API-BLEND: A Comprehensive Corpora for Training and Benchmarking API LLMs

Kinjal Basu*, **Ibrahim Abdelaziz***, **Subhajit Chaudhury, Soham Dan, Maxwell Crouse, Asim Munawar, Vernon Austel, Sadhana Kumaravel, Vinod Muthusamy, Pavan Kapanipathi, and Luis A. Lastras**
IBM Research
{kinjal.basu, ibrahim.abdelaziz1, subhajit, soham.dan, maxwell.crouse, asim}@ibm.com
sadhana.kumaravel1@ibm.com, {austel, vmuthus, kapanipa, lastrasl}@us.ibm.com

Abstract

There is a growing need for Large Language Models (LLMs) to effectively use tools and external Application Programming Interfaces (APIs) to plan and complete tasks. As such, there is tremendous interest in methods that can acquire sufficient quantities of train and test data that involve calls to tools / APIs. Two lines of research have emerged as the predominant strategies for addressing this challenge. The first has focused on synthetic data generation techniques, while the second has involved curating task-adjacent datasets which can be transformed into API / Tool-based tasks. In this paper, we focus on the task of identifying, curating, and transforming existing datasets and, in turn, introduce API-BLEND, a large corpora for training and systematic testing of tool-augmented LLMs. The datasets mimic real-world scenarios involving API-tasks such as API / tool detection, slot filling, and sequencing of the detected APIs. We demonstrate the utility of the API-BLEND dataset for both training and benchmarking purposes[1].

1 Introduction

Large Language Models (LLMs) have shown remarkable abilities across a variety of Natural Language Understanding (NLU) tasks (Min et al., 2023), e.g., text generation (Brown et al., 2020; Radford et al., 2019), summarization (Zhang et al., 2020; Beltagy et al., 2020), and mathematical reasoning (Imani et al., 2023). There has been strong recent interest in enabling LLMs to call APIs or external tools (such as calculators, calendars, or web searches (Hao et al., 2023; Qin et al., 2023; Tang et al., 2023a)) to accomplish high level tasks like booking a hotel, reserving a table, and automating a job requisition tasks. These higher-level tasks are generally conversational and complex. However, in order to perform such complex tasks, LLMs should be able to perform simpler tasks with APIs such as (a) APIs detection: Based on a user query, correctly choose which API to call, (b) Slot filling[2]: Given the API, extract either the slots/parameters from the user utterances or request from the user more information to fill the required parameters of the detected API, and (c) Sequencing: Given an utterance specifying a task, write the sequence of APIs that needs to be called to accomplish the task.

Data for the above-mentioned API tasks, both for training and benchmarking LLMs has been scarce. Addressing this issue, in the recent past,

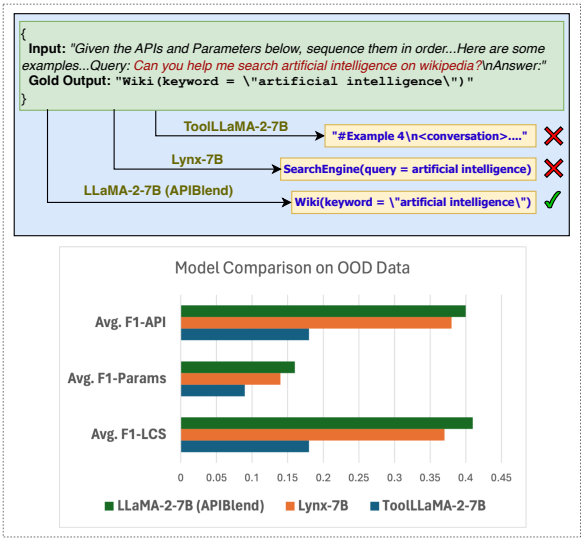

Figure 1: *Top*: an example from the API Bank-Level1 dataset (OOD) that showcases LLaMA-2-7b fine-tuned with API-BLEND generates the correct API and parameter, whereas the other models hallucinate. *Bottom*: performance comparison among three models of similar sizes; two recent tool-augmented models (Lynx and ToolLLaMA-2-7B) and a LLaMA-2-7B model trained with API-BLEND (API-BLEND-LLaMA-2-7B), which significantly outperforms the other two models.

