Building Natural Language Interface for Product Search

Vijit Malik
Amazon
vijitvm@amazon.com

Vinayak Puranik
Amazon
puranikv@amazon.com

Anirban Majumder
Amazon
majumda@amazon.com

Vivek Sembium
Amazon
viveksem@amazon.com

ABSTRACT

Automatic extraction of attribute preferences from search queries is a critical problem in providing accurate product recommendations to customer. The task becomes even more challenging in cold-start settings where we do not have any supervised/labelled data available to train ML models. In this work, we implement a novel dataset generation pipeline (LLM-API) that leverages Large Language Models (LLMs), search logs and proprietary product information data from an ecommerce website to create a high quality dataset. Our proposed pipeline of LLM-API is robust as it can generalize to any product category with minimal changes in the LLM prompts. For the problem of converting product search queries to API calls we propose a *multi-task schema generator model* which we train on our generated dataset. Experiments on an internal test set reveals that our proposed model achieves an improvement of ≈ 9.6% and ≈ 5% in Exact Match and Micro-F1 respectively, over competitive baselines. Benchmarking our approach on public test set of search queries further reveals a gain of ≈ 8.6% and ≈ 10.5% in Exact Match and Micro-F1. We further demonstrate that our approach outperforms a state-of-the-art LLM (Claude) applied on our task using few-shot prompting and CoT reasoning, while at the same time, achieves improvement in inference latency.

CCS CONCEPTS

• **Information systems** → **Search interfaces**; • **Computing methodologies** → **Information extraction**.

KEYWORDS

Product Search, Large Language Models, Natural Language Understanding, Query-Refinement extraction

ACM Reference Format:
Vijit Malik, Vinayak Puranik, Anirban Majumder, and Vivek Sembium. 2024. Building Natural Language Interface for Product Search. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3627673.3680070

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3680070

1 INTRODUCTION

Large selection and attractive pricing have made online shopping extremely popular in recent years. Users can search from a vast and diverse catalog of products and purchase them as per their convenience. Despite its popularity, online shopping lacks the *human touch* of an offline store where a trained sales-person can assist users in identifying the right product that suits their need. Navigating through a vast array of products on an e-commerce website can be a daunting task for users. Further, limited capability of search engines in comprehending natural language queries can make the experience quite harrowing.

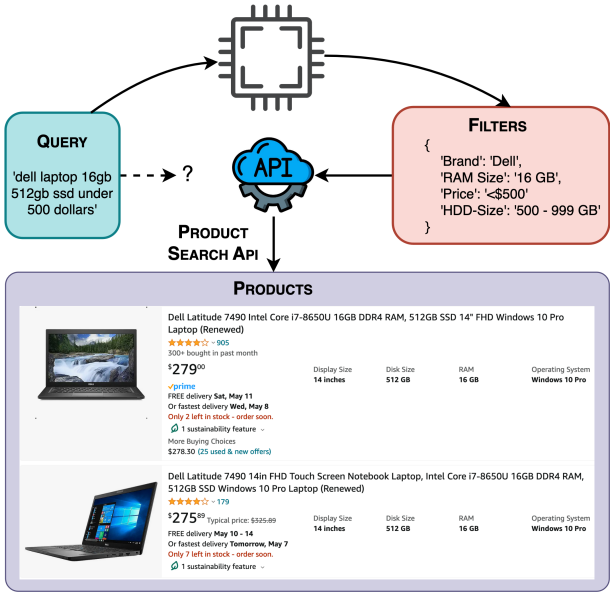

Figure 1: Motivation behind the task of mapping customer query to filters. Having a proprietary system that is able to map the search queries to a set of filters which need to applied, can yield more appropriate search results.

As an example, consider the following scenario: e-commerce catalog contains product attribute information which are key-value pairs listed by the seller e.g. a laptop is described by attributes such as memory size ("RAM=16GB"), processing unit ("CPU=Intel core i9"), storage ("HDD size=512GB") etc. Users mention these product attributes in their search queries to find products that meet their requirement. For example, if a user types *"HP laptop 4g or more, under 1k"* the intent is to search for popular laptops with

"Brand=HP", "RAM >= 4 GB" and "price <= $ 1000" The utterance can be ambiguous ('4g' can mean mobile communication standard 4G or 4GB of RAM) and complex ('under 1k' implies that the price should be below $1000). Traditional product search engines have limited abilities in understanding natural language queries; as a result, when users input intricate queries such as above, the search result may not reflect or adhere to all the specified criteria, resulting in an unsatisfactory user experience.

The advent of Large Language Models (LLMs) in recent years has opened up new avenues for developing sophisticated natural language interfaces for product search. The primary expectation from such an interface is to provide users with the flexibility to formulate queries using natural language, while ensuring that the search results accurately reflect and honor the preferences and criteria specified in those queries. We set few desiderata for such an interface: 1) should accurately extract user's mention of attribute key-values from a complex, ambiguous and noisy natural language query 2) scale the solution across categories with limited to no supervision, 3) should be of low latency so that the solution can be deployed in a live system serving millions of customers.

