Product Attribute Value Extraction using Large Language Models

Alexander Brinkmann
alexander.brinkmann@uni-mannheim.de
University of Mannheim
Mannheim, Germany

Roee Shraga
rshraga@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

Christian Bizer
christian.bizer@uni-mannheim.de
University of Mannheim
Mannheim, Germany

ABSTRACT

E-commerce platforms rely on structured product descriptions, typically in the form of attribute/value pairs to enable features such as faceted product search and product comparison. However, vendors on these platforms often provide unstructured product descriptions consisting of only an offer title and a textual description. To process such offers, e-commerce platforms must extract attribute/value pairs from the titles and textual descriptions. State-of-the-art attribute/value extraction methods based on pre-trained language models (PLMs), such as BERT, face two drawbacks (i) the methods require significant amounts of task-specific training data and (ii) the fine-tuned models have problems to generalize to unseen attribute values that were not part of the training data. This paper explores the potential of using large language models (LLMs) as a more training data-efficient and more robust alternative to existing attribute/value extraction methods. We propose different prompt templates for instructing LLMs about the target schema of the extraction, covering both zero-shot and few-shot scenarios. In the zero-shot scenario, textual and JSON-based approaches for representing information about the target attributes are compared. In the scenario with training data, we investigate (i) the provision of example attribute values, (ii) the selection of in-context demonstrations, (iii) shuffled ensembling to prevent position bias, and (iv) fine-tuning the LLM. We evaluate the proposed prompt templates in combination with hosted LLMs, such as GPT-3.5 and GPT-4, and open-source LLMs based on Llama2 which can be run locally. We compare the performance of the LLMs to the PLM-based methods SU-OpenTag, AVEQA, and MAVEQA. The overall best average F1-score of 86% was reached by GPT-4 using an ensemble of shuffled prompts that combine knowledge about the target schema in the form of attribute names, attribute descriptions, example values, and demonstrations. Given the same amount of training data, this prompt/model combination outperforms the best PLM baseline by an average of 6% F1. Fine-tuning GPT-3.5 results in a comparable performance to GPT-4 but it harms the LLM's ability to generalize.

CCS CONCEPTS

• **Information systems → Data extraction and integration**.

KEYWORDS

Product attribute value extraction, large language models, e-commerce

1 INTRODUCTION

Online shoppers are used to filtering and comparing products along criteria such as brand, color, or screen-size to find products that fit their needs [19]. In contrast, vendors often upload product offers to e-commerce platforms that contain only textual descriptions such as offer titles or product descriptions [24, 38]. To enable faceted search and attribute-based product comparison for online shoppers, attribute/value pairs need to be extracted from these textual descriptions [26, 32, 37]. Figure 1 shows an example offer for a toothbrush set and a result of applying attribute/value extraction to its title. The colored boxes visualize the identified attributes while the attribute values are underlined.

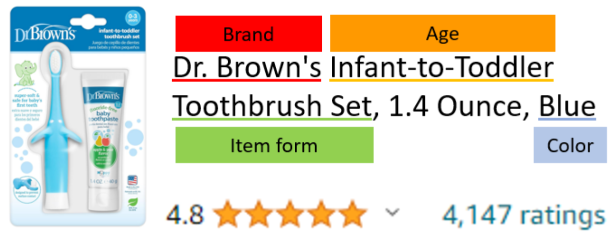

Figure 1: An example product title with tagged attribute/value pairs. Vendors include product attribute values in the title to enhance visibility.

State-of-the-art techniques for product attribute/value extraction rely on pre-trained language models (PLMs) [24, 29, 32, 38] such as BERT [4]. These approaches have two main drawbacks: (i) they require a large amount of task-specific training data for fine-tuning, and (ii) the fine-tuned models have problems to generalize to unseen attribute values that were not part of the training data. Large autoregressive language models (LLMs) such as GPT-3.5 [16], GPT-4 [15], or Llama2 [21] have shown their potential to overcome these shortcomings for various natural language processing tasks [1, 3, 23, 27].

This paper explores the potential of LLMs for extracting attribute values from product titles. We consider hosted LLMs, such as OpenAI's GPT-3.5 and GPT-4, and open-source LLMs, such as StableBeluga and SOLAR, which can be run locally. The performance of an LLM for a specific task strongly depends on the formulation of the prompt [11, 13]. Thus, we propose and evaluate a wide range of different prompt templates for instructing LLMs about the target schema of the extraction, covering both zero-shot and few-shot scenarios. In the zero-shot scenario without task-specific training data, we evaluate different approaches for representing attribute descriptions. In the few-shot scenario, we compare four different approaches to utilize the training data: (i) providing example attribute values, (ii) selecting in-context demonstrations, (iii) shuffled ensembling to avoid position bias, and (iv) fine-tuning the LLM. Afterwards, we compare the performance of the best prompt/model combinations to the performance of PLM-based baseline methods using different amounts of training data. In summary, we make the following contributions:

(1) To the best of our knowledge, this paper is the first scientific paper to investigate the potential of LLMs for product attribute value extraction.
(2) We propose different prompt templates for instructing LLMs about the target schema of the attribute value extraction task. The templates cover use cases with and without task-specific training data.
(3) Our experiments show that to reach decent performance, LLMs require a small set of task-specific training data for picking example values and/or demonstrations. Simply describing the target attributes without providing examples (zero-shot) results in much lower performance.
(4) The comparison of the different approaches to utilizing training data shows that providing demonstrations is more effective than providing example values. Out of the different demonstration selection methods, selecting semantically related demonstrations proved the most effective.
(5) Comparing the performance of the different LLMs, we show that GPT-4 outperforms all other models with a top average F1 score of 86%. GPT-4's top score is 10% higher than the F1 score of the best open-source LLM Beluga-2.
(6) Comparing LLMs and PLMs, we show that LLMs are more training data-efficient. Given the same amount of training data, GPT-4 reaches a 6% higher average F1 score than the best PLM-based baselines AVEQA and SU-OpenTag. GPT-4 is also more robust to unseen values. The model achieves a 19% higher average F1 score for unseen values than the most robust PLM-based method AVEQA.
(7) We are the first to experiment with fine-tuning LLMs for product attribute value extraction. Our experiments show that a fine-tuned GPT-3.5 model can achieve a similar performance as GPT-4 but that the fine-tuning damages the ability of the model to generalize to different target schemata.

The paper is structured as follows. First, we review related work. Next, we describe the experimental setup (Section 3) before delving into prompt engineering. We introduce zero-shot prompt engineering (Section 4) as well as in-context learning and fine-tuning (Section 5) to evaluate the usage of task-specific training data. Lastly, we compare LLM- and PLM-based methods in Section 6. The code and data for replicating all our experiments are available online[1].