*These authors contributed equally to this work

[1] API-BLEND data generation code can be accessed here: https://github.com/IBM/API-BLEND

[2] In this paper, *slot* and *input parameters* are used interchangeably.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12859–12870
August 11-16, 2024 ©2024 Association for Computational Linguistics

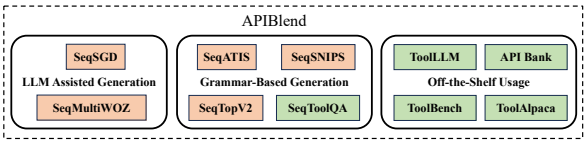

Figure 2: API-BLEND Datasets: 10 datasets, 6 curated as part of this paper and 4 are off-the-shelf datasets used for out-of-domain testing.

most approaches have relied on synthetically generated data for training API-augmented LLMs. For instance, ToolLLM (Qin et al., 2023) produces multi-sequence REST-API data sourced from GPT-4 (Achiam et al., 2023), while datasets like Gorilla (Patil et al., 2023) utilize synthetic, single-sequence API data, specifically Deep Learning libraries' APIs, generated from language models.

Although generating synthetic data from LLMs offers a cost-effective means of obtaining substantial training data, they suffer from several drawbacks. First, data generation methods are plagued by critical issues such as bias inherent in the sampling model, and a lack of diversity in the training dataset. Previous work has shown synthetically generated data suffers from lack of diversity (Gupta et al., 2023), and diverse data can improve out-of-domain (OOD) generalization (Yu et al., 2022). Consequently, models trained on such data can overfit on in-distribution APIs and struggle to generalize to OOD APIs that were not encountered during training, restricting the broad applicability of LLMs for tool usage. In addition, datasets have primarily included API detection (single and multiple) and Slot filling in different settings, whereas Sequencing, a prominent task to perform higher-level human tasks using APIs has rarely been the focus in existing works. Lastly, datasets in domains such as digital assistants and semantic parsing that are related to API-tasks and are human-annotated have gone unnoticed in literature due to the emergence of synthetic data generation techniques;

In light of the above issues, we have developed an API-focused training dataset that leverages a hybrid approach for data generation. This is built upon five human-annotated datasets with LLM-assisted generation comprising of over 150k, 17k, and 11k train, development, and test instances. The transformation primarily focuses on sequencing, including API detection and slot-filling due to the sparsity and importance of sequencing data for training models. Furthermore, these datasets are collected from different domains such as semantic parsing, dialog, and digital assistants resulting in a higher diversity of API data. We show that models trained on this diverse dataset yield significantly better OOD generalization performance compared to other state-of-the-art methods, with an example shown in Figure 1. As an evaluation/benchmarking dataset, we include five different existing benchmarks for OOD evaluation. In conclusion, we release API-Blend, a comprehensive API training, and benchmarking dataset, comprising of 10 datasets (5 for training, and 5 for OOD testing), see Figure 2.

2 Related Work

2.1 Tool-Usage by LLMs

Many recent works (Komeili et al., 2022; Thoppilan et al., 2022; Gao et al., 2023; Schick et al., 2023) have explored how to address the susceptibility of current LLMs to certain errors (e.g., arithmetic (Patel et al., 2021)) through the use of external tools. Such tools can be called by an LLM to provide itself with access to up-to-date information (Komeili et al., 2022; Schick et al., 2023), perform mathematical operations (He-Yueya et al., 2023), and even execute formal programs (Gao et al., 2023).

Early approaches to general-purpose training of LLM tool-use leveraged large amounts of human-annotated data (Komeili et al., 2022; Thoppilan et al., 2022). The difficulty in scaling these approaches was addressed by later works, which utilized self-supervision (Schick et al., 2023; Parisi et al., 2022) and few-shot prompting (Yao et al., 2022). The prompting framework of (Yao et al., 2022) has become widely used when augmenting LLMs with tools, with many follow-up works exploring how to improve its cost-effectiveness (Xu et al., 2023a), performance (Shinn et al., 2023; Yang et al., 2023), and data generation quality (Qin et al., 2023; Tang et al., 2023b).