In this work, we study the problem of designing a natural language interface [**?**], [**?**] for product search. We assume that the underlying search system can be abstracted as an API (referred to as Product Search API) which allows programmatic access to retrieve relevant products based on a search query. The core problem is to extract attribute key-value pairs from noisy, ambiguous and non-canonicalized user query and use them to filter search results returned by the Search API. We specifically make the following contributions in this paper:

1.1 Contributions

(1) We propose a novel automated data-set generation approach using which we generate a large-scale paired data-set of user queries from an e-commerce search engine logs collected from a marketplace and product attributes. Our technique is completely unsupervised and can scale across categories effortlessly.
(2) We introduce a novel Large Language Model (LLM) architecture specifically designed for our task and fine-tuned on our dataset. This architecture outperforms strong baselines by up to +9.6% in exact match accuracy. Our solution offers low latency (approximately 110 milliseconds median latency) and can scale to serve millions of users.
(3) To demonstrate the effectiveness of our approach, we report our findings on a public data-set of search queries where our technique outperforms strong baselines by +8.6% on exact match and +10.5% on Micro-F1 score.

2 RELATED WORK

Large Language Models: Recently, Large Language models [**?**], [**?**], [**?**], [**?**] have shown to perform diverse range of tasks quite effectively with zero-shot and few-shot prompting [**?**], [**?**], [**?**]. Instruction-tuned language models [**?**], [**?**], [**?**], [**?**], [**?**] have shown capabilites of following instructions to solve complex tasks. By converting task descriptions to natural language prompts and injecting them into PLMs (Pre-trained Language Models), prompt-based approaches [**?**], [**?**] leverage task-specific information for better zero-shot results. Even though these models provide ease of building solutions with the use of complex prompting which typically involves In-Context Learning [**?**] and Chain-of-Thought [**?**], the latency of these solutions cannot be ignored. As the complexity and number of tokens needed to fulfill the task increase, the latency increases correspondingly, given the huge size of multiple billions of parameters of these models. In settings where latency plays a major role like product search, adverstisement clicks etc., using these models out of the box is not the best solution.

Product Attribute Extraction is a close line of work that is related to our task and solution. The first learning-based methods for this challenging task required an extensive amount of feature engineering and did not generalize to unknown attributes and values [**?**], [**?**], [**?**]. Recent works have adopted BiLSTM-CRF architectures [**?**] , [**?**] to tag attribute values in product titles, due to the advancement of neural networks. [**?**] employed BERT [**?**] with a mixture-of-experts module to extract attribute values. Additionally, some studies have explored soft prompt tuning to fine-tune a small number of trainable parameters in a language model [**?**], [**?**]. Recently, [new llm product extraction paper] have adopted large language models to for the extraction of product attributes. However, the task of NL2API requires the attributes and their values to be canonicalized in the format as required by the target API.

NL2API: The task of translating natural language inputs into API invocations is a relatively long-standing problem, with initial approaches relying on rule-based methods [**?**]. Only recently, deep learning methods [**? ?**] are found to outperform rule-based techniques on multiple tasks. DNN based solution for mapping natural language inputs to API calls (NL2API) have found application in databases[**?**], knowledge graphs [**?**], and web tables [**?**]. In some works, NL2API is also addressed as a semantic parsing task that maps natural language utterances to executable Web APIs. The main hurdle in the task lies in the lack of training data and the non-trivial nature of the construction of such training data on large scale. We target specifically this aspect of the problem in our work. While crowdsourcing has become a common practice in language-related research, it is particularly challenging and intriguing when it comes to NL2API, because of the complicated interplay of natural language and formal meaning representations as inputs to API.

3 PROBLEM FORMULATION

Given a customer's query regarding product search, q, the customer's desired product category c and its corresponding filter space f_c, the task is to build a model $\mathcal{M}$ to identify customer's *preferred filters schema* $\mathcal{P}$ such that

$$\mathcal{M}(q, c, f_c) = \mathcal{P} = \{k_i : v_i\}, v_i \in f_{(c,k_i)} \quad (1)$$

where, $\{k_i : v_i\}$ are the filter keys and values in $\mathcal{P}$. Note that the filter keys $k \in f_c$ can be of two types which are Free Value (FV) filters and Categorical Value (CV) filters. For a FV filter k_i (for example Price, Colour etc.), the value can be any string (e.g. 'Obsidian Black' for 'Colour') or real value (e.g. '2000' as 'Price'). For a CV filter k_i the value space is ordinal/categorical, with $\{k_i : [v_{ij}]\}$, where k_i is the filter key and $v_{ij} \in f_{(c,k_i)}$ are the possible categories.

