., 2024) or generating synthetic explanations from knowledge graph triplets (Yang et al., 2025)—these approaches primarily measure downstream task performance or overall knowledge gain. They offer limited insight into whether LLMs effectively encode knowledge from structurally diverse inputs such as tables or entity lists.

In industrial technical records, where data is limited yet highly information-dense, identifying which input structures hinder LLM learning is crucial for monitoring and diagnosing the model's knowledge acquisition during domain adaptation.

To address this gap, we make two complementary contributions:

- **Structure-aware Parametric Knowledge Evaluation (SParK-Eval):** We introduce a method that jointly generates evaluation questions from training data and annotates them with structure labels (e.g., natural sentence, entity list, tabular). This enables fine-grained analysis of how different data structures contribute to the parametric knowledge acquired during DAPT.

- **Isolating structure effects via input normalization within SParK-Eval:** To isolate the impact of data structure, we develop a prompt-based transformation method that converts all structurally diverse inputs into coherent, natural sentences. This provides a reference against which performance from original structures can be compared. We observe that:
  - LLMs achieve the highest knowledge gain from naturally structured sentences—consistent with findings from "Textbooks are all you need" (Gunasekar et al., 2023).
  - Structurally non-standard inputs—such as entity lists and tabular data—lead to significantly lower knowledge acquisition compared to their naturalized counterparts.

Building on these observations, by identifying hard-to-learn structures (e.g., tables or entity lists), SParK-Eval guides targeted normalization or rephrasing and also enables systematic evaluation of these data transformation techniques. In our industrial dataset, normalization yields substantial improvements for entity-list structures but only modest gains for tabular data, underscoring the need for more specialized data transformation approaches, such as table comprehension methods, for handling tabular content. Thus, it serves not only as an evaluation framework but also as a practical tool for optimizing domain-adaptive pretraining pipelines in industrial applications.

## 2 Related Work

### 2.1 DAPT Evaluation Techniques

Evaluating domain-adapted LLMs typically relies on downstream tasks such as question answering, summarization, conversational benchmarks, or professional certifications (Kumar et al., 2026; Singhal et al., 2023; Wu et al., 2023; Liu et al., 2024), which capture overall domain competence but reveal little about what knowledge the model actually acquires.

To isolate knowledge gain, fact-based probing methods have been proposed. LAMA (Petroni et al., 2019) reformulates knowledge triplets (subject, relation and object) into cloze-style queries and obtains a ranking of the ground truth token among vocabulary candidates in order to evaluate masked LMs. Recent studies including AdaptLLM (Cheng et al., 2024; Kim et al., 2025) apply a similar approach to causal LMs by measuring next-token likelihood on "predict-the-next-token" queries. EntiGraph (Yang et al., 2025) instead employs few-shot multiple-choice questions (MCQs) and additionally derives queries from training-aligned data to analyze training efficiency, though MCQ formulations have been shown to introduce positional and token bias (Wang et al.,

2024; Pezeshkpour and Hruschka, 2024; Zheng et al., 2024; Zhao et al., 2021).

Our approach similarly constructs probing queries from training data but extends prior work by explicitly linking them to diverse structural patterns for fine-grained analysis. Furthermore, we replace bias-prone MCQs with simpler question–answering queries, ensuring an unbiased and interpretable evaluation pipeline.

### 2.2 Training Data Characteristics

Beyond evaluation techniques, the characteristics of training data themselves critically shape LLM performance during domain adaptation. Prior work shows that book-like or Wikipedia-style text improves knowledge acquisition (Gunasekar et al., 2023), whereas low-context or noisy text—e.g., short, repetitive, or punctuation-free sentences—often degrades model quality (Zhao et al., 2024; Longpre et al., 2024).

To reduce noise in broad-domain corpora, preprocessing pipelines apply aggressive filtering; for example, the C4 dataset (used by T5) removes short or unpunctuated lines (Raffel et al., 2020). While effective for large redundant datasets, such filtering risks discarding structurally diverse but informative text in specialized industrial domains, where every sentence may encode critical knowledge.

To address this, recent work explores data conversion techniques that transform non-standard inputs into natural language. RephraseWeb (Maini et al., 2024) rephrases raw text into Wikipedia-like sentences using prompt-based LMs; AdaptLLM (Cheng et al., 2024) generates reading-comprehension tasks from raw text via regex transformations; and EntiGraph (Yang et al., 2025) constructs knowledge graphs and prompts LMs to produce edge-level explanations. While RephraseWeb and AdaptLLM target large general-purpose datasets, EntiGraph focuses on small-scale domain adaptation but operates primarily on natural-sentence corpora, without addressing structurally diverse formats like bullet points, tables, or entity lists.

A complementary effort, QuRating (Wettig et al., 2024), assesses data quality across dimensions such as writing style, factuality, and educational value using LLM-based judgments (GPT-3.5). However, this approach remains training-independent and relies on the subjective judgments of LLM evaluators rather than explicit alignment with model learning dynamics.

In summary, while prior work studies data filtering, data conversion, and data quality assessment, no existing method systematically analyzes how structural diversity in training data influences knowledge acquisition during DAPT. Accordingly, these methods are not directly comparable to SParK-Eval, as they address different tasks or settings; rather, they can be incorporated as alternative methods within different components of our framework.

## 3 Method

We propose SParK-Eval to analyze how LLMs encode knowledge from diverse structures in training data. This setting is particularly important for industrial technical records, where data is information-dense, highly specialized, and available only at a relatively small scale (often a few million tokens). Our method consists of two components: SParK-Eval for structure-aware evaluation, and an input normalization procedure to isolate structural effects.