The utility of tool-calling itself has been explored with many standard benchmarks for question answering (Saikh et al., 2022; Yang et al., 2018), mathematical reasoning (Cobbe et al., 2021), machine translation (Scarton et al., 2019; Lewis et al., 2020), and planning (Shridhar et al., 2020). While useful to serve as a comparison against task-specific, supervised methods, it is unclear to what extent these datasets actually require the usage of tools. As observed by (Zhuang et al., 2023), such benchmarks do not adequately distinguish be-

tween problems that can be solved using only an LLM's internal knowledge and those that can only be solved through tool calls.

2.2 API Datasets

The first self-supervised approaches to constructing tool-use datasets (Schick et al., 2023; Parisi et al., 2022) focused on a small set of general-purpose tools. Soon after, tool-use was quickly expanded to general API function calling (Qin et al., 2023; Tang et al., 2023b; Patil et al., 2023), where the volume and diversity of APIs and scenarios were instead emphasized. While all of the aforementioned datasets highlight the number of APIs involved in their respective corpora, they each vary in terms of how those API calls are utilized. For instance, some datasets curate scenarios involving only a single API call (Tang et al., 2023b; Patil et al., 2023; Xu et al., 2023b) while others involve multiple calls (Qin et al., 2023; Hao et al., 2023). In addition, some require actual calls to a real API to solve their problems (Qin et al., 2023; Li et al., 2023a; Xu et al., 2023b), which contrasts with other works that simulate API calls with a prompted LLM (Tang et al., 2023b; Patil et al., 2023).

A limitation of the above-listed self-supervised corpora lies in the evaluation of API-use scenarios. Some approaches evaluate based on hallucination rate (Patil et al., 2023) while others rely on a separate LLM to assess the quality of an example (Tang et al., 2023b; Qin et al., 2023). Recent works have focused on this issue, with Farn and Shin (2023) relying on smaller sets of manually collected ground truth annotations and Huang et al. (2023) performing manual inspection of generated data.

3 API-BLEND Dataset Curation

We focus on the setting where the input is a single natural language utterance and the output is a sequence of API calls with their parameter names and values. API-BLEND consists of datasets created via the following three approaches: (1) Language Model Assisted approach where prompts are used based on existing API outputs, (2) a grammar rule-based approach to convert existing semantic parsing and personal assistant notations into API data, and (3) off-the-shelf datasets. Table 1 depicts the statistics of each dataset and the details of the approach/dataset is below.

Datasets	Train	Dev	Test	Avg. Seq. Len	Avg. No. Params
SeqATIS	11,670	694	774	2.13	4.85
SeqSGD	6,782	1,092	1,567	2.44	3.5
SeqSNIPS	39,750	2,198	2,199	1.96	5.06
SeqMultiWOZ	6,816	485	983	2.36	3.67
SeqTopV2	94,458	13,477	6,095	1.2	1.98
Total	**159,476**	**17,946**	**11,618**		
ToolLLM-G1	-	-	197	2.28	-
ToolLLM-G2	-	-	197	2.55	-
ToolLLM-G3	-	-	97	2.91	-
API Bank-1	-	-	386	1.65	2.25
API Bank-2	-	-	71	1.34	2.44
ToolBench-HS	-	-	100	7.01	0.86
ToolBench-B	-	-	120	9.45	0.89
SeqToolQA	-	-	358	2.42	1.45
ToolAlpaca	-	-	211	1.38	2.01
Total	-	-	**1737**		

Table 1: API-BLEND Datasets Statistics: datasets colored in red are used for training and in-domain testing, while the green ones are used for OOD testing only

3.1 Language Model Assisted Generation

SeqSGD: We created SeqSGD, a dataset based on Schema-Guided Dialogue (SGD) (Rastogi et al., 2020) dataset tailored towards API sequence evaluation. SGD contains about 20k annotated conversations between a human and a virtual assistant. These dialogues consist of engagements with various services and APIs across 20 domains, including banks, events, media, calendar, travel, and weather. To convert this dataset, for each conversation, we prompted a pretrained FLAN-T5-XXL[3] model to convert each API into a request in natural language. We then append the corresponding text of each API to generate the summarized utterance. Figure 3 shows an example. We did not summarize the conversation itself, because it also contains API execution results, and using this as input to a summarization model resulted in many noisy details. To make sure the generated text captures all APIs and parameter values, we post-process the dataset to remove any examples where the utterance does not correctly reflect the ground truth APIs. As a result of this process, we generated 6.8K train, 1.1K validation, and 1.6K test examples having an average API count of 2.44 and an average parameters count per API of 3.5.