2 RELATED WORK

Product Attribute Value Extraction. Early works on attribute value extraction used domain-specific rules to extract attribute/value pairs [22, 35] from product descriptions. The first learning-based methods required an extensive amount of feature engineering and did not generalize to unknown attributes and values [5, 18, 28]. Recent works have adopted BiLSTM-CRF architectures [9, 37] to tag attribute values in product titles, due to the advancement of neural networks. OpenTag trains a BiLSTM-CRF model with active learning [37]. SU-OpenTag [29] is an extension of OpenTag that encodes both a target attribute and the product title using the pre-trained language model, BERT [4]. AdaTag [30] employs BERT [4] and a mixture-of-experts module to extract attribute values. TXtract [7] integrates a product taxonomy into the extraction model. AVEQA [24] and MAVEQA [32] approach attribute value extraction as a question-answering task, using different pre-trained language models to encode the target attribute, product category, and product title. SU-OpenTag, AVEQA and MAVEQA serve as baselines for our experiments. OA-Mine [36] employs BERT [4] to mine for unknown attribute values and attributes. Recent studies have utilized soft prompt tuning to fine-tune a small number of trainable parameters in a language model [2, 31].

Information Extraction using LLMs. LLMs often show better zero-shot performance compared to PLMs and are more robust to unseen examples [3] because they are pre-trained on large amounts of text, and have emergent effects due to their model size [27]. LLMs have successfully been used for information extraction tasks in other application domains [1, 12, 20, 25]: Wang et al. [25] and Parekh et al. [17] employed OpenAI's LLMs to extract structured data about events from unstructured text. Goel et al. [6] combine LLMs with human expertise to annotate patient-related information in medical texts. Agrawal et al. [1] use InstructGPT with zero-shot and few-shot prompts to extract information from clinical notes. Shyr et al. [20] evaluate ChatGPT to extract rare disease phenotypes from unstructured text. LLMs have been used to re-rank information extracted by PLM-based models [12]. To the best of our knowledge, we are the first to investigate the potential of LLMs for product attribute value extraction.

3 EXPERIMENTAL SETUP

This section introduces the experimental setup, including the datasets, LLMs, and evaluation metrics.

3.1 Datasets

The OA-Mine [36] and the AE-110K [29] datasets are benchmark datasets for product attribute value extraction that have been used in related work [24, 31, 32]. Both datasets consist of English product offers with annotated attribute/value pairs.

OA-Mine. We use the human-annotated product offers of the OA-Mine dataset[2] [36]. The subset includes 10 product categories, with up to 200 product offers per category. Each category has between 8 to 15 attributes, resulting in a total of 115 unique attributes. Attributes with the same name but different product categories are treated as distinct attributes. No further pre-processing is applied to the subset of OA-Mine.

AE-110K. The AE-110K dataset[3] comprises triples of product titles, attributes and attribute values from the AliExpress Sports & Entertainment category [29]. Product offers are derived by grouping the triples by product title. The subset includes 10 product categories, with up to 400 product offers per category. For each category, 6 to 17 attributes are known, resulting in a total of 101 unique attributes.

Training/Test Split. The subsets OA-Mine and AE-110K are split into training and test sets in a 75:25 ratio, stratified by product category to ensure the presence of all attributes in both sets. To

[1] https://github.com/wbsg-uni-mannheim/ExtractGPT

[2] https://github.com/xinyangz/OAMine/tree/main/data

[3] https://raw.githubusercontent.com/lanmanok/ACL19_Scaling_Up_Open_Tagging/master/publish_data.txt

Table 1: Example product offers, attribute/value pairs and predicted attribute values for the two datasets OA-Mine and AE-110k.

	(a)	(b)	(c)	(d)
Dataset	OA-Mine	OA-Mine	AE-110k	AE-110k
Category	Vitamin	Coffee	Guitar	Shorts
Attribute	Net Content	Flavor	Body Material	Brand Name
Title	NOW Supplements, Vitamin A (Fish Liver Oil) 25,000 IU, Essential Nutrition, 250 Softgels	Cafe Du Monde Coffee Chicory, 15-Ounce (Pack of 3)	Factory customization Acoustic Guitar Sitika Solid Spruce Vintage Sunburst high-quality	PENERAN 2018 Sport Shorts Man Running Fitness Gym Short Mens Sportswear Jogging Athletic Short Dry Fit Training New Arrival 3XL
Target Value	250	Coffee Chicory	Solid Spruce	PENERAN
list	n/a ✗	n/a ✗	Sitika Solid Spruce ✗	PENERAN ✓
json-val	250 Softgels ✗	n/a ✗	Solid Spruce ✓	n/a ✗
json-val-dem	250 ✓	Coffee Chicory ✓	Solid Spruce ✓	PENERAN ✓

evaluate the impact of the amount of training data on LLM performance, we create small and large training sets for OA-Mine and AE-110K. The large training sets include all available training data. For the small training set, 20% of the product offers per category are sampled from the test set. Table 2 contains statistics for both datasets. The datasets are accessible on our GitHub repository.

Example Extractions. Table 1 shows example product offer titles, attributes and the target attribute values from the datasets. The target values are also underlined in the titles. The lower part of the table shows the attribute values that were extracted by GPT4 using the prompt templates `list`, `json-val` and `json-val-dem` which will be introduced in Section 4. Correctly extracted attribute values are marked with ✓ and incorrectly extracted attribute values are marked with ✗.

Table 2: Statistics for OA-Mine and AE-110K.

	OA-Mine			AE-110K		
	Small Train	Large Train	Test	Small Train	Large Train	Test
A/V Pairs	1,467	7,360	2,451	859	4,360	1,482
Unique A/Vs	1,120	4,177	1,749	302	978	454
Product Offers	286	1,452	491	311	1,568	524

3.2 Large Language Models

This paper evaluates both proprietary LLMs such as GPT-3.5 and GPT-4, hosted by OpenAI, and open-source LLMs based on Llama2 [21]. Table 3 lists all evaluated LLMs with the exact model name, number of parameters, and access via API or number of GPUs for running the LLM locally. We access GPT-3.5 and GPT-4 through the OpenAI API. The open-source models Beluga2[4], Beluga-7B[5], and Solar[6] are publicly available on the Huggingface hub and can be run on local GPUs.

[4]https://huggingface.co/stabilityai/StableBeluga2
[5]https://huggingface.co/stabilityai/StableBeluga-7B
[6]https://huggingface.co/upstage/SOLAR-0-70b-16bit

Table 3: List of evaluated LLMs.