### 3.1 SParK-Eval

Figure 2 provides an overview of SParK-Eval. We generate evaluation queries directly from the training corpus rather than relying on public datasets (Petroni et al., 2019; Cheng et al., 2024), enabling direct monitoring of knowledge acquisition from specific portions and structures of the training data while preserving links to the original source documents.

**Structure definitions** We categorize source snippets into three mutually exclusive structure types commonly observed in industrial records: *natural sentences* (standard prose conveying facts), *entity lists* (linear enumerations such as product names, versions, or supported platforms that are not tables), and *tabular data* (facts expressed in table form). Short or imperative fragments typically occur within these types; we therefore restrict labeling to these three categories to enable clear, non-overlapping analysis.

**Evaluation query design** To evaluate knowledge recall rather than reasoning ability, we initially considered multiple-choice questions (MCQs) because they simplify scoring, but prior work (Wang et al., 2024; Pezeshkpour and Hruschka, 2024; Zheng et al., 2024; Zhao et al., 2021) shows that MCQs introduce positional and token biases and *conflate*

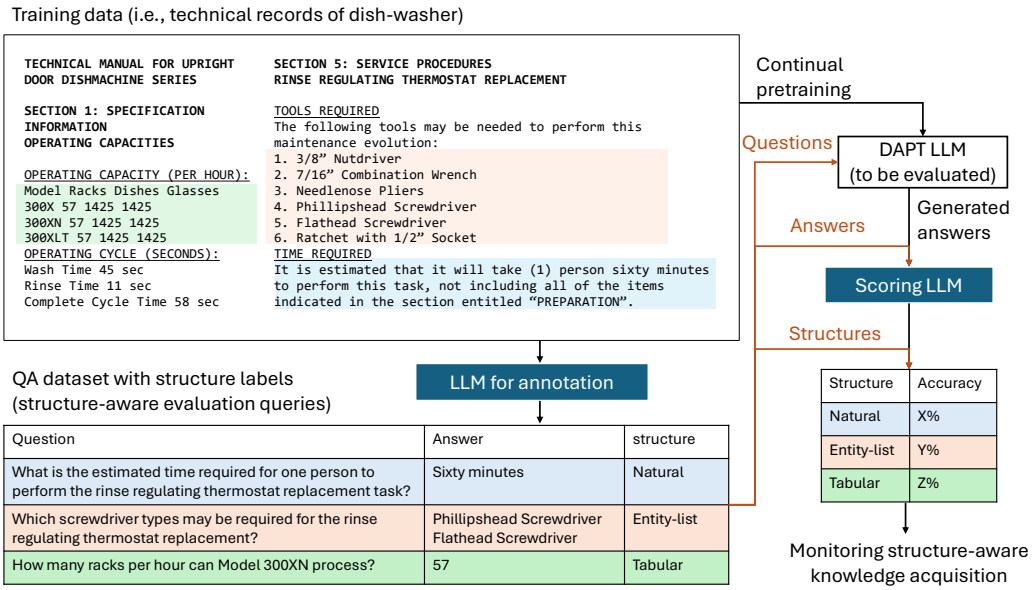


Figure 2: SParK-Eval for evaluating structure-aware knowledge acquisition from LLM pretraining. *Example shown uses publicly available dishwasher specifications (Footnote 1); actual experiments use proprietary technical records.*

knowledge assessment with additional reasoning requirements. We therefore use open-ended evaluation queries whose answers are constrained to short factual phrases or newline-delimited lists of entities, enabling reliable LLM-based scoring while avoiding MCQ-related biases.

**LLM-based query generation** We use a general-purpose instruction-following LLM (GPT-4.1-2025-04-14) with a carefully designed prompt that enforces three constraints: (i) each query must derive from exactly one structure type to ensure clean structural categorization, (ii) ambiguous or mixed cases are discarded to maintain annotation reliability, and (iii) questions must be self-contained so that answers depend only on the provided source passage rather than external context. Apart from a minimal requirement to include any additional context needed for disambiguation, all instructions remain domain-agnostic; full details appear in Appendix A.1. We typically provide the entire document as input; for longer files, we manually segment at semantically coherent boundaries (e.g., section or subdocument level) to preserve meaning while fitting within context limits.

**Evaluation metrics** We first experimented with reference-guided pairwise comparison (Zheng et al., 2023), where a judge LLM compares two model answers against a reference. While this yields pairwise win-rates and head-to-head scores (Appendix A.2), it lacks global consistency when comparing more than two models and remains susceptible to scalability and positional biases (Wang et al., 2024). To overcome these limitations, we adopt pointwise reference-guided evaluation: the judge LLM examines a single model answer and the reference, returning a binary decision indicating whether the prediction semantically matches the reference while ignoring surplus text or formatting artifacts (Appendix A.3).

This evaluation provides a fine-grained measure of knowledge acquisition across different structures; the next step isolates the effect of structure itself by normalizing inputs for direct comparison.

### 3.2 Isolating Structure Effects via Input Normalization within SParK-Eval

While SParK-Eval measures knowledge acquisition across diverse structures, performance may also be influenced by factors beyond structure, such as the clarity or inherent difficulty of individual queries. To isolate structural effects, we construct a *reference baseline* by converting all inputs into a single, standardized narrative format. This enables direct comparison between the original and normalized data, ensuring that performance differences arise solely from structural variation rather than query complexity.