SeqMultiWoz: MultiWoz (Ye et al., 2021) is another multi-domain task-oriented dialogue dataset. Following the same process of curating SeqSGD from the SGD dataset, we created SeqMultiWoz, another API dataset based on MultiWoz. The resulting dataset includes about 6.8k train, 485 val-

[3] https://huggingface.co/google/flan-t5-xxl

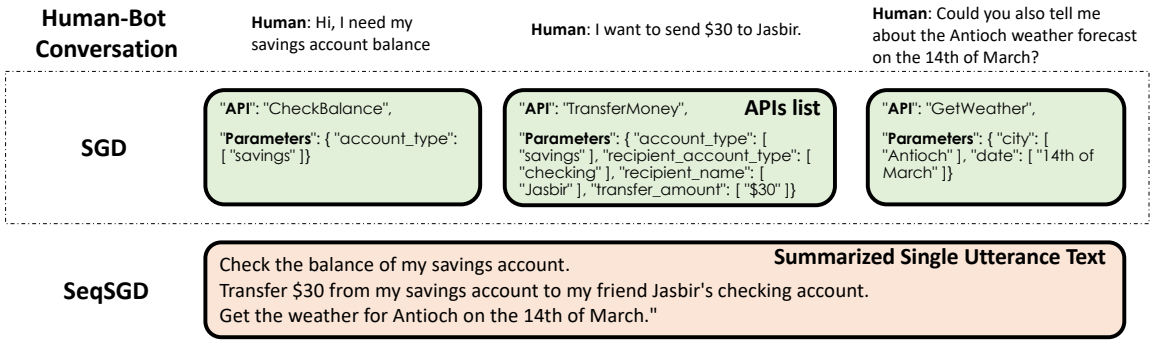

Figure 3: Example of the creation process of seqSGD. Starting from the list of APIs, we use few-shot prompting to generate the summarized single utterance.

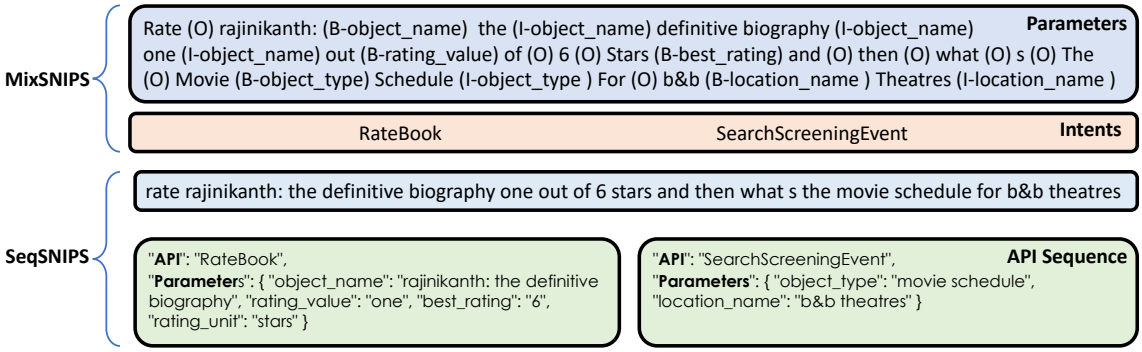

Figure 4: Example of how SeqSNIPS is created. Using a natural language utterance from MixSNIPS and the flat list of slots, we convert it into a sequence of API calls, each with a dictionary of parameter names and values.

idation, and 1k test samples with an average API count of 2.36 and an average parameters count per API of 3.67.