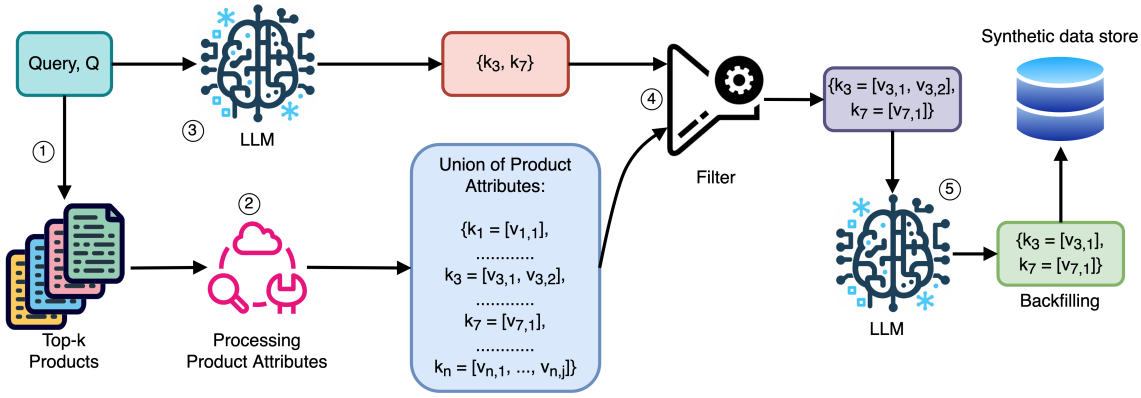

Figure 2: LLM-API Pipeline

4 DATA GENERATION PIPELINE (LLM-API)

In this section, we discuss our proposed novel dataset generation pipeline to obtain (q, c, f_c) tuples and their corresponding preferred filter schema, $\mathcal{P}$ at scale with high accuracy and zero supervision. We rely on the intuition that, for a search query, q, if the customer buys a product in that search session, it is highly likely that the customer found the search results relevant and precise. In order to extract unlabelled data on large-scale, we make use of the search logs and extract the set of customer search queries along with the top-k ($k = 3$) search results (out of which one is a purchased product). We apply additional constraints on the this data (removing noisy out of category queries, queries having at least l_{min} and l_{max} characters, etc.) to make sure that we are getting high quality search queries. In this work, we experiment on 5 product categories[1], however the pipeline is generic and can be extended to other categories as well.

Next, we utilize the zero shot generalization capability of LLMs in a binary classification setting, to determine whether a filter[2] $k \in f_c$, is mentioned in the customer query q. Refer to the prompt we used for filter detection in Prompt 1. Therefore, we obtain the list of filter keys $\{k_i\}$ where $i \in (1, n)$ and n is the number of predicted keys after using LLM zero shot prompting. We also have the top-k search results for the query q from search logs. Now, each product has a set of attribute filter assignments associated with it. We perform a union on the set of filter assignments across the top-k products and obtain the combined filter assignments. Since these products are highly relevant for the query q, it is quite likely that the preferred filters by the customer in q is a subset of the collected filter assignments. Hence, we simply choose the filter assignments corresponding to the customer preferred filters (Figure 2).

The union operation in the second step, provides a list of values (since two products can have different values for the same filter key). In addition, attribute filter assignments for products can be noisy. Therefore, we use LLM for backfilling these filter keys, where we mainly perform two operations. Firstly, we disambiguate the union of values for each filter to pick a single value conditioned on the customer's query q. Secondly, we prompt the LLM to verify if the assignment of a filter to its value is correct provided the query q. To reduce the number of LLM calls, we achieve the aforementioned using a single prompt (refer to Prompt 2).

Our dataset construction pipeline LLM-API can be plugged-in with any arbitrary LLM and is highly dependent on its zero-shot and few-shot capabilities. We benchmark 4 open-source LLMs FLAN-T5 [?], FLAN-UL2 [?], Mistral-7b [?] and Falcon-7b [?] insruct version and empirically determine FLAN-T5 to be the best performing LLM on the task of filter detection (Table 1). Using FLAN-T5, we obtain the preferred filters schema of the customer's query with a relative improvement in exact match of +4.60% (over FLAN-T5 End-to-End baseline) on our Internal test set and 61.2% on our publically available test set (Table 2). Note that we did not use bigger LMs like Claude or ChatGPT due to legal risks associated with training models on their generated data for commercial use. We call this overall data generation pipeline LLM-API using which, we generate about 50k rows of training data for our downstream models.

Model	Precision	Recall	F1
Falcon-7b-Instruct (7B)	_	_	_
Mistral (7B)	+0.093	+0.419	+0.294
FLAN-UL2 (20B)	+0.395	+0.462	+0.303
FLAN-T5-XXL (11B)	**+0.430**	**+0.628**	**+0.578**

Table 1: Performance of open-source models on the task of few-shot Filter Detection. Since these numbers are obtained on the 'Internal' set, we report performance metrics relative to performance of Falcon-7b-Instruct.

5 DATASET DETAILS

Using the LLM-API pipeline with FLAN-T5, we obtain 50k rows of training data (10k per category). We also collect 1000 unlabelled queries (200 per category) and conduct annotations over them to curate our gold standard 'Internal' test set. In the annotation process, we obtain the preferred filters schema $\mathcal{P}$ for each (q, c, f_c) tuple.

[1]To make the setting more challenging with overlapping filter spaces, we choose closely correlated categories of laptops, smartphones, television, air conditioners and refrigerators.

[2]We found that open source LLMs do not perform well on determining the exact filter values as we will see later in the zero-shot prompting results in the experiments section.