**Reference baseline and normalization prompt** Motivated by prior work showing that book-like (Gunasekar et al., 2023) and Wikipedia-

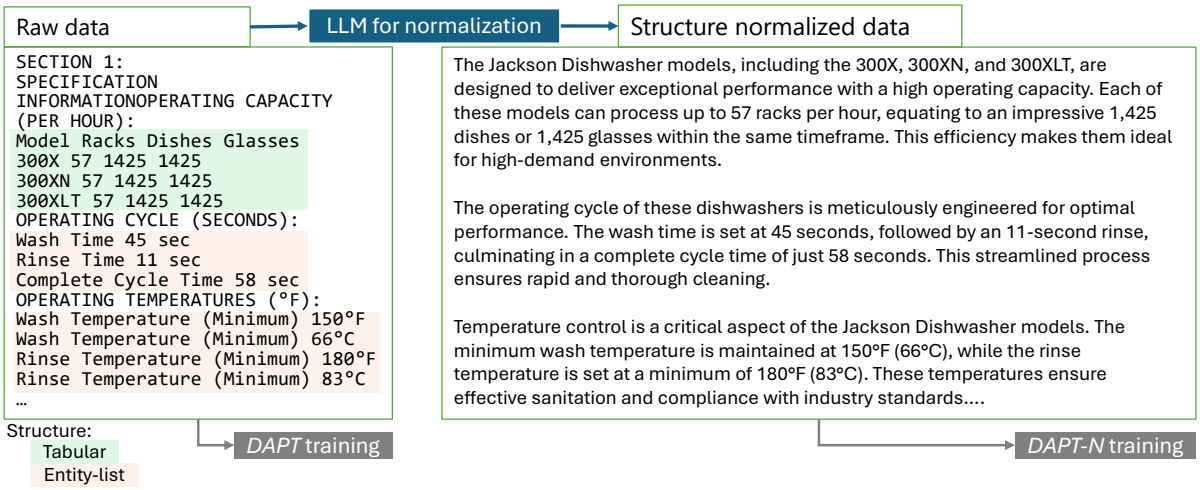


Figure 3: An illustration of structure normalization, where raw structured data is converted into high-context narrative sentences. *Example shown uses publicly available dishwasher specifications (Footnote 1); actual experiments were conducted on proprietary technical records.*

style (Maini et al., 2024) text provides high-quality pretraining signals, we approximate natural sentences by converting structurally diverse inputs into *high-context narrative sentences* (Figure 3). This normalized representation preserves all relevant information while presenting it in a coherent, descriptive style designed to maximize training utility. To achieve this, we design a detailed prompt for GPT-4.1 that explicitly accounts for structural diversity and the information-dense nature of industrial data. Unlike RephraseWeb (Maini et al., 2024), which paraphrases text using simple, short instructions for large-scale general-purpose corpora, our prompt applies a chain-of-thought strategy where the LLM (1) rates the quality of each segment, (2) comments on its training suitability, and (3) rewrites it into high-context narrative sentences for structure-controlled evaluation (Appendix A.4).

**Segmentation details** Because input length was found to substantially affect normalization quality (Maini et al., 2024), we empirically set the maximum segment length to 700 tokens: longer segments often introduced information loss through summarization, whereas much shorter ones disrupted document-level coherence. Each segment, together with the instructions, is processed by the LLM, and the converted text is extracted automatically using regular expressions, achieving a 99.4% extraction success rate; the negligible remainder is discarded.

## 4 Experiments

### 4.1 Data and Evaluation

We conduct experiments on proprietary industrial technical records from the release notes of a large enterprise IT management software suite. The dataset covers topics such as installation, configuration, administration, integration, and operational guidelines.

The dataset exhibits four key properties typical of industrial data: it is *unseen* for public LLMs, *limited* in size, *information-dense*, and *structurally diverse*. As this corpus is proprietary, it is not included in the pretraining data of public general-purpose LLMs, making it a suitable benchmark for industrial domain adaptation. The dataset size (42 documents, 4M tokens) reflects typical constraints of industrial adaptation settings, where available text is limited but highly information-dense. Unlike open domain corpora where the same fact may appear in multiple forms across documents, industrial records often provide a single, authoritative source for each fact, with minimal redundancy and no external repetition. Beyond natural sentences, it contains extensive *entity lists* (e.g., product names, supported operating systems) and *tabular data* capturing version details, feature availability, and platform support.

Using SParK-Eval, we generate *structure-aware evaluation queries*, each comprising a question, an answer, and a structure label. These queries are derived exclusively from the training set, using a subset of ten documents selected to maximize di-

| Structure Type | #Queries | Avg. Q Len | Avg. A Len | Avg. Entities | Min–Max Entities | Entity Word Len |
|---|---|---|---|---|---|---|
| Natural Sentences | 65 | 18.09 | 2.57 | – | – | – |
| Entity Lists | 85 | 16.27 | 16.72 | 5.44 | 1–33 | 3.12 |
| Tabular Data | 41 | 18.41 | 3.07 | – | – | – |

Table 1: Statistics of structure-aware evaluation queries. Q Len = average question length (words); A Len = average answer length (words); Entities = number of entities in entity-list answers; Entity Word Len = average words per entity.

versity across products, hardware platforms, and operating systems. A total of 191 queries are constructed across three structural types: 65 from natural sentences, 85 from entity lists, and 41 from tabular data. Table 1 summarizes query-level statistics, including question lengths, entity counts, and answer lengths. On average, the question component of each query contains 17.35 words, with only minor variation across structures. In contrast, answer lengths differ substantially: 2–3 words for natural and tabular structures and 16.7 words for entity lists. This difference arises because entity list queries include multiple entities in their answers, averaging 5.4 entities per answer (ranging from 1 to 33).