3.2 Grammar-Based Generation

SeqATIS and SeqSNIPS: ATIS (Hemphill et al., 1990) is a collection of human-annotated queries about flights. Queries ask for information such as flight numbers, airports, dates, and other relevant details. The dataset also provides a range of semantic labels, including intent and slot values. Intents are the overall goals of the queries, such as "flight query" or "airfare query" while slot values are the specific pieces of information that are being requested, such as "departure city" or "arrival time". SNIPS (Coucke et al., 2018) is another dataset focused on voice assistants. It consists of human-annotated queries that cover various domains such as weather, music, and calendar.

MixATIS and MixSNIPS are multi-intent datasets (Qin et al., 2020) built based on ATIS and SNIPS, respectively. It was created by collecting sentences from the ATIS/SNIPS dataset and connecting them with conjunctions, such as "and". The resulting data had sentences with 1-3 intents at a probability of 30%, 50%, and 20%, respectively.

One issue with using these two datasets for API calling is that they do not indicate which parameters should be associated with which API. For example, as Figure 4 shows, the original MixSNIPS dataset only evaluates model's ability to detect the two gold intents and which segments of the text are the target slots. To convert MixATIS and MixSNIPS datasets to a sequence of API calls, we divided utterances back to their original single intent utterances to get the corresponding parameters for each API. We then parsed its IOB (Inside/Outside/Beginning) parameter notations to generate the list of API parameter names and values. Now, we merge the utterances back along with the APIs and their parameters to get the sequence of API calls. In this way, we have curated SeqATIS and SeqSNIPS from MixATIS and MixSNIPS, respectively. In SeqATIS, we have around 11.5k train, 700 validation, and 800 test examples having an average API sequence length of 2.13 and an average parameter count per API of 4.85. Whereas, SeqSNIPS consists of around 40k train, 2.2k vali-

dation, and 2.2k test samples with an average API sequence length of 1.96 and an average parameters count per API of 5.06.

SeqToolQA: ToolQA (Zhuang et al., 2023) is an open-source dataset designed for tool-augmented LLMs evaluation. The dataset tries to address the issue with current tool-based evaluation methods which do not distinguish between questions that can be answered using LLMs' internal knowledge and those that require external information through tool use. To do so, the datasets come with 8 independant datasets (e.g. Kaggle Flights, Coffee, Yelp, Airbnb, etc.) and a set of questions that can be answered only by querying those datasets, hence ruling out the internal knowledge of LLMs. The dataset is provided as a set of template-based questions with their final answers.

However, ToolQA does not provide the intermediate set of steps (tools) that need to be executed to generate the answer to the given questions. The listing below shows some examples:

```
{
 "qid": "easy-flight-0003",
 "question": "What was the departure time of the
      DL891 flight from SEA to LAX on 2022-01-22?",
 "answer": "11:54"
}
```

Too address this issue, we propose SeqToolQA where we used the provided implementation of ToolQA on how each template question is answered and transformed into a corresponding set of APIs. We abstracted 17 different APIs covering 6 domains and created a total of 358 examples for testing purposes. We show an example below:

```
{
 "qid": "easy-flight-0000",
 "question": "What was the departure time of the UA5
     480 flight from ORD to HSV on 2022-07-06?",
 "apis": [
    "LoadDB[DBName=flights]",
    "FilterDB[Origin=ORD,  Dest= HSV, FlightDate= 20
         22-07-06, Flight_Number_Marketing_Airline=5
         480, IATA_Code_Marketing_Airline=UA]",
    "GetValue[ValueName=DepTime]"
    ],
 "answer": "18:11"
}
}
```

SeqTopV2: Topv2 (Chen et al., 2020) is a multi-domain task-oriented semantic parsing dataset comprising examples from eight domains; alarm, event, messaging, music, navigation, reminder, timer, and weather. The total dataset consists of 180k samples, randomly divided into training, development, and test sets for each domain. We followed a straightforward approach to convert this dataset into APIs using its intents "IN:" and slot "SL:" notations. Note that this dataset genuinely has a sequence of APIs that has to be followed. In the example "Remind me to email Joy about the details with the new client tonight", we transform it into two APIs; "SEND_MESSAGE" and "CREATE_REMINDER" where "CREATE_REMINDER" has prerequisite for "SEND_MESSAGE".

The original dataset had a lot of "UNSUPPORTED" notations for examples where there is no matching intent. We excluded these cases from our API dataset. Along with them, we also removed the samples that had duplicate utterances, ambiguous intents, and analogous slot names inside the same intent. We call the resulting dataset SeqTopV2, and it has 94K, 13.5K, and 6K for training, development, and testing splits, respectively.