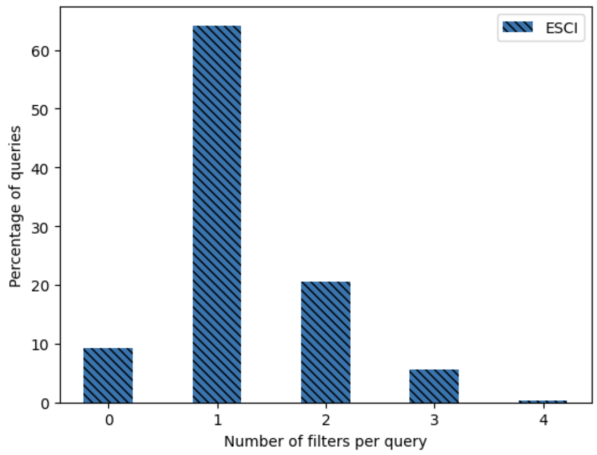

Figure 3: Filter distribution of ESCI test set.

Similarly, we obtain an 'Internal' validation set of 500 queries (100 queries each category) which we used to tune the hyperparameters of our models. Note that the 'Internal' search queries we choose are long queries with lengths of at least 8 tokens long to increase the complexity of the task and train more robust models.

Prompt 1: Filter Detection

Prompt:
"Instruction: You will be given a customer query asking for product recommendations about `{category}`. Your task is to determine whether the customer mentions any preference regarding a particular aspect about the `{category}`. Here are some examples as follows: "

In-context example:
"Query: `{icl_example_query}`
Question: Does the above customer query mention any particular preference regarding `{icl_example_filter}` of the `{category}`?
-Yes
-No
Answer: `{gold_answer}` "

To benchmark our approach on publically available search queries, we use the search queries in the Amazon Shopping Queries [**?**] 'ESCI' dataset. We obtain 268 search queries in the ESCI dataset belonging to our selected set of product categories after some processing on the available data and conduct annotation over them.

We quantify the complexity of both the test sets based on the number of filters in the search queries' annotations. Mean number of filters per query in 'ESCI' test set is 1.2, which is comparatively less than our 'Internal' test set. The distribution of number of filters in ESCI test set is presented in Fig 3. Note that due to our strategy of data collection through LLM-API, we are able to pick up search queries that are longer and more complex than queries in the ESCI set.

6 MODELS

We view the task of mapping customer's search queries, q to their preferred filters schema, $\mathcal{P}$ given the product category c and complete filter space, f_c through the lens of generative models specifically. The task is more complex than NER since not only do we need to detect the filters in the search query, but also, need to be canonicalized in the same namespace as the applicable filters. We categorise the models we used for experimentation into the following categories.

Prompt 2: Backfilling Verification

Prompt:
"Instruction: You will be given a customer query asking for product recommendations about `{category}`. Your task is to determine the customer mentions preference about `{filter}` as which of the following:

`{union_of_filter_values_as_options}`

Answer: "

Baselines: We experiment with zero-shot and few-shot prompting in different settings with Claude-v2 [**?**] and Flan-T5 [**?**]. For Flan-T5, directly zero-shot prompting the LLM to generate preferred filters schema is not feasible since the model does not adhere to the filters namespace or structure. Due to this, we prompt Flan-T5 model in a zero-shot manner filter-by-filter (refer to Section 7.1). For Claude-v2, we experiment with both Chain-of-Thought and no Chain-of-Thought (only ICL) settings. In addition to these approaches, we benchmark our LLM-API pipeline with Flan-T5 as the LLM to compare our pipeline's pseudo-labelling capability compared to other baselines.

Prompt 3: CLAUDE CoT

Prompt:
"Human: <purpose>You are an expert in detecting filters or customer preferences from a customer query. Consider the below filter space below.</purpose>

<filters>
`{filter_space}`
</filters>

<instruction>Given the information in the above dictionary for each category, and the customer query about a product. Your task is to identify the filters that the customer has specified and identify the category of the product that the customer is looking for and construct JSON with filter key as keys, filter values as values.</instruction>

<rules>
<rule>First task is to identify the product category. ALWAYS include that in the response JSON.</rule>
<rule>Second task is to construct the filters JSON</rule>
<rule>Assign only ONE filter value per filter key.</rule>
<rule>Select a filter ONLY if it is mentioned by the customer. You need to be highly precise about which filters are being specified by the customer and not assume a filter.</rule>
</rules> "

In-context example:
"<example>
<query> `{icl_query}` </query>
<thinking> `{icl_thoughts}` </thinking>
<response> `{icl_ground_truth}` </response>
</example> "

Generative models: Using the pseudo-labelled training data from our LLM-API pipeline, we experiment with generative models by fine-tuning them on the training data and testing them on the Internal and ESCI test datasets. We employ the generative models with different input and output types to study their performance on our task (refer to Section 7.2).

Baseline	Dataset	Exact Match (%)	Micro Precision	Micro Recall	Micro F1	Latency (s)
FLAN-T5 End-to-End	Internal	_	_	_	_	2.4582
LLM-API w/ Flan-T5	Internal	+4.60%	+0.0293	+0.0096	+0.0178	15.526
Claude-v2 no-CoT@5	Internal	+6.80%	-0.0462	+0.0589	+0.0123	3.0658
Claude-v2 CoT@5	Internal	**+14.80%**	+0.0178	+0.0910	**+0.0593**	5.8480
FLAN-T5 End-to-End	ESCI	56.34%	0.7739	0.8027	0.788	2.24
LLM-API w/ Flan-T5	ESCI	61.19%	0.8315	0.7936	0.8121	12.22
Claude-v2 no-CoT@5	ESCI	52.99%	0.6846	0.7962	0.7362	2.69
Claude-v2 CoT@5	ESCI	**64.93%**	0.8033	0.9177	**0.8567**	5.53

Table 2: Baselines performance on generating filters schema on ESCI and Internal test datasets. For confidentiality, we report relative improvements over FLAN-T5 End-to-End baseline on our Internal test set.