During evaluation, model outputs are generated in a zero-shot setting using deterministic decoding (do_sample=False) to ensure reproducibility. Since the experimental dataset is proprietary, we used publicly available data with similar characteristics for the demonstrations in Figures 1, 2, and 3.

### 4.2 Models and Continual Pretraining

**Base** We used Meta-Llama-3.1-8B (Grattafiori et al., 2024) as our base model, selected for its strong performance-to-size ratio and efficiency in domain adaptation tasks.

**DAPT** To obtain a domain-adapted LLM, we continually pretrained Meta-Llama-3.1-8B on our proprietary industrial technical records. The training–validation split was 90–10, and training ran for 11 epochs with a fixed random seed (42). We used the AdamW optimizer (Loshchilov and Hutter, 2019) with an initial learning rate of 5e-6, a linear scheduler, 20-step warmup, and weight decay of 0.1. Training was performed on eight NVIDIA H100 (80 GB) GPUs using DeepSpeed Stage 1 (Rajbhandari et al., 2020) with bfloat16 precision, requiring a total of ≈2.0 GPU-hours. We set gradient accumulation steps to 8 and the per-device batch size to 1, yielding an effective global batch size of 64. The maximum sequence length was 4,096 tokens.

**DAPT-N** Input normalization was performed using GPT-4o-2024-08-06 with the prompt described in Appendix A.4. Unlike augmentation approaches, normalization is applied once to isolate structural effects, reducing the dataset size from 4 M to 1.5 M tokens due to the removal of redundant phrases, particularly in entity lists. Each input segment was restricted to 700 tokens, following the empirical analysis in Section 3.2, which balances semantic coherence and processing efficiency. Text segmentation employed the `semchunk`[2] library with chunk_size as 700 to split documents into semantically meaningful segments before normalization. Training used the same hyperparameters as DAPT and was configured for up to 20 epochs, requiring ≈3.0 GPU-hours in total. Validation loss and gradient norms were examined after training and were found to stabilize around epoch 11, indicating convergence. We therefore selected epoch 11 as the stopping point.

### 4.3 SParK-Eval Effects

Table 2 summarizes structure-aware evaluation results for the base model, *DAPT*, and *DAPT-N*. We report findings for overall performance, structure-specific differences, and the impact of normalization. While the experiments use an industrial dataset, SParK-Eval is designed as a *general-purpose evaluation framework* for analyzing how input structure influences knowledge acquisition during domain-adaptive pretraining.

**Structure-agnostic evaluation** The base model (Llama-3.1-8B) achieves only 2.62% accuracy overall, confirming that general-domain LLMs retain little or no parametric knowledge from this proprietary industrial domain. Domain-adaptive pretraining on raw data (*DAPT*) improves accuracy to 16.23%, showing that even a small, structurally diverse corpus of technical records enables meaningful knowledge acquisition. Note that we apply

[2] https://github.com/isaacus-dev/semchunk

| Model | All | Natural | Entity-List | Tabular |
|---|---|---|---|---|
| Base | 2.62% | 6.15% | 0.00% | 2.44% |
| DAPT | 16.23% | 26.15% | 12.94% | 7.32% |
| DAPT-N | 24.61% | 26.15% | 28.24% | 14.63% |

Table 2: Accuracy (%) of base model, DAPT, and DAPT-N across all and structure-specific evaluation queries.

| Model | MMLU | HellaSwag | LAMBADA |
|---|---|---|---|
| Base | 63.35 ± 0.38 | 60.06 ± 0.49 | 75.51 ± 0.60 |
| DAPT | 62.33 ± 0.38 | 60.45 ± 0.49 | 73.67 ± 0.61 |
| Δ | -1.02 | +0.39 | -1.84 |
| DAPT-N | 62.41 ± 0.38 | 59.95 ± 0.49 | 73.26 ± 0.62 |
| Δ | -0.94 | -0.11 | -2.25 |

Table 3: Performance (%) of the base model, DAPT, and DAPT-N on standard LLM benchmarks with mean ± standard error. Rows marked Δ show differences vs. Base.

a strict scoring criterion: partially correct answers are counted as incorrect. Thus, reported scores reflect fully correct knowledge retrieval.

While this confirms that DAPT can acquire knowledge from the industrial domain, structure-agnostic evaluation masks structural variation in performance. To enable fine-grained analysis, we next turn to structure-specific results via SParK-Eval.

**Structure-specific evaluation** SParK-Eval enables disaggregation of performance by input structure. Under *DAPT*, natural-sentence queries achieve the highest accuracy (26.15%), while entity-list (12.94%) and tabular (7.32%) queries trail behind. This indicates that knowledge gains during DAPT arise primarily from naturally structured inputs, while non-natural formats pose greater learning challenges.

These differences highlight the importance of analyzing structure-specific effects, especially in domains with highly heterogeneous formats. However, since query difficulty and other confounders can also affect performance, we next examine results under *DAPT-N*, where all inputs are normalized into high-context narrative sentences.

**Impact of normalization (DAPT-N)** DAPT-N serves both as a conceptual reference for isolating structural effects and as a practical data transformation strategy. Based on Table 2, we observe:

1. *DAPT-N* improves overall performance, confirming that narrative-style inputs facilitate more effective pretraining.

2. The entity-list format benefits most from normalization (12.94% → 28.24%), indicating that raw entity-lists are particularly difficult for the model to learn from in their original structure.

3. Tabular data, while scoring lowest under DAPT (7.32%), show only modest improvement under DAPT-N (14.63%). This suggests that the degradation may stem not only from structural format but also from structure-agnostic factors such as the dense technical content of tabular inputs. A detailed analysis of these factors is presented in Section §5.2.