LLM	Exact Name	Model Size	API/GPUs
GPT-3.5 [16]	gpt-3.5-turbo-0613	175B	API
GPT-4 [15]	gpt-4-0613	unknown	API
Beluga2	StableBeluga2	70B	4
Beluga-7B	StableBeluga-7B	7B	1
Solar	SOLAR-0-70b-16bit	70B	3

The temperature parameter of the LLMs is set to 0 to reduce the randomness. The OpenAI API and the local execution of open-source models are facilitated using the langchain Python package[7]. Computational experiments were performed on a shared server equipped with 96 × 3.6 GHz CPU cores, 1024 GB RAM, and 8 NVIDIA RTX A6000 GPUs.

3.3 Evaluation Metrics

In our experiments, we report F1-scores, which were calculated by categorising predictions into five categories, following previous works [24, 29, 30, 32]. The five categories are NN (no predicted value when the ground truth does not contain an attribute value), NV (incorrect predicted value when the ground truth does not contain an attribute value), VN (no predicted value when the ground truth contains an attribute value), VC (correct predicted value that exactly matches the attribute value in the ground truth), and VW (incorrect predicted value that does not match the attribute value in the ground truth). The precision (P), recall (R), and F1-score (F1) are calculated using the following formulas: $P = VC/(NV + VC + VW)$, $R = VC/(VN + VC + VW)$ and $F1 = 2PR/(P + R)$.

4 ZERO-SHOT PROMPT ENGINEERING

This section covers a zero-shot scenario, where no training data is available. Firstly, we introduce the structure of our prompt templates. Secondly, we discuss the main challenge for zero-shot natural language prompt templates: how to describe the target schema to the LLMs, i.e. defining which attributes should be extracted. We

[7]https://python.langchain.com/en/latest/index.html

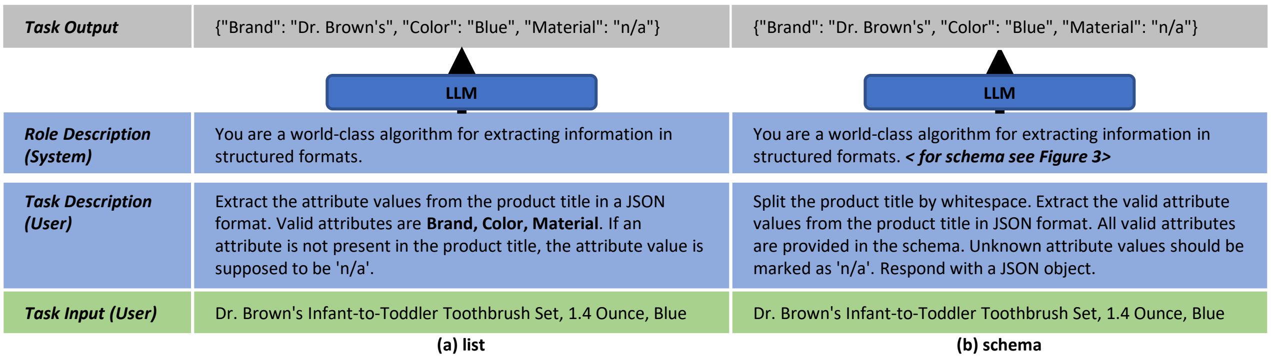

Figure 2: Zero-shot prompt templates `list` and `schema`.

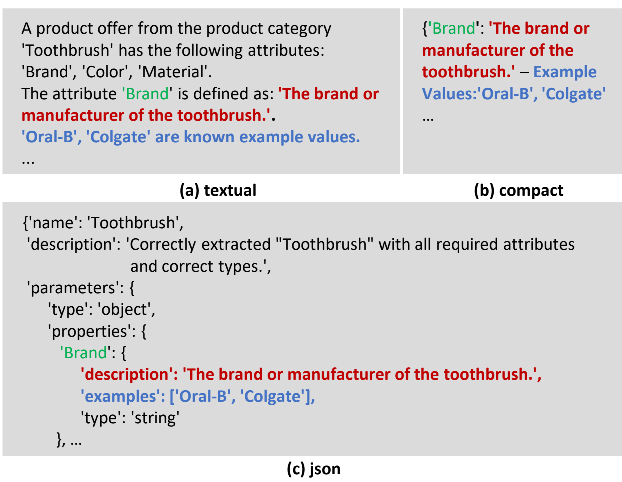

Figure 3: Target schema representations (a) `textual`, (b) `compact` and (c) `json`.

analyze the representation of the target schema along two dimensions: the level of detail of the attribute descriptions and the format in which the target schema is presented.

4.1 Prompt Templates

Prompt design is a key component in tuning an LLM [11, 34]. Accordingly, we develop prompt templates to systematically analyze various representations of the target attributes. All templates request the LLM to extract all known attributes of a product simultaneously to take advantage of synergies between the attributes. The templates consist of four chat messages: role description (blue), task description (blue), task input (green) and task output (grey), as shown in Figure 2. The role description outlines the behaviour of the LLM. The task description provides instructions for attribute/value extraction, including pre-processing the product title, formatting the response as a JSON document, and marking attribute values not present in the product title as 'n/a'. This is crucial for identifying the evaluation scenario NN, where no value is predicted and the ground truth does not contain a value. The task input consists of the product title. Role and task description are static whereas task input and task output change for each extraction.

The chat prompt combines the role description, task description, and task input, with each message having a message type. The role description is defined as `system`, while the task description and input are defined as `user`. The LLM processes the chat prompt and should respond with a JSON document that follows the target schema, making it straightforward to evaluate the task output.

4.2 Representation of the Target Schema

Our goal in this section is to investigate the impact of different representations of the target schema on the extraction of attribute values by LLMs. The evaluation of these representations is based on two dimensions: (i) the level of detail and (ii) the representation format. An attribute in the target schema can be described by its name, a description, and example values. The level of detail increases with additional information about an attribute. The four attribute representation formats that we evaluate are:

- `list` enumerates the names as illustrated in Figure 2 (a).
- `textual` articulates name (green), description (red) and example values in plain text as depicted in Figure 3 (a).
- `compact` densely combines names (green), descriptions (red) and example values (blue) as depicted in Figure 3 (b).
- `json` represents names (green), descriptions (red) and example values (blue) using the JSON schema vocabulary[8] as depicted in Figure 3 (c).

The set of attributes contained in a target schema representation is determined by the product category. On average, OA-Mine and AE-110k define 10 attributes per product category. For demonstration purposes, Figure 2 and Figure 3 display only a subset of the attributes for the 'toothbrush' product category. The prompt template `schema` integrates the attribute representations `textual`, `compact` and `json` into the role description as depicted in Figure 2.