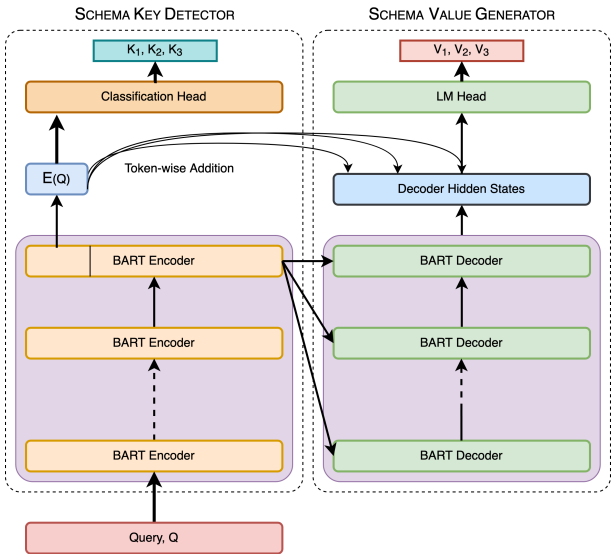

Figure 4: Proposed Multi-task Schema Generator (MTSG) model (best viewed in color).

Hybrid models: Motivated by the performance of Value Generator models in presence of filter space f_c, we decided to build hybrid models. A hybrid model consists of a multi-label classifier, C, which is trained on the task of filter keys prediction and a generative model which predicts the filter values conditioned on the predicted filter keys. Firsly, we train the multi-class classifier to predict the keys present in the query. Next, we train a value generator model with two approaches. In the first approach, we train the model to generate the list of values in a single shot, conditioned on the tuples $(q, c, C(q, c))$. We call this approach as *Multi-Value generation* since we are generating the list of values in a single forward pass. In the second approach, we train the model to generate one value in a single forward pass given only one predicted key. We refer to this approach as *Single-Value generation*. Although this approach has more latency due to multiple forward passes, the simplicity of the task improves the performance as we see in Section 7.3.

Proposed Model (MTSG): Although the single value generation hybrid models outperform the baselines and performs competitively with Claude-v2 few-shot prompting with Chain-of-Thought (Table 4), there is the inconvenience of maintaining two different models. To simplify our approach while maintaining the gain in performance, we propose the Multi-task Schema Generator (MTSG) model. Our proposed model, uses the BART model as the base architecture. BART is an encoder-decoder model where the encoder hidden states have cross attention with the decoder outputs on every layer. We use the BART's Encoder as the shared encoder for both the tasks of key detection and value generation. On top of the encoder we establish a multi-label classification head which performs the filter key detection over the filter space f_c. To encourage the sharing of parameters more explicitly, we establish a connection between the <bos> token embedding to the decoder hidden states using token-wise addition of <bos> token embedding to each token's decoder hidden state (Figure 4). The model is jointly trained on the objective of reducing both the classification loss and the language modelling loss as $L_{loss} = \lambda L_{CL} + (1 - \lambda)L_{LM}$, where L_{CL} is the multi-label classification loss and L_{LM} is the language modelling loss. The parameter λ is tuned on the validation set.

7 EXPERIMENTS

We experiment and obtain results on both the Internal test set and the ESCI test sets. We measure standard metrics like standard Micro Precision, Recall and F1-scores. The former are more lenient metrics, hence we also report Exact Match (EM) % of the extracted filters. Note that a predicted filters schema $\mathcal{P}$ is an exact match with the ground truth, $\mathcal{G}$ only if all the predicted filter values are correct and there are no missing or extra filter keys in $\mathcal{P}$. In other words, exact match is defined as $\frac{1}{N}\sum_{i=1}^{N} I(\mathcal{P}_i = \mathcal{G}_i)$.

7.1 Baselines

Zero-shot prompting and few-shot prompting with Flan-T5 resulted in poor performance due to the complex nature of the task. To mitigate this, we divided the task into two parts. Firstly, we used Flan-T5 to detect which filter keys are customer's preferred keys in f_c. For this task, we prompted the LLM $|f_c|$ times in a key-by-key manner. Now, for each predicted key, k, we again prompted Flan-T5 to provide us the filter values conditioned on the value space $f_{(c,k)}$. Note that we use this baseline of Flan-T5 End-to-End to calculate relative improvements of other baselines/proposed approaches on our Internal test set. Using the Flan-T5 end-to-end approach, we obtained an exact match of 56.34% (Table 2) on the ESCI test set.