4. Natural-sentence performance remains unchanged between DAPT and DAPT-N, confirming that normalization preserves the original information without loss.

**Practical implications** SParK-Eval provides fine-grained, structure-aware evaluation of LLM knowledge acquisition, revealing which input formats hinder learning. Such structure-level diagnostics are especially valuable in industrial settings, where data are limited, heterogeneous, and information-dense. By identifying hard-to-learn structures (e.g., tables or entity lists), SParK-Eval guides structure-dependent preprocessing decisions such as targeted normalization or re-phrasing and also enables systematic evaluation of these data-transformation techniques. Thus, it serves not only as an evaluation framework but also as a practical tool for optimizing domain-adaptive pretraining pipelines in industrial applications.

## 5 Analysis and Discussion

### 5.1 General-domain Retention Without Replay

Table 3 reports performance on general benchmarks with mean ± standard error along with changes relative to the base model. Continual pretraining on industrial data yields small drops on MMLU (≤1.0 points) (Hendrycks et al., 2021) and LAMBADA (≤2.3 points) (Paperno et al., 2016), with negligible changes on HellaSwag (≤0.4 points) (Zellers et al., 2019). These modest differences indicate limited catastrophic forgetting, so replay would likely provide minimal additional benefit. We omit general-domain replay data intentionally to focus on domain knowledge

acquisition rather than optimizing general-purpose performance or prompting behavior.

### 5.2 Structure Normalization Efficacy and Tabular Performance

Figure 3 illustrates examples of structure normalization. Prompt-based LLMs effectively preserve semantic content from raw data and convert it into coherent narrative text. Both tabular data and entity lists are successfully transformed into natural-language form. Entity lists often contain repetitive phrases with minor variations; normalization merges these redundancies and adds contextual coherence (e.g., *Wash Temperature (Minimum) 150°F Wash Temperature (Minimum) 66°C* $\rightarrow$ *The minimum wash temperature is maintained at 150°F (66°C)*). Such redundancy frequently occurs in our industrial dataset, particularly across product names and OS variants, making entity lists well suited for normalization and contributing to the strong gains observed under DAPT-N.

**Understanding low tabular gain** Tabular inputs are also normalized into narrative sentences, reducing structural redundancy. To understand their relatively low performance, we manually examined 12 tabular queries (half correctly answered and half incorrect) to verify normalization quality. The normalization LLM effectively translated table contents into coherent sentences, occasionally incorporating relevant cross-references beyond the table. We did not account for segmentation-induced input errors, as such artifacts occur across all structural types. Nevertheless, tabular data tend to encode denser, more granular technical details—such as memory sizes, directory paths, or port specifications—than natural or entity-list inputs, which primarily contain lexical or categorical information (e.g., product names or component lists). We interpret this as a structure-agnostic factor contributing to lower tabular accuracy. This aligns with SParK-Eval's structure isolation results, where DAPT-N yields smaller gains for tabular inputs than for entity lists, suggesting that non-structural complexity plays a more dominant role in tabular performance.

**Numerical query analysis** Technical-domain data often include numerical answers, which may influence model performance. Using regular expressions and rule-based matching, we identified 39/191 numerical queries (20.4% of all questions): 19/65 from natural-sentence inputs, 2/85 from entity-list inputs, and 18/41 from tabular inputs (Table 5 in Appendix B). Given their higher proportion in tabular data, we initially suspected LLM weaknesses in numeral modeling (Spithourakis and Riedel, 2018; Mahendra et al., 2025) as a cause of low tabular accuracy. However, we observed no major discrepancy between numerical and non-numerical performance within the tabular subset. Interestingly, normalization significantly improved overall numerical accuracy from 7.7% (DAPT) to 25.6% (DAPT-N), while non-numerical accuracy rose from 18.4% (DAPT) to 24.3% (DAPT-N). Although this trend suggests that narrative-style normalization may help encode numerical information more effectively, the limited number of numerical queries calls for cautious interpretation and further validation in future work.

In summary, our study is the first to propose a framework for analyzing structural effects through SParK-Eval and structure normalization. Building on this foundation, our work opens avenues for stronger structure annotation and isolation techniques within the SParK-Eval framework, including ablations with synthetically controlled structure and content to further disentangle structural effects.

## 6 Conclusion

We presented SParK-Eval, a structure-aware evaluation framework for analyzing how LLMs acquire knowledge from diverse input formats during domain-adaptive pretraining. By generating structure-labeled evaluation queries directly from the training corpus, SParK-Eval enables fine-grained analysis of parametric knowledge retention across natural sentences, entity lists, and tabular data.

To isolate the effects of structure, we introduced a prompt-based normalization method that transforms structured inputs into coherent narrative text. Our experiments on industrial technical records show that natural sentences contribute most to knowledge acquisition, while entity lists and tabular data yield significantly lower gains unless normalized.

These findings highlight the importance of input structure in domain adaptation and demonstrate that normalization can serve as a practical intervention for improving learning from structurally diverse corpora. SParK-Eval offers a general and extensible framework for structure-controlled evaluation, with future work aimed at expanding structural definitions, improving normalization quality,

and applying the method to broader domains and model families.

## Limitations

While SParK-Eval provides a structured framework for evaluating knowledge acquisition across diverse input structures, several assumptions and constraints may limit its generalizability.

SParK-Eval assumes that structural categories (e.g., tables, lists, and natural sentences) are disjoint and that each fact appears in a single structure type. This assumption generally holds in technical documentation such as industrial release notes, where redundancy is minimal. However, in more open-domain or web-scale corpora, the same information may appear in multiple structural formats, complicating efforts to isolate structure-specific learning effects.