Attribute Descriptions. Since attribute descriptions are not available in the original datasets, we generate descriptions for each attribute using GPT-3.5. The resulting prompts are found in our GitHub repository. In the zero-shot scenario, example values are

[8] https://json-schema.org/

Table 4: F1-scores for zero-shot prompt templates. The highest F1-score per dataset is marked in bold.

Dataset	Model	list	textual	compact	json
OA-Mine	GPT-3.5	63.3	62.7	65.1	64.8
	GPT-4	**69.1**	68.9	68.8	68.1
	Beluga-7B	36.9	39.6	37.6	6.8
	Beluga2	58.3	50.8	36.2	0.0
	SOLAR	60.8	55.2	56.6	11.4
AE-110K	GPT-3.5	61.4	61.3	**63.6**	61.5
	GPT-4	56.3	55.5	53.7	62.1
	Beluga-7B	46.4	47.4	42.7	0.3
	Beluga2	52.5	52.1	38.2	0.0
	SOLAR	52.1	49.2	47.3	11.1

not available as they need to be selected from training data. Section 5.1 discusses the use of example values.

Discussion. As shown in Table 4, overall, GPT-3.5 and GPT-4 achieve zero-shot F1-scores above 60%. While the `compact` representation works best for GPT-3.5, on average GPT-4 benefits most from the `json` representation with attribute descriptions. The difference between the attribute representations without example values is minor for GPT-3.5 and GPT-4. Once we add example values in Section 5, GPT-3.5 and GPT-4 benefit most from the `json` representation. The open-source models perform worse than OpenAI's LLMs on all prompts and do not benefit from the attribute representations with a great level of detail. The `list` prompt works best for open-source models. Looking at the extraction examples for the `list` prompt template in Table 1, we see that GPT-4's background knowledge about brands is sufficient for correctly extracting the brand name 'PENERAN' in examples (d). On the other hand, GPT-4 does not correctly predict the attribute values of the examples (a), (b) and (c) using the `list` prompt template.

In Section 5 we explore how training data can be used by the LLMs to improve their performance.

5 USING TRAINING DATA

In this section, we evaluate a scenario where training data is available. This training data is used for (i) selecting example values to be added to the attribute descriptions as discussed in Section 4.1, (ii) sample demonstrations for in-context learning, (iii) shuffled ensembling to deal with position bias, and (iv) fine-tuning GPT-3.5.

5.1 Example Values

This section explores the effect of adding attribute values from the training set to the target attribute representations introduced in Section 4. The evaluation is conducted in two steps. First, we compare the attribute representations `textual`, `compact` and `json` with example values to the attribute representation `compact` without example values. The highest average F1-score in Section 4 was achieved by the latter. Second, we assess how varying amounts of sampled attribute values impact the performance of the best attribute representation.

Table 5: F1-scores for prompt templates with example values. The highest F1-score per dataset is marked in bold.

Dataset	Model	compact	textual 10-val	compact 10-val	json 10-val
OA-Mine	GPT-3.5	65.1	60.7	61.4	69.8
	GPT-4	68.8	64.8	66.4	**75.0**
	Beluga-7B	37.6	25.6	40.3	4.6
	Beluga2	36.2	55.1	41.4	0.0
	Solar	56.6	12.2	52.0	52.9
AE-110K	GPT-3.5	63.6	55.0	40.5	**74.4**
	GPT-4	53.7	53.5	46.5	70.1
	Beluga-7B	42.7	35.8	37.3	0.2
	Beluga2	38.2	50.8	36.2	0.0
	Solar	47.3	6.0	47.7	46.8

By default, up to 10 unique attribute values are randomly sampled from the small training set for each attribute. If an attribute in the training set has less than 10 unique values, all available values for that attribute will be retrieved. Figure 3 illustrates how the example values are integrated into the attribute representations.

Attribute Representation. Table 5 shows the results of adding attribute values to the different prompts. We see that the average F1 performance of GPT-3.5 and GPT-4 increases by 8% and 11% respectively compared to the `compact` representation without attribute values. It is worth noting that the open-source models do not benefit from the example values. The `list` prompt remains the most effective option for open-source models. Revisiting the extraction examples in Table 1, we see for the prompt template `json-val` that the example values guide the extraction in the right direction in cases (a) and (c). In contrast, example (d) shows that GPT-4 omits the attribute value 'PENERAN', which has been extracted correctly using the simple `list` prompt template. This is likely due to the value not being included in the list of example values.

Amount of example values. We now evaluate the impact of the number of unique example values sampled from the training set on the performance of GPT-3.5 and GPT-4. Up to 3, 5, and 10 example values per attribute are sampled from the small training set. For each configuration, the total number of unique attribute values sampled from the training set (Sampled A/V) and the percentage of attribute values in the test set that are included in the sampled set of attribute values (Seen A/V) are calculated. Table 6 shows that providing either 3 or 10 example values can be harmful to GPT-3.5. In contrast, GPT-4's F1-scores remain consistent, with the best results achieved when 5 example values are provided. The sampled attribute values represent only a small percentage (6% to 28%) of all unique attribute values in the test set. Merely looking up the sampled attribute values is insufficient to achieve F1-scores above 70% for GPT-3.5 and GPT-4. These models must derive general patterns for extraction from the `json-val` prompt template. Furthermore, including example values in the prompt increases the number of tokens and the associated cost. When comparing the cost per 1k extracted attribute/value pairs of GPT-3.5 and GPT-4 to the shortest prompt `list`, it is evident that the `json-val` prompts are three times more expensive.

Table 6: F1-scores for `json-val` prompt templates with different amounts of example values. The highest F1-score per dataset is marked in bold.

Dataset	Prompt	Sampled A/V	Seen A/V	GPT-3.5	GPT-4
OA-Mine	list	0	0%	63.3	69.1
	json-3-val	283	6%	74.4	74.8
	json-5-val	429	9%	74.0	**74.6**
	json-10-val	719	13%	69.8	75.0
AE-110K	list	0	0%	61.4	56.3
	json-3-val	172	20%	66.6	69.5
	json-5-val	222	24%	69.7	72.8
	json-10-val	271	28%	**74.4**	70.1

5.2 In-Context Demonstrations

This section examines the impact of adding in-context learning demonstrations from the training set to the prompts. The analysis covers four perspectives. Firstly, we experiment with different demonstration selectors to choose demonstrations from the training set. Secondly, we evaluate the number of demonstrations selected from the training set. Thirdly, we analyze the effect of the training set size on the performance of LLMs. Lastly, we evaluate the best configuration for GPT-3.5, GPT-4, and Llama2-based LLMs.