Setting	Target	Exact Match (%)	Micro Precision	Micro Recall	Micro F1	Latency (s)
Key-Value Generation	Internal	-12.10%	+0.0408	-0.1885	-0.1133	0.1790
Value Generation	Internal	+4.70%	+0.0480	+0.0254	+0.0349	0.1394
Value Generation w/ f_c	Internal	**+4.80%**	+0.0491	+0.0287	**+0.0374**	0.1367
Key-Value Generation	ESCI	61.19%	0.8351	0.7803	0.8068	0.1315
Value Generation	ESCI	**63.67%**	0.8720	0.8571	0.8645	0.1003
Value Generation w/ f_c	ESCI	63.28%	0.8758	0.8729	**0.8744**	0.0978

Table 3: Performance of BART-large models fine-tuned on text generation task upon pseudo-labelled queries with LLM-API (w. FLAN-T5) in different settings. We record the performance and latency on Internal and ESCI test datasets separately. For Internal set, we report relative improvements over FLAN-T5 End-to-End baseline in Table 2.

Approach Type	Models	Dataset	Exact Match (%)	Micro Precision	Micro Recall	Micro F1	Latency (s)
Multi-Value generators	Keysource = LLM-API	Internal	+6.50%	+0.0399	+0.0460	+0.0438	0.1447
Multi-Value generators	Keysource = Student	Internal	+9.60%	+0.0317	+0.0984	+0.0697	0.1230
Multi-Value generators†	Keysource = Oracle†	Internal	+23.00%	+0.0784	+0.2038	+0.1474	0.1491
Single-Value generators	Keysource = LLM-API	Internal	+13.90%	+0.0337	+0.1162	+0.0803	0.2175
Single-Value generators	Keysource = Student	Internal	**+16.60%**	+0.0602	+0.1397	**+0.1051**	0.2517
Single-Value generators†	Keysource = Oracle†	Internal	+27.70%	+0.0803	+0.2232	+0.1581	0.2252
Proposed Model	MTSG-BART-large	Internal	*+14.20%*	*+0.0145*	*+0.1118*	*+0.0688*	*0.1082*
Baseline	LLM-API w/ Flan-T5	Internal	+4.60%	+0.0293	+0.0096	+0.0178	15.53
Baseline	Claude-v2 CoT@5	Internal	+14.80%	+0.0178	+0.0910	+0.0593	5.848
Multi-Value generators	Keysource = LLM-API	ESCI	64.18%	0.7976	0.9306	0.859	0.0985
Multi-Value generators	Keysource = Student	ESCI	66.67%	0.8595	0.9007	0.8796	0.0887
Multi-Value generators†	Keysource = Oracle†	ESCI	79.48%	0.9649	0.9649	0.9649	0.0994
Single-Value generators	Keysource = LLM-API	ESCI	65.67%	0.8000	0.9315	0.8608	0.1198
Single-Value generators	Keysource = Student	ESCI	**70.15%**	0.863	0.9659	0.9115	0.1302
Single-Value generators†	Keysource = Oracle†	ESCI	80.60%	0.9653	0.9619	0.9636	0.1072
Proposed Model	MTSG-BART-large	ESCI	*69.78%*	*0.8926*	*0.9438*	***0.9174***	*0.0993*
Baseline	LLM-API w/ Flan-T5	ESCI	61.19%	0.8315	0.7936	0.8121	12.22
Baseline	Claude-v2 CoT@5	ESCI	64.93%	0.8033	0.9177	0.8567	5.532

Table 4: Performance of our proposed models and approaches on generating filters schema on ESCI and Internal test datasets. We also show comparison with two competitive baselines of LLM-API with Flan-T5 and Claude-v2 5-shot prompting with Chain-of-Thought. Results which are marked with †, are based on hypothetical Oracle which supplies the generative model with gold filter keys. We provide this scenario only for comparison and hence those results are not highlighted in bold. Our proposed model (MTSG) results are highlighted in italics. Here also, for our Internal set, we report relative improvements over FLAN-T5 End-to-End baseline in Table 2.

We run our proposed LLM-API pipeline with Flan-T5 as the LLM on the test sets and obtain the performance numbers. We can see that our proposed data curation pipeline as baseline obtains an exact match of +4.60% and 61.19% on the Internal and ESCI test datasets. This suggests that our pseudo-labelling approach with LLM-API results in better training examples for our smaller models to learn from.

Next, we use the Claude-v2 model's few-shot prompting technique to generate preferred filters schema from customer's query. We employ best prompting practices with presenting the information and providing the rules in XML tags for the model to understand the prediction setting better. We report performances with 5 examples with Chain-of-Thought and without Chain-of-Thought prompting. As indicated in Table 2, Chain-of-Thought prompting achieves good performance on the both the datasets with exact matches of +14.8% and 64.93% on Internal and ESCI test datasets. Although the performance of Claude is quite promising, the latency is about 6 seconds, rendering it quite inefficient. This motivated us to experiment with smaller models which are not computationally expensive.

7.2 Generative models

We train our generative models specifically BART-large on the task of filter schema generation in three settings. In the first setting of 'Key-Value generation', the input sequence consists of the customer query and the product category, and the target sequence consists

of both filter keys and values separated by delimiter ';'. In another setting 'Value Generation', we keep the same input sequence as the former, but in the target sequence we only generate filter values[3]. In the next setting, we include the complete filter space, f_c in the input sequence along with the customer query and product category and then we generate filter values. We symbolize this setting as 'Value Generation w f_c'.