Although the framework is conceptually domain-agnostic, applying it to new domains may require light tuning, such as redefining structure types or adjusting prompt templates to match domain-specific patterns. The released prompts (Appendix A.1 and A.4) serve as adaptable starting points, and in practice, a brief inspection of domain data is typically sufficient to apply the method effectively. As such, generalization requires minimal effort rather than major redesign.

The normalization process depends on LLM prompting and input segmentation, both of which can influence output consistency. In particular, variations in input length may lead to outputs that diverge from expectations. However, with the rapid advancement of LLMs in handling longer and more complex contexts, we expect the robustness and reliability of normalization to improve accordingly.

## Acknowledgments

We thank Kana Ozaki for carefully reviewing an earlier draft and providing valuable comments that helped sharpen the analysis. The first author wishes to dedicate this work to the memory of Professor Pushpak Bhattacharyya, whose passion for NLP research has been a guiding inspiration throughout this work.

## References

Daixuan Cheng, Shaohan Huang, and Furu Wei. 2024. Adapting large language models via reading comprehension. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need. *Preprint*, arXiv:2306.11644.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Jiyeon Kim, Hyunji Lee, Hyowon Cho, Joel Jang, Hyeonbin Hwang, Seungpil Won, Youbin Ahn, Dohaeng Lee, and Minjoon Seo. 2025. Knowledge entropy decay during language model pretraining hinders new knowledge acquisition. In *The Thirteenth International Conference on Learning Representations*.

Aman Kumar, Ekant Muljibhai Amin, Xian Yeow Lee, Lasitha Vidyaratne, Ahmed K. Farahat, Dipanjan D. Ghosh, Yuta Koreeda, and Chetan Gupta. 2026. Building domain-specific small language models via guided data generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(47):40287–40294.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, Bonita Bhaskaran, Bryan Catanzaro, Arjun Chaudhuri, Sharon Clay, Bill Dally, Laura Dang, Parikshit Deshpande, Siddhanth Dhodhi, Sameer Halepete, and 22 others. 2024. ChipNeMo: Domain-adapted LLMs for chip design. *Preprint*, arXiv:2311.00176.

Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, and Daphne Ippolito. 2024. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity. In *Proceedings of the*

*2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3245–3276, Mexico City, Mexico. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Rahmad Mahendra, Damiano Spina, Lawrence Cavedon, and Karin Verspoor. 2025. Evaluating numeracy of language models as a natural language inference task. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 8336–8361, Albuquerque, New Mexico. Association for Computational Linguistics.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14044–14072, Bangkok, Thailand. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, and 61 others. 2022. Scaling language models: Methods, analysis & insights from training Gopher. *Preprint*, arXiv:2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRo: Memory optimizations toward training trillion parameter models. *Preprint*, arXiv:1910.02054.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, and 12 others. 2023. Towards expert-level medical question answering with large language models. *Preprint*, arXiv:2305.09617.

Georgios Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2104–2115, Melbourne, Australia. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. QuRating: Selecting high-quality data for training language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A large language model for finance. *Preprint*, arXiv:2303.17564.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candes, and Tatsunori Hashimoto. 2025. Synthetic continued pretraining. In *The Thirteenth International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Yang Zhao, Li Du, Xiao Ding, Kai Xiong, Zhouhao Sun, Shi Jun, Ting Liu, and Bing Qin. 2024. Deciphering the impact of pretraining data on large language models through machine unlearning. In *Findings of the Association for Computational Linguistics: ACL*

*2024*, pages 9386–9406, Bangkok, Thailand. Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