3.3 Off-the-Shelf Usage

ToolBench: This is a subset of ToolBench (Xu et al., 2023b) focused on two domains, HomeSearch and Booking. We did not do any transformation to these datasets and rather used it "as-is", since they are already in API form.

ToolLLM (Qin et al., 2023) proposes an instruction-tuning tools dataset and a tool-augmented LLM model based on LLaMA-2-7B. The dataset is created synthetically based on ChatGPT and a collection of 16K APIs from RapidAPI. The dataset includes three subsets; G1,G2, G3 which refer to single-tool, intra-category multi-tool and intra-collection multi-tool data, respectively. We used those three subsets above as-is for out-of-distribution model evaluation.

API Bank (Li et al., 2023b): is a benchmark designed for evaluating tool-augmented LLMs. It includes 314 tool-use dialogues with 753 API calls to assess the existing LLMs' capabilities in planning, retrieving, and calling APIs. It also comes with a larger training dataset of 1.8K dialogues and a model trained on it (Lynx) initialized from Alpaca. In this paper, we use the the test sets of API Bank for out-of-distribution evaluation. Since this is a dialoge with tools being called in between, we divided each dialogue into multiple test examples where each example include a conversation and a sequence of APIs needed thus far.

ToolAlpaca (Tang et al., 2023a): ToolAlpaca is a training and evaluation benchmark that is automatically curated through a multi-agent simulation environment using ChatGPT and GPT-3.5. The corpus contains 3,938 tool-use instances from more than 400 real-world tool APIs spanning 50 distinct

Datasets	Data Quality (Avg)			Annotator Agreement (Avg)		
	Intent	Sequence (Intent)	Slot	Intent	Sequence (Intent)	Slot
SeqATIS	0.96	0.96	0.89	1.00	1.00	0.90
SeqSGD	0.98	0.98	0.85	1.00	1.00	0.94
SeqSNIPS	0.93	0.94	0.89	0.98	0.96	0.94
SeqMultiWOZ	1.00	1.00	0.97	1.00	1.00	0.98
SeqTopV2	0.97	0.97	0.89	0.98	0.98	0.90

Table 2: Data Quality with Inter-Annotator Agreement scores for API-BLEND In-Distribution Datasets.

categories. Although it comes with a training set, in this paper, we have only used the test set for out-of-distribution evaluation.

4 Data Quality Assessment

To check the quality of API-BLEND datasets, we perform human annotation over 50 randomly sampled data from 5 in-distribution datasets (at least 2 annotators per dataset). Table 2 demonstrates the data quality and annotator agreement scores. The following steps are taken to measure the data quality: first, the annotators score each gold sample (1 if correct, 0 otherwise) based on the following 3 metrics: (1) "*Intent*": whether the intents/APIs in the output are correct to answer the query; (2) "*Sequence(Intent)*": whether the intents are in the proper sequence; and (3) "*Slot*": whether the slot names and slot-values are correct. Then, we calculate the average scores across all the annotators and present them under the "*Data Quality*" section in table 2. Furthermore, we assess the annotator agreement by examining each of the three segments: (1) *Intent*, (2) *Sequence(Intent)*, and (3) *Slot*, to determine whether there is a unanimous agreement among annotators (assigning a value of 1 for agreement and 0 otherwise). Subsequently, we calculate the average agreement across all these segments. As the API-BLEND data suite has been generated from high-quality datasets from different domains, we are seeing very high scores in our data quality assessment, particularly for the easier "*Intent*" and "*Sequence(Intent)*" tasks.