Prompt Template. The prompt templates, which were introduced in Section 4, are extended to add demonstrations to the prompts. Figure 4 illustrates this extension. Each demonstration (light blue) consists of a task input and a task output. The demonstrations are added to the chat prompt following the role and task description (blue). The task description (blue) is then repeated, followed by the task input (green). The task input for the demonstration is of message type `user` and the task output is of message type `assistent`. This approach adds demonstrations to both the `list` and the `schema` prompts.

Role Description (System)	You are a world-class algorithm for extracting information in structured formats.
Task Description (User)	Extract the attribute values from the product title in a JSON format. Valid attributes are **Brand, Color, Material**. If an attribute is not present in the product title, the attribute value is supposed to be 'n/a'.
Demonstration – Task Input (User)	Quip Kids Electric Toothbrush Set - Electric toothbrush with multi-use cover (Green)
Demonstration – Task Output (Assistent)	{"Brand": "Quip", "Color": "Green", "Material": "n/a"}
Task Description (User)	Extract the attribute values from the product title in a JSON format. Valid attributes are **Brand, Color, Material**. If an attribute is not present in the product title, the attribute value is supposed to be 'n/a'.
Task Input (User)	Dr. Brown's Infant-to-Toddler Toothbrush Set, 1.4 Ounce, Blue

Figure 4: Prompt template for in-context learning.

Demonstration Selectors. We experiment with five demonstration selectors that apply different strategies to select demonstrations from the training set for each task input. We ensure that the selected demonstrations and the target product offer belong to the same product category. The five demonstration selectors are listed below:

- **Fixed** always selects the same set of demonstrations per category. This mimics a scenario in which only a few demonstrations are available.
- **Random** selects a random set of demonstrations per product category.
- **Semantic Similarity (Sem. Sim.)** embeds the product titles of the training demonstrations using OpenAI's embedding model `text-embedding-ada-002`[9] and selects embedded demonstrations, which have the greatest cosine similarity with the product title in the task input [10].
- **Maximum Marginal Relevance (MMR)** is similar to the Semantic Similarity demonstration selector and optimizes for the diversity of product titles. In addition to the greatest cosine similarity, it adds the demonstrations iteratively and does not add a demonstration if its product title is too similar to already added demonstrations [33].
- **Semantic Similarity & Attribute Value Diversity (AVD)** is similar to the Semantic Similarity demonstration selector and optimizes for the diversity of attribute values. In addition to the greatest cosine similarity, it adds the demonstrations iteratively. It does not add a demonstration if it contains only attribute values that have already been mentioned in an already-added demonstration.

To explore the influence of the demonstration selectors, we evaluate the `list` and `json-val` prompts to understand how the combination of attribute representation with different levels of detail and demonstrations affects the LLM's performance. Both prompts are extended with differently selected demonstrations to the prompts `list-dem` and `json-val-dem` and run with GPT-3.5. The demonstrations are chosen from the small training set. In Section 5.1, we added 3 example values for OA-Mine and 10 example values for AE-110K, sampled from the small training set, to the `json-val-dem` prompt. This configuration was found to work best without demonstrations. We compare the results to prompts with no demonstrations, which are specified by the 'None' selector.

Effectiveness. The results in Table 7 show that GPT-3.5 benefits most from semantically similar demonstrations by up to 16% leading to the best F1-score of 77.9%. When comparing the best results achieved by the `list-dem` and `json-val-dem` prompts, it is evident that the `list-dem` prompt achieves F1-scores that are 1-2% higher than the `json-val-dem` prompt. This suggests that GPT-3.5 does not benefit from combining attribute representation with a high level of detail (as seen in `json-val`) and demonstrations. The differences in F1-score between the demonstration selectors Sem. Sim., MMR and AVD are negligible. For the next experiments, Sem. Sim. will be used as the demonstration selector because filtering the semantically similar demonstrations by their maximum marginal

[9]https://platform.openai.com/docs/guides/embeddings/

relevance and attribute value diversity only has a minimal impact on the LLM's performance.

Table 7: F1-scores for prompt templates with different demonstration selectors. The highest F1-score per dataset is marked in bold.

		OA-Mine		AE-110K		Average	
Model	Selector	list-dem	json-val-dem	list-dem	json-val-dem	list-dem	json-val-dem
GPT-3.5	None	63.3	74.4	61.4	74.4	62.3	74.4
	Fixed	69.8	68.7	71.7	72.6	70.8	70.7
	Random	69.1	69.7	73.4	73.8	71.3	71.8
	Sem. Sim.	76.8	73.1	79.0	77.0	77.9	75.1
	MMR	76.2	76.4	79.4	76.2	77.8	76.3
	AVD	77.2	77.0	78.9	76.6	78.1	76.8
GPT-4	None	63.1	75.7	56.0	70.7	59.6	73.2
	Sem. Sim.	**78.8**	**78.8**	74.4	**83.3**	76.6	**81.1**

We now run the prompts `list-dem` and `json-val-dem` with GPT-4 and the best demonstration selector Sem. Sim.. Table 7 shows that GPT-4 benefits from the selected demonstrations. In contrast to GPT-3.5, GPT-4 can also benefit from the combination of the `json-val` attribute representation and the demonstrations into the prompt template `json-val-dem`, which leads to an average F1-score 6% higher than GPT-3.5's score with the `json-val-dem` prompt.

Extraction Examples. Looking again at the extraction examples in Table 1, GPT-4 correctly extracts the attribute value 'Coffee Chicory' in example (b) when semantically similar demonstrations are provided in the prompt `json-val-dem`. Example (b) is particularly challenging because the 'Flavor' attribute within the 'Coffee' category has 126 unique values. The range of attributes in the OA-Mine dataset is wider than others, with a median of 21 unique values per attribute. The `json-val` prompt template only provides five example values, while the `json-val-dem` demonstrations are selected based on their semantic similarity to the task input and include relevant attribute values for the extraction being performed. Example (b) demonstrates this advantage. Although the target attribute value, 'Coffee Chicory', is not included in the list of example values, it is included in a demonstration selected from the training set.

Amount of Demonstrations. The second dimension we explore is the number of demonstrations selected from the training set. We evaluate GPT-3.5 using only the prompt `list-dem` with 1, 3, 5, 10 and 15 demonstrations selected by Sem. Sim.. Table 8 shows that the highest F1-score of 76.2% is achieved with 10 or 15 demonstrations, which is 2% better than with 3 demonstrations. The usage fees for hosted LLMs, such as GPT-3.5, depend on the number of tokens in the input prompts, among other factors. Thus, increasing the number of demonstrations may result in a significant increase in usage fees. The right columns in Table 8 illustrate this based on the December 2023 OpenAI usage fees[10]. We see that using 10 demonstrations is twice as expensive as using only 3 demonstrations. Using 15 demonstrations further increases the cost without resulting in a higher F1-score.