# A SParK-Eval Method

## A.1 Prompt for Annotation

The prompt designed for structure-aware annotation, shown in Figure 4, specifies the task objective, constraints, chain-of-thought steps, output format guidelines, and few-shot examples. The example few-shots originally derived from proprietary data are replaced in Figure 4 with few-shots constructed from publicly available Jackson dishwasher specifications. Interestingly, these few-shots are generated directly from the original prompt without any modifications, demonstrating that the LLM-based annotation approach generalizes across datasets. *Note: Curly-braced expressions (e.g., enterprise-job-scheduler) denote anonymized proprietary terms.*

```
**I want you to create Question Answers from the document. I will provide the instructions and
    constraints, and then later on I will provide the document.**

---

## Instructions and Constraints for Creating Evaluation QA Pairs from Product Documentation

### **Purpose**

The purpose of this task is to generate evaluation data for Large Language Model (LLM) training
    and assessment. The aim is to test whether the LLM has acquired specific domain knowledge, *
    not* to evaluate its reasoning or inference abilities. Questions must be factual and
    directly answerable by someone who has read the source document.

---

### **Constraints**

#### 1. **Source Text Selection**

Questions should be created from **three mutually exclusive types** of source reference:

* **Natural Sentences:** Full, narrative/explanatory sentences from the document.
* **Entity-lists:** List structures, such as product lists, enumerations of software, hardware,
    supported platforms, abbreviations, or similar iterative lists that are not in table format.
     (Answer may be a multi-line list.)
* **Tabular Data:** Information directly presented in table format (not just listed). The answer
     should be simple, knowledge-based, and not require multi-step reasoning. Tabular and entity
    -list types must not overlap.

> **Discard any cases where you cannot confidently classify the source reference as one of these
     three types, or if the information is ambiguous, mixed, or otherwise complex. Be strict in
    ensuring that each QA pair is created from a source that clearly fits one and only one of
    these types.**

#### 2. **Question Construction**

* For **all types**, each question should check a specific, knowledge-based fact **explicitly**
    stated in the source.
* Questions must be specific enough that the answer is only obvious if the source reference has
    been read.
* For tabular and entity-list types, phrase questions so they can stand alone without requiring
    reference to a specific table or list number/name. The question should be understandable and
     answerable independently, even if the responder does not recall the specific source
    structure.
* If the document is specific to a particular OS (such as AIX), or if the answer may vary by OS/
    platform, explicitly include the OS (e.g., "on AIX") in the question along with the product
    name. This ensures clarity when the same product may support multiple platforms or
    environments.

#### 3. **Answer Format**

* For **natural sentences and tabular** sources, the answer must be a single word, short phrase,
     or (rarely) a simple sentence. Avoid explanations or multi-sentence answers.
```

```
* For **entity-list** sources, the answer should be a list with each entity on a new line.
    Entity lists of any length are acceptable.
* For **tabular** and **natural** types, avoid combining or mixing answers from other types.

#### 4. **Product and OS Context**

* If the question or answer could apply to more than one product or document, or if the
    information is OS-specific, specify the full product name and the OS (e.g., "{enterprise-job
    -scheduler} - Agent on AIX") in the question for clarity.

#### 5. **Source Reference**

* For each QA pair, include the original reference text (1-3 sentences, or the relevant entity-
    list/table snippet) from which the QA is derived. Quote or lightly trim for clarity.

#### 6. **Entity-list Extra Requirement**

* For QA pairs with an **entity-list** answer, also include the number of entities in the list
    as described in the output format.

#### 7. **Strict Exclusivity**

* Strictly discard any pairs where type classification is ambiguous or where information is
    mixed, complex, or not knowledge-based.

---

### **Step-by-Step Procedure**

1. **Carefully read** the provided document, identifying suitable source references that match **
    exactly one** of the three allowed types: natural sentence, entity-list, or tabular.

2. **Draft a specific, knowledge-based question** for each selected reference.
   * If the document or answer is OS-specific, always include the OS/platform (e.g., "on AIX") in
       the question text.

3. **Write a precise answer** that matches the information, in the required format for its type.

4. **If needed, add product and OS name** to the question for clarity.

5. **Attach the exact source reference text** (1-3 sentences or the relevant list/table snippet).


6. **Assign the correct type** for each QA pair, as described in the output format. For entity-
    list answers, also add the entity count.

7. **Finalize and Output:**
   * Ensure all pairs meet these constraints.
   * Create and present exactly ten QA pairs if possible (otherwise, as many as suitable).
   * Output the results as a JSON array in the format below.
   * Output must be only the JSON file so that the user can easily copy it. Do not include any
       additional conversational text in the response.

---

### **JSON Output Format**

Each JSON entry **must** include the following fields:
* "question": The constructed question.
* "answer": The answer in the appropriate format for its type.
* "source_reference": The original reference excerpt or relevant snippet.
* "source_reference_type": One of "natural", "entity-list", or "tabular".
* If the type is "entity-list", also include "entity_list_length" (number of entities, integer).

**Example:**
[
  {
```

```
    "question": "What is the estimated time required for one person to perform the rinse
        regulating thermostat replacement task?",
    "answer": "Sixty minutes",
    "source_reference": "TIME REQUIRED It is estimated that it will take (1) person sixty minutes
         to perform this task, not including all of the items indicated in the section entitled
        "PREPARATION".",
    "source_reference_type": "natural"
  },
  {
    "question": "Which screwdriver types may be required for the rinse regulating thermostat
        replacement?",
    "answer": "Phillipshead Screwdriver\nFlathead Screwdriver",
    "source_reference": "TOOLS REQUIRED The following tools may be needed to perform this
        maintenance evolution: 4. Phillipshead Screwdriver 5. Flathead Screwdriver",
    "source_reference_type": "entity-list",
    "entity_list_length": 2
  },
  {
    "question": "How many racks per hour can Model 300XN process?",
    "answer": "57",
    "source_reference": "OPERATING CAPACITY (PER HOUR): Model Racks Dishes Glasses 300XN 57 1425
        1425",
    "source_reference_type": "tabular"
  }
]

**Your output must be only the JSON file as shown above, with no extra text. It should not
     contain JSON md like response with ‘json‘. Only json list as a answer is accepted.**

---

### **Document to Process**
{{document}}
```

Figure 4: LLM prompt for SParK-Eval query annotation. *Note: The example few-shots originally derived from proprietary data are replaced with few-shots derived from public data (Footnote 1).*

### A.2 Win-rate as a Scoring Metric

Before adopting accuracy as our primary evaluation metric, we experimented with a win-rate approach inspired by the LLM-as-a-judge framework (Zheng et al., 2023). In this setup, a reference answer and two model-generated responses are presented to a judge LLM (GPT-4o-2024-08-06), which is tasked with selecting the better response.

We evaluated model outputs along two dimensions: (1) Domain relevance to the given domain (ignoring factual correctness), and (2) Correctness (ignoring domain specificity and other factors). This separation allows us to assess whether models leverage domain-specific knowledge independently of factual accuracy. Win-rate evaluation offers qualitative insights—for instance, the DAPT model generally outperforms the base model, though both struggle with tabular inputs. Ties are frequent when both models fail. However, this approach suffers from positional bias and lacks scalability for evaluations beyond pairwise comparisons.

Therefore, we adopt a simpler, reference-based evaluation metric that directly compares predicted answers with reference answers, reporting binary correctness. The full evaluation prompt is provided in a later section.