In Table 3, we show the results of all three settings on both Internal and ESCI datasets. Empirically, we observe that the Key-Value Generation setting does not work well compared to Value generation settings. The best performing generative setting is when we include the filter space in the input sequence and generate the filter values (see Table 3). The results indicate that including the key names in the input sequence positively affects the model and boosts the performance.

7.3 Hybrid models

For building hybrid models, first step is to train a multi-label classifier on the LLM-API pseudo-labelled train set. We choose BERT-large model on the task of filter key prediction and use it in our further experimentation.

As mentioned in section 6, we explore two approaches in hybrid models, Multi-Value and Single-value generation. In both the approaches, we hypothesized that the filter key source during prediction step have a significant effect on the performance of the hybrid models. To study this, we experiment with three different predicted key sources, which are LLM-API predicted keys, Student model predicted keys (BERT-large trained on LLM-API outputs) and Oracle keys. The oracle setting is a hypothetical setting, which assumes access to gold filter keys from the test data and the generative model is tasked to predict their corresponding filter values.

We present our findings in Table 4, where we can observe that the hypothetical oracle setting obtains much higher exact match of +27.7% and 80.6% on Internal and ESCI datasets respectively. Our best performing hybrid model with Single-value generation approach and keys obtained from student model, obtains an exact match of +16.6% and 70.15% on Internal and ESCI datasets respectively. Note that our approach of single value generation also outperforms Claude-v2 with CoT baseline by 1.8% and 5% on both test sets with over 95% reduction in latency.

7.4 Multi-Task Schema Generator (MTSG)

Using the architecture described in Section 6, we optimize the hyperparameter λ which assigns the weight of the two tasks (classification and generation). We perform a linear sweep over the values of λ from 0.1 to 0.9 with the intervals of 0.1 and observe the best performance on the validation set. Our proposed approach with BART-large performs competitively even against the single value generative setting, obtaining an exact match of +14.2% and 69.78% on Internal and ESCI test sets respectiveky. We empirically observe that $\lambda = 0.4$ performs the best on the validation set and report our metrics on the same. Our proposed approach performs the best after the Single value generation hybrid models predicted by student model with over 55% more reduction in latency. Our model not only outperforms the LLM-API baseline significantly, but can also be trained end-to-end, easier to maintain and provides a comparable performance against the best performing models.

[3]Feature values by themselves can be mapped to their feature keys deterministically.

Customer Query	Predicted Filters
dell ryzen 5 laptop 16gb ram 512 ssd 4gb grafic	{'Brand': 'Dell', 'CPU Type': 'AMD Ryzen 5', 'HDD-Size': '500 - 999 GB', 'Dedicated Graphics Memory': '4 to 5 GB', 'RAM Size': '16 GB'}
motorola mobile phones under 500	{'Brand': 'Motorola', 'Price Upper Limit': '$ 500'}
samsung led smart tv 43 inch google	{'Brand': 'Samsung', 'Display Size': '40 - 47 in', 'Smart TV Platform': 'Google TV', 'Display Technology': 'LED'}
lg dual inverter ac 1.5 ton copper ocean black	{'Brand': 'LG', 'Color': 'Ocean Black', 'Capacity': '1.4 to 1.5 Tons'}

Table 5: Example predictions from our proposed MTSG model across different product categories, showing its innate query understanding capabilities.

8 QUALITATIVE STUDY

We provide example predictions of our model in Table 5. Note that our approach is able to predict complex filter sets involving multiple customer preferences. We can also see innate understanding of the model in binning the '43 inch' value rightly into the '40 - 47 inch' filter value. The provided examples also highlight the capability of the model in accurately determininng non-categorical values of 'Price Upper Limit' and 'Color'.

We present a case study analyzing the impact of search results from a Search API by comparing the results obtained from (1) using the customer's query as keywords, and (2) extracting and applying the customer's preferred filters while keeping other parameters the same. Since we need to determine the relevance of the products manually, we conduct this qualitative study only on a small set of query examples, randomly sampled from the test set and analyze top 10 search results per query. For the query, 'laptop brand HP price less than 400', we observe that 100% products are HP laptops and 80% products satisfying price limit. In case of the query 'laptop gaming ram 16gb', 100% products retrieved by querying with our techniques satisfy the customer preferences exactly. We conclude that extracting filters from search query and applying them on search API has positive impact towards relevance of search results.

9 CONCLUSION

We leverage large language models and search logs to generate accurate supervised data for product categories. A student model trained on this pseudo-labeled data outperforms baselines in Exact Match. We also propose a novel multi-task architecture whichw extracts preferred filters from customer utterances. With good training data, smaller pre-trained language models can outperform LLMs in performance and latency. Future work includes self-training on generated data and extending to other APIs.

REFERENCES

[] [n. d.]. A New Open Source Flan 20B with UL2. https://www.yitay.net/blog/flan-ul2-20b.

[] OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim'on Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Adeola Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing, Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W. Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin D. Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. GPT-4 Technical Report. https://api.semanticscholar.org/CorpusID:257532815

[] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The Falcon Series of Open Language Models. arXiv:2311.16867 [cs.CL] https://arxiv.org/abs/2311.16867

[] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, John Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, Benjamin Mann, and Jared Kaplan. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *ArXiv* abs/2204.05862 (2022). https://api.semanticscholar.org/CorpusID:248118878

[] Ansel Blume, Nasser Zalmout, Heng Ji, and Xian Li. 2023. Generative Models for Product Attribute Extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Mingxuan Wang and Imed Zitouni (Eds.). Association for Computational Linguistics, Singapore, 575–585. https://doi.org/10.18653/v1/2023.emnlp-industry.55

[] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[] Robson A. Campêlo, Alberto H. F. Laender, and Altigran S. da Silva. 2023. Using Knowledge Graphs tonbsp;Generate SQL Queries fromnbsp;Textual Specifications. In *Advances in Conceptual Modeling: ER 2023 Workshops, CMLS, CMOMM4FAIR, EmpER, JUSMOD, OntoCom, QUAMES, and SmartFood, Lisbon, Portugal, November 6–9, 2023, Proceedings* (<conf-loc content-type="InPerson">Lisbon, Portugal</conf-loc>). Springer-Verlag, Berlin, Heidelberg, 85–94. https://doi.org/10.1007/978-3-031-47112-4_8

[] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[] Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *ArXiv* abs/2210.11416 (2022). https://api.semanticscholar.org/CorpusID:253018554

[] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. arXiv:2210.11416 [cs.LG] https://arxiv.org/abs/2210.11416

[] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *SIGKDD Explor. Newsl.* 8, 1 (jun 2006), 41–48. https://doi.org/10.1145/1147234.1147241

[] Saghar Hosseini, Ahmed Hassan Awadallah, and Yu Su. 2021. Compositional Generalization for Natural Language Interfaces to Web APIs. *ArXiv* abs/2112.05209 (2021).

[] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics* 8 (2020), 423–438. https://doi.org/10.1162/tacl_a_00324

[] Mayank Kothyari, Dhruva Dhingra, Sunita Sarawagi, and Soumen Chakrabarti. 2023. CRUSH4SQL: Collective Retrieval Using Schema Hallucination For Text2SQL. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 14054–14066. https://doi.org/10.18653/v1/2023.emnlp-main.868

[] Zornitsa Kozareva, Qi Li, Ke Zhai, and Weiwei Guo. 2016. Recognizing Salient Entities in Shopping Queries. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Katrin Erk and Noah A. Smith (Eds.). Association for Computational Linguistics, Berlin, Germany, 107–111. https://doi.org/10.18653/v1/P16-2018

[] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116* (2023). arXiv:2306.01116 https://arxiv.org/abs/2306.01116

[] 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*, Regina Barzilay and Mark Johnson (Eds.). Association for Computational Linguistics, Edinburgh, Scotland, UK., 1557–1567. https://aclanthology.org/D11-1144

[] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[] Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. 2022. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. (2022). arXiv:2206.06588

[] Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (2021). https://api.semanticscholar.org/CorpusID:231925131

[] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 4222–4235. https://doi.org/10.18653/v1/2020.emnlp-main.346

[] Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark Encarnacion. 2017. Building Natural Language Interfaces to Web APIs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 177–186. https://doi.org/10.1145/3132847.3133009

[] Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table Cell Search for Question Answering. In *Proceedings of the 25th International Conference on World Wide Web* (Montréal, Québec, Canada) *(WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 771–782. https://doi.org/10.1145/2872427.2883080

[] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier García, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2022. UL2: Unifying Language Learning Paradigms. In *International Conference on Learning Representations*. https://api.semanticscholar.org/CorpusID:252780443

[] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv* abs/2307.09288 (2023). https://api.semanticscholar.org/CorpusID:259950998

[] Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. GPT-NER: Named Entity Recognition via Large Language Models. *ArXiv* abs/2304.10428 (2023). https://api.semanticscholar.org/CorpusID:258236561

[] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *ArXiv* abs/2201.11903 (2022). https://api.semanticscholar.org/CorpusID:246411621

[] Yuk Wah Wong, Dominic Widdows, Tom Lokovic, and Kamal Nigam. 2009. Scalable Attribute-Value Extraction from Semi-structured Text. In *2009 IEEE International Conference on Data Mining Workshops*. 302–307. https://doi.org/10.1109/ICDMW.2009.81

[] W. A. Woods. 1973. Progress in Natural Language Understanding: An Application to Lunar Geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition* (New York, New York) *(AFIPS '73)*. Association for Computing Machinery, New York, NY, USA, 441–450. https://doi.org/10.1145/1499586.1499695

[] Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 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 (Volume 1: Long Papers)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 4694–4705. https://doi.org/10.18653/v1/2021.acl-long.362

[] Li Yang, Qifan Wang, Jingang Wang, Xiaojun Quan, Fuli Feng, Yu Chen, Madian Khabsa, Sinong Wang, Zenglin Xu, and Dongfang Liu. 2023. MixPAVE: Mix-Prompt Tuning for Few-shot Product Attribute Value Extraction. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 9978–9991. https://doi.org/10.18653/v1/2023.findings-acl.633

[] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 1321–1331. https://doi.org/10.3115/v1/P15-1128

[] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1049–1058. https://doi.org/10.1145/3219819.3219839

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009