[10]https://openai.com/pricing

Table 8: F1-scores and extraction cost ($ for 1k A/V pairs) of the `list` prompt template with different amounts of demonstrations. The highest F1-score per dataset is marked in bold.

	OA-Mine		AE-110K		Average	
	F1	$ for 1k A/V pairs	F1	$ for 1k A/V pairs	F1	$ for 1k A/V pairs
1	71.3	0.0520	73.4	0.0507	72.3	0.0513
3	72.3	0.0746	77.3	0.0664	74.8	0.0705
5	72.7	0.0978	78.0	0.0835	75.3	0.0907
10	73.3	0.1558	**79.1**	0.1259	**76.2**	0.1408
15	**73.6**	0.2133	78.8	0.1643	**76.2**	0.1888

Amount of Training Data. We now evaluate the training data efficiency of LLMs. To conduct the analysis, we assess the `list` prompt using 10 demonstrations chosen by Sem. Sim. once from the small (S) and once from the large (L) training set for all LLMs. Results for the `json-val-dem` prompt are reported for GPT-4, as our previous findings suggest that GPT-4 benefits from a combination of an extended attribute representation and demonstrations.

Table 9: F1-scores for selecting demonstrations from either the small training set (S) or the large training set (L). The highest F1-score per dataset is marked in bold.

		OA-Mine		AE-110K		Average	
Prompt	Model	S	L	S	L	S	L
list-dem	GPT-3.5	73.3	74.0	79.1	82.1	76.2	78.1
	GPT-4	78.2	78.0	80.4	82.6	79.3	80.3
	Beluga-7B	59.0	59.0	79.9	80.0	69.4	69.5
	Beluga2	65.7	65.8	84.5	84.5	75.1	75.2
	SOLAR	63.9	63.8	83.5	83.3	63.9	63.8
json-val-dem	GPT-4	80.2	**82.2**	85.5	**87.5**	82.8	**84.9**

Table 9 shows that the LLMs achieve a marginal gain in F1-score if the demonstrations are sampled from the larger training set. GPT-4 gains 2% by using the large training set with the `json-val-dem` prompt template, which is the best average F1-score of 85%.

Comparison to Llama2-based LLMs. Table 9 also reports results for the open-source Llama2-based LLMs Beluga-7B, Beluga and SOLAR. The best-performing open-source LLM, Beluga2, is 10% worse than GPT-4. The smaller Beluga-7B has an average F1-score that is only 5% worse than Beluga2's F1-score. Beluga-7B requires only a single GPU while Beluga2 requires 4. All LLMs achieve higher F1-scores on AE-110K than on OA-Mine, indicating that OA-Mine is more challenging than AE-110K. This is because OA-Mine contains four times more unique attribute values and 82% of the unique attribute values in the test set are unseen in the training demonstrations of OA-Mine, while only 71% of the unique attribute values are unseen in the AE-110K training demonstrations. According to the results, GPT-4 performs better on OA-Mine, indicating that it learns more general extraction rules. On the other hand, the open-source LLMs are better at detecting similar values between the training and test sets, as demonstrated by their performance on AE-110K.

5.3 Ensembling

This section explores the effect of ensembling the output of multiple variants of a prompt to improve extraction results. Recent work has shown that LLMs can bias answers to multiple-choice questions based on the position of the answers in the prompt [8, 14]. We hypothesise that such position bias may also occur for information extraction tasks as we define the attributes of the target schema in a specific order in the prompts. To address the position bias, we run the *list* and *json-val* prompt templates three times with 10 in-context demonstrations. In each run, we shuffle the order of the attributes and make sure that the attributes mentioned in the task description and in the demonstrations follow the same shuffled order. Afterwards, we use majority voting to combine the three outputs and predict a value for each attribute.

Table 10 shows that ensembling slightly improves the extraction performance of GPT-4. The simpler *list* prompt template benefits more than the *json-val* prompt template with a rich attribute representation. Also, the prompts on OA-Mine benefit more than the prompts on AE-110k, probably because OA-Mine is more challenging than AE-110k. It should be noted that an ensemble of three LLMs is three times more expensive than a single run.

Table 10: F1-scores for GPT-4 ensembles with the `list-dem` and `json-val-dem` prompt templates. The highest F1-score per dataset is marked in bold.

		OA-Mine		AE-110k	
Model	Prompt	F1	$ for 1k A/V pairs	F1	$ for 1k A/V pairs
GPT-4	list-dem	78.0	3.0039	82.6	2.8147
	list-dem-ens	82.5	9.7316	83.0	9.1187
	json-val-dem	82.2	6.4411	87.5	6.0354
	json-val-dem-ens	**83.4**	19.3172	**87.6**	18.1007

5.4 Fine-Tuning

In this section, we evaluate how fine-tuning affects the performance of the LLM GPT3.5. First, we compare the performance of fine-tuned GPT3.5 models to the performance of GPT-4 using sample values and demonstrations from the same training data. Second, we investigate whether the model acquires knowledge in the fine-tuning process that is useful for extracting attribute values of products that were not in the training set.

Fine-Tuning Procedure. GPT-3.5 is fine-tuned on the large training sets of OA-Mine and AE-110K. The fine-tuning sets are obtained by formatting the records in the training sets according to the prompt templates `list` and `json-val` as shown in Figure 2, which generate role descriptions, task descriptions, and task inputs for each training record. The task output contains the attribute/value pairs for the respective training record. To complete the prompt `json-val`, 10 example values per attribute are sampled from the training set. The pre-processed datasets are uploaded to OpenAI's fine-tuning API[11], and fine-tuning is triggered for each dataset. The default parameter setting is used, and each fine-tuning procedure runs for three epochs.

Fine-Tuning Performance. We now evaluate the F1-scores and extraction costs of the fine-tuned models and compare them with the performance of a plain GPT-4 using the prompt template `json-val-dem-ens`. The F1-scores in Table 11 show that the fine-tuned GPT-3.5 models perform similarly to the GPT-4 model on both prompts, with an average F1-score of 85%. Using the fine-tuned LLMs results in lower API usage fees than using GPT4: Extracting 1k attribute/value pairs using GPT-4 costs up to 70 times more than using the fine-tuned GPT-3.5 with the `list` prompt. Fine-tuning GPT-3.5 with the `list` prompt costs 9.6$ on OA-Mine and 7.2$ on AE-110K. Considering the cost of fine-tuning and extraction, it is more cost-effective to tune GPT-3.5 with the `list` prompt on the large training set, starting with about 500 attribute/value pairs, than to use GPT-4.