### A.3 Scoring LLM Prompt with Accuracy Metric

Figure 5 shows the prompt used to evaluate model responses using the accuracy-based metric. This scoring LLM prompt is designed to compare a model's output against a reference answer and determine binary correctness.

| Comparison | Question Type | Correctness (%) | | | Domain Relevance (%) | | |
|---|---|---|---|---|---|---|---|
| | | A Wins | B Wins | Tie | A Wins | B Wins | Tie |
| Base vs. DAPT (A vs B) | Overall | 10.99 | **49.21** | 39.79 | 19.37 | **65.97** | 14.66 |
| | Natural | 13.85 | **47.69** | 38.46 | 24.62 | **61.54** | 13.85 |
| | Entity-list | 9.41 | **61.18** | 29.41 | 16.47 | **68.24** | 15.29 |
| | Tabular | 9.76 | 26.83 | **63.41** | 17.07 | **68.29** | 14.63 |
| Base vs. DAPT-N (A vs B) | Overall | 13.09 | **54.45** | 32.46 | 23.56 | **67.02** | 9.42 |
| | Natural | 15.38 | **52.31** | 32.31 | 26.15 | **60.00** | 13.85 |
| | Entity-list | 12.94 | **64.71** | 22.35 | 16.47 | **75.29** | 8.24 |
| | Tabular | 9.76 | 36.59 | **53.66** | 34.15 | **60.98** | 4.88 |

Table 4: Pairwise win-rate comparison between models inspired by the LLM-as-a-judge framework (Zheng et al., 2023).

| | Base | | DAPT | | DAPT-N | |
|---|---|---|---|---|---|---|
| Type | Num. Acc. | Non-Num. Acc. | Num. Acc. | Non-Num. Acc. | Num. Acc. | Non-Num. Acc. |
| Natural | 0.0 (0/19) | 8.7 (4/46) | 10.5 (2/19) | 32.6 (15/46) | 31.6 (6/19) | 23.9 (11/46) |
| Entity-list | 0.0 (0/2) | 0.0 (0/83) | 0.0 (0/2) | 13.3 (11/83) | 0.0 (0/2) | 28.9 (24/83) |
| Tabular | 0.0 (0/18) | 4.4 (1/23) | 5.6 (1/18) | 8.7 (2/23) | 22.2 (4/18) | 8.7 (2/23) |
| Overall | 0.0 (0/39) | 3.3 (5/152) | 7.7 (3/39) | 18.4 (28/152) | 25.6 (10/39) | 24.3 (37/152) |

Table 5: Accuracy (%, with correct/total counts) on numerical vs. non-numerical queries across Base, DAPT, and DAPT-N. Numerical queries are generally harder and normalization (DAPT-N) appears to improve numerical accuracy.

```
[Instruction]
Please act as an impartial judge and evaluate the quality of the response provided by an AI
    assistant to the user question displayed below. Your evaluation should consider only
    correctness and helpfulness. You will be given a reference answer and the assistant's answer.
     Begin your evaluation by comparing the assistant's answer with the reference answer.
    Identify and correct any mistakes. Be as objective as possible. If the assistant's answer
    contains surplus information or irrelevant text after providing the answer, please ignore it
     in your evaluation. If assistant's answer is partially correct it is considered as
    incorrect. After providing your explanation, output your final verdict by strictly following
     this format:"[[True]]" if assistant is correct, "[[False]]" if assistant is incorrect.

[Question]
{question}

[The Start of Reference Answer]
{reference_answer}
[The End of Reference Answer]

[The Start of Assistant's Answer]
{model1_answer}
[The End of Assistant's Answer]
```

Figure 5: Prompt used by the scoring LLM for accuracy-based scoring.

### A.4 Input Normalization Prompt

The prompt used for structure normalization (Section 3.2) converts industrial data (release notes from an enterprise IT system) into high-context narrative sentences. *Note: Curly-braced expressions (e.g., enterprise-name and xx.xx) denote anonymized proprietary terms.*

```
Please analyze the data I provide, which consists of continual training data for domain
    adaptation, specifically from the {enterprise-name} release notes version {xx.xx}. As you
    may already know, low-quality data is typically removed from training datasets to enhance
    performance. Publicly available large models are generally trained on data containing
    natural sentences. For instance, book-like data is considered to be of the highest quality
    for training large language models (LLMs).

With this in mind, I would like you to rate the quality of the data mentioned below on a scale
    of 1 to 10. Additionally, please provide insights into its suitability for training LLMs. In
     the next step, convert the text into high-quality data. Please ensure that the meaning is
    preserved and that no information is lost during the conversion. The data is part of an
    entire document, which I will later combine with all the converted parts, so do not shuffle
    the sentences during conversion. Pay careful attention to page headers and footers, and
    consider linking important information such as product codes or product names from the
    release notes in the converted data for better reference. Ensure that all elements and
    information are included when lists are encountered; do not trim the lists. Provide a
    narrative structure and a natural sentence flow even for lists.

The converted text should be formatted as follows: ‘‘Conversion to High-Quality Data:<<<<<{{
    converted-data}}>>>>>’’

The data is as follows:
{data}
```

Figure 6: Prompt for normalizing diverse structures into high-context narrative sentences.

For the Jackson Dishwasher manual example in Figure 3, we replaced "*enterprise-name release notes version xx.xx*" with "*technical manual for Jackson Dishwasher models*", and added the instruction "*Task Purpose: Convert given data into high-context narrative sentences.*" at the beginning of the prompt.

## B Analysis

We provide a breakdown of performance on numerical vs. non-numerical queries in Table 5.