5 Experiments and Results

5.1 Baselines

In our experiments, we have used 9 open sourced models as baselines: (1) LLaMA-2-70B (Touvron et al., 2023), (2) Falcon-180B (Almazrouei et al., 2023), (3) LLAMA-2-7B (Touvron et al., 2023), (4) FLAN-T5-XXL (Chung et al., 2022), (5) Falcon-40B (Almazrouei et al., 2023), (6) StarCoder-15B (Li et al., 2023c), (7) MPT-30B (Team et al., 2023), (8) ToolLLaMA-2-7B (Qin et al., 2023), and (9) Lynx-7B (Li et al., 2023b). We tested these models in three settings; (1) few shot testing: we evaluated LLAMA-2-70B, Falcon-180B, and ToolLLaMA-2-7B in a 3 shot mode; (2) Instruction fine-tuning on target dataset: we consider this setting for FLAN-T5-XXL, StarCoder-15B, Falcon-40B, and MPT-30B; and (3) Instruction fine-tuning on combined datasets: we evaluated this setting for all models in (2) along with LLaMA-2-7B to evaluate whether we can get a single model trained on the plurality of all datasets and still perform well on each individual test set. For the OOD experiments, we have used all the fine-tuned models from (3) in conjunction with the ToolLLaMA-2-7B and Lynx-7B which are already fine-tuned with the ToolLLM and APIBench data, respectively.

5.2 Instruction Tuning

In all experiments, we have used the same instructions for training and testing. We show below the instruction template. Only when evaluating non-fine-tuned models or for the OOD experiments, we also provide 3 ICL examples via the part "Here are some examples: ICL_EXAMPLES" in the instruction) and remove it otherwise.

```
### Instruction Template with ICL Examples ###
Given the APIs and Slots below, sequence them in the
      order in which they have to be called to
    answer the following query.
Possible APIs: {INTENT_LIST}
Possible Slots: {SLOT_LIST}
Here are some examples: {ICL_EXAMPLES}
Query: {QUERY}
Answer:
```

5.3 Settings and Parameters:

We used QLoRA (Dettmers et al., 2023) to fine-tune all our models. While fine-tuning the models on targeted datasets, we made sure that the model saw 100k samples in the training process. In combined data training, we fine-tuned the models for 2 epochs over the cumulated datasets. In both cases, the batch size was 1 with gradient accumulation steps of 8 and a learning rate of $5e^{-5}$.

5.4 Metrics

To perform a fine-grained evaluation of the generated responses, we use two kinds of evaluation - standard information retrieval metrics (precision, recall, and F1 scores) and Longest Common Subsequence (LCS). We report F1 APIs and F1 slots/-Parameters to compute the F1 scores by comparing the predicted APIs with the gold ones and the pre-

FT Types	Models	SeqATIS	SeqSNIPS	SeqSGD	SeqMultiWOZ	SeqTopV2	Weighted Avg.
No FT	**Falcon-180B**	0.15 | 0.02 | 0.15	0.39 | 0.07 | 0.40	0.21 | 0.06 | 0.21	0.44 | 0.25 | 0.45	0.08 | 0.00 | 0.09	0.19 | 0.04 | 0.20
	LLaMA-2-70B	0.10 | 0.01 | 0.11	0.26 | 0.03 | 0.27	0.10 | 0.02 | 0.10	0.23 | 0.12 | 0.25	0.04 | 0.00 | 0.04	0.11 | 0.02 | 0.12
	ToolLLaMA-2-7B	0.29 | 0.03 | 0.32	0.49 | 0.04 | 0.47	0.43 | 0.05 | 0.45	0.78 | 0.40 | 0.79	0.07 | 0.00 | 0.08	0.27 | 0.05 | 0.28
FT w. dataset mentioned in the column	**FLAN-T5-XXL**	0.92 | 0.72 | 0.92	0.97 | 0.90 | 0.97	0.98 | 0.69 | 0.98	1.00 | 0.99 | 1.00	0.96 | 0.83 | 0.96	0.97 | 0.83 | 0.97
	StarCoder-15B	0.99 | 0.84 | 0.99	0.96 | 0.87 | 0.96	0.98 | 0.67 | 0.98	1.00 | 0.99 | 1.00	0.95 | 0.78 | 0.95	0.96 | 0.80 | 0.96
	Facon-40B	0.92 | 0.70 | 0.92	0.97 | 0.89 | 0.97	0.96 | 0.62 | 0.96	1.00 | 0.97 | 1.00	0.90 | 0.56 | 0.91	0.93 | 0.67 | 0.94
	MPT-30b	0.96 | 0.81 | 0.96	0.97 | 0.90 | 0.97	0.98 | 0.