Table 11: F1-scores and the extraction cost ($ for 1k A/V pairs) of fine-tuned GPT-3.5 models and GPT-4. The highest F1-score per dataset is marked in bold.

Dataset	Prompt	Model	F1	$ for 1k A/V pairs
OA-Mine	json-val-dem-ens	GPT-4	83.4	19.3172
	list	ft-GPT-3.5	83.6	0.2600
	json-val	ft-GPT-3.5	**84.5**	1.2307
AE-110k	json-val-dem-ens	GPT-4	**87.6**	18.1007
	list	ft-GPT-3.5	85.7	0.2643
	json-val	ft-GPT-3.5	86.0	0.9709

Generalization. The following analysis investigates whether the fine-tuned GPT-3.5 LLMs can generalize to products and their attribute values that were not included in the training data. This is a crucial requirement for product attribute value extraction methods, given the constant emergence of new products. To conduct this analysis, we apply the two LLMs fine-tuned on OA-Mine with the prompts `list` and `json-val` to AE-110k, and apply the two LLMs fine-tuned on AE-110k to OA-Mine. To establish a baseline, the four fine-tuned LLMs are compared to a plain GPT-3.5 model. The results in Table 12 indicate that the fine-tuned GPT-3.5 LLMs perform, on average, 17% worse than the plain GPT-3.5 LLM. This suggests that the fine-tuned models may not be able to generalize to products with attribute values that were not included in the training sets and may have lost some of their general language comprehension skills that were present before fine-tuning.

For users who use GPT-3.5 or GPT-4 for attribute/value extraction, this presents two scenarios: (i) When performing frequent attribute value extraction on a known set of products, the

[11] https://platform.openai.com/docs/guides/fine-tuning

Table 12: F1-scores of the fine-tuned GPT-3.5 models transferred to the other dataset.

Dataset	Model	list	json-val
OA-Mine	GPT-3.5	63.3	69.8
	ft-GPT-3.5 on AE-110k	43.7	53.9
AE-110K	GPT-3.5	61.4	74.4
	ft-GPT-3.5 on OA-Mine	46.2	55.9

lower token usage fee of the fine-tuned model compensates for the expensive fine-tuning. (ii) If attribute value extraction is infrequent and the products are constantly changing, GPT-4 with the `json-val-dem-ens` prompt template should be preferred.

6 COMPARISON TO PLM-BASED BASELINES

We compare the performance of an ensemble of GPT-4 with the prompt template `json-val-ens`, using example values and demonstrations, to the PLM-based baselines SU-OpenTag[12] [29], AVEQA[13] [24] and MAVEQA[14] [32]. We also run a simple dictionary baseline (Dict.) which checks the containment of exact values from the training set to predict attribute/value pairs. The baselines are fine-tuned separately on the small and large training sets to evaluate training data-efficiency. This comparison is considered fair as the same training sets are used to select the demonstrations for GPT-4.

Absolute Performance. Table 13 shows the F1-scores of the fine-tuned PLM-based baselines on OA-Mine and AE-110k. The best average score of the SU-OpenTag and AVEQA baselines is 6% lower than the GPT-4 average with the `json-val-ens` prompt template using the large training set for fine-tuning and demonstration selection, respectively.

Table 13: F1-scores of the PLM baselines and GPT-4. The small (S) and large training set (L) are used as source for training data. The highest F1-score per dataset is marked in bold.

	OA-Mine		AE-110K		Average	
Model	S	L	S	L	S	L
SU-OpenTag	55.1	73.9	70.6	85.5	62.8	79.7
AVEQA	67.0	78.7	76.8	80.9	71.9	79.8
MAVEQA	23.1	65.7	57.7	76.8	40.4	71.3
Dict.	42.8	53.5	74.5	77.8	58.7	65.6
GPT-4 (json-val-dem-ens)	82.0	**83.4**	85.9	**87.6**	83.9	**85.5**

Training Data-Efficiency. When comparing the performance of PLM-based baselines fine-tuned on small and large training sets, a performance gap of 8% to 31% is observed meaning that the PLM-based methods require a large amount of training data for reaching a good performance. In contrast, GPT-4 demonstrates a performance gap of only 2% between the small and large training sets. When comparing the performance of PLM-based methods using the large training set to that of GPT-4 using the small training set, it is evident that GPT-4 outperforms all PLM-based methods, despite having access to a training set that is only five times smaller.

Unseen Attribute Values. Our final analysis investigates how PLM-based models and GPT-4 perform for attribute values that are not included in the training set, e.g. unseen attribute values. As new products frequently appear in e-commerce scenarios, attribute value extraction methods must be robust concerning unseen values. For this analysis, we measure the performance of GPT-4 using the prompt `json-val-dem-ens` as well as the performance of the PLM-based methods using the subset of the attribute values in the test sets that are not contained in the training sets. Table 14 shows that all methods struggle with unseen attribute values, as evidenced by lower F1-scores compared to those in Table 13. GPT-4 is more robust towards unseen attribute values than PLM-based models. On the small training set, GPT-4 achieves an average F1-score that is 19% higher than the best PLM-based method AVEQA.

Table 14: F1-scores of the PLM-based baselines and GPT-4 on the unseen attribute-value pairs in the test sets. The highest F1-score per dataset is marked in bold.

	OA-Mine		AE-110k		Average	
	S	L	S	L	S	L
Model - Unseen A/V Pairs	1,607	1,073	572	414	-	-
SU-OpenTag	42.0	58.4	32.0	40.9	37.0	49.6
AVEQA	56.6	65.0	49.8	49.3	53.2	57.2
MAVEQA	18.8	42.6	6.2	24.9	12.5	33.7
GPT-4 (json-val-dem-ens)	**76.4**	74.7	**68.8**	59.1	**72.6**	66.9

7 CONCLUSIONS

This paper is the first scientific paper to investigate the potential of LLMs for product attribute value extraction. We propose a wide range of zero-shot and few-shot prompt templates and evaluate them in combination with different LLMs. The overall best average F1-score of 86% was reached by GPT-4 using an ensemble of shuffled prompts that combine knowledge about the target schema in the form of attribute names, attribute descriptions, example values, and demonstrations. GPT-4's top score is 10% higher than the F1 score of the best open-source LLM Beluga-2. We compared PLM- and LLM-based extraction methods and showed that LLMs are more training-data efficient. Given the same amount of training data, GPT-4 outperforms the best PLM-based methods AVEQA and SU-Opentag by 6% F1. Our experiments also show that GPT-4 is more robust to unseen attribute values than PLM-based methods. We showed that fine-tuning GPT-3.5 can result in a similar performance to GPT-4 without fine-tuning, but damages GPT-3.5's ability to generalize to different target schemata.

[12] https://github.com/hackerxiaobai/OpenTag_2019/tree/master
[13] https://github.com/google-research/google-research/tree/master/mave
[14] https://github.com/google-research/google-research/tree/master/mave

REFERENCES

[1] Monica Agrawal, Stefan Hegselmann, Hunter Lang, et al. 2022. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 1998–2022.

[2] Ansel Blume, Nasser Zalmout, Heng Ji, et al. 2023. Generative Models for Product Attribute Extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Association for Computational Linguistics, Singapore, 575–585.

[3] Tom Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 1877–1901.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[5] Rayid Ghani, Katharina Probst, Yan Liu, et al. 2006. Text mining for product attribute extraction. In *ACM SIGKDD Explorations Newsletter*, Vol. 8. Association for Computing Machinery, New York, NY, United States, 41–48.

[6] Akshay Goel, Almog Gueta, Omry Gilon, et al. 2023. LLMs Accelerate Annotation for Medical Information Extraction. In *Proceedings of the 3rd Machine Learning for Health Symposium*. PMLR, 82–100.

[7] Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. 2020. TXtract: Taxonomy-Aware Knowledge Extraction for Thousands of Product Categories. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 8489–8502.

[8] Miyoung Ko, Jinhyuk Lee, Hyunjae Kim, et al. 2020. Look at the First Sentence: Position Bias in Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online, 1109–1121.

[9] Zornitsa Kozareva, Qi Li, Ke Zhai, et al. 2016. Recognizing Salient Entities in Shopping Queries. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 2. Association for Computational Linguistics, Berlin, Germany, 107–111.

[10] Jiachang Liu, Dinghan Shen, Yizhe Zhang, et al. 2022. What Makes Good In-Context Examples for GPT-3?. In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Association for Computational Linguistics, Dublin, Ireland and Online, 100–114.

[11] Pengfei Liu, Weizhe Yuan, Jinlan Fu, et al. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9 (Sept. 2023), 1–35.

[12] Yubo Ma, Yixin Cao, YongChing Hong, and Aixin others. 2023. Large Language Model Is Not a Good Few-shot Information Extractor, but a Good Reranker for Hard Samples! arXiv:2303.08559 [cs].

[13] Bonan Min, Hayley Ross, Elior Sulem, et al. 2023. Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey. *ACM Comput. Surv.* 56, 2 (Sept. 2023), 30:1–30:40.

[14] Harsha Nori, Yin Tat Lee, Sheng Zhang, et al. 2023. Can Generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine. arXiv:2311.16452 [cs].

[15] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs].

[16] Long Ouyang, Jeffrey Wu, and Xu Jiang. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, others (Ed.), Vol. 35. Curran Associates, Inc., 27730–27744.

[17] Tanmay Parekh, I-Hung Hsu, Kuan-Hao Huang, et al. 2023. GENEVA: Benchmarking Generalizability for Event Argument Extraction with Hundreds of Event Types and Argument Roles. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Toronto, Canada, 3664–3686.

[18] Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped Named Entity Recognition for Product Attribute Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK, 1557–1567.

[19] Zhaochun Ren, Xiangnan He, Dawei Yin, et al. 2018. Information Discovery in E-commerce: Half-day SIGIR 2018 Tutorial. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, Ann Arbor MI USA, 1379–1382.

[20] Cathy Shyr, Yan Hu, Paul A. Harris, et al. 2023. Identifying and Extracting Rare Disease Phenotypes with Large Language Models. arXiv:2306.12656 [cs].

[21] Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs].

[22] Damir Vandic, Jan-Willem van Dam, and Flavius Frasincar. 2012. Faceted product search powered by the Semantic Web. *Decision Support Systems* 53, 3 (June 2012), 425–437.

[23] Somin Wadhwa, Silvio Amir, and Byron Wallace. 2023. Revisiting Relation Extraction in the era of Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Vol. 1. Association for Computational Linguistics, Toronto, Canada, 15566–15589.

[24] Qifan Wang, Li Yang, Bhargav Kanagal, et al. 2020. Learning to Extract Attribute Value from Product via Question Answering: A Multi-task Approach. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, Online, 47–55.

[25] Xingyao Wang, Sha Li, and Heng Ji. 2023. Code4Struct: Code Generation for Few-Shot Event Structure Prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Vol. 1. Association for Computational Linguistics, Toronto, Canada, 3640–3663.

[26] Bifan Wei, Jun Liu, Qinghua Zheng, et al. 2013. A SURVEY OF FACETED SEARCH. *Journal of Web Engineering* 12, 1-2 (Nov. 2013), 41–64.

[27] Jason Wei, Yi Tay, Rishi Bommasani, et al. 2022. Emergent Abilities of Large Language Models. *TMLR* (Aug. 2022).

[28] Yuk Wah Wong, Dominic Widdows, Tom Lokovic, et al. 2009. Scalable Attribute-Value Extraction from Semi-structured Text. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, Miami, FL, USA, 302–307.

[29] Huimin Xu, Wenting Wang, Xin Mao, et al. 2019. Scaling up Open Tagging from Tens to Thousands: Comprehension Empowered Attribute Value Extraction from Product Title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5214–5223.

[30] Jun Yan, Nasser Zalmout, Yan Liang, et al. 2021. AdaTag: Multi-Attribute Value Extraction from Product Profiles with Adaptive Decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Online, 4694–4705.

[31] Li Yang, Qifan Wang, Jingang Wang, et al. 2023. MixPAVE: Mix-Prompt Tuning for Few-shot Product Attribute Value Extraction. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 9978–9991.

[32] Li Yang, Qifan Wang, Zac Yu, et al. 2022. MAVE: A Product Dataset for Multi-source Attribute Value Extraction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, New York, NY, USA, 1256–1265.

[33] Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, et al. 2023. Complementary Explanations for Effective In-Context Learning. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 4469–4484.

[34] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, et al. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg Germany, 1–21.

[35] Liyi Zhang, Mingzhu Zhu, and Wei Huang. 2009. A Framework for an Ontology-based E-commerce Product Information Retrieval System. *JCP* 4, 6 (June 2009), 436–443.

[36] Xinyang Zhang, Chenwei Zhang, Xian Li, et al. 2022. OA-Mine: Open-World Attribute Mining for E-Commerce Products with Weak Supervision. In *Proceedings of the ACM Web Conference 2022*. Association for Computing Machinery, New York, NY, USA, 3153–3161.

[37] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, et al. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, New York, NY, USA, 1049–1058.

[38] Tiangang Zhu, Yue Wang, Haoran Li, et al. 2020. Multimodal Joint Attribute Prediction and Value Extraction for E-commerce Product. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online, 2129–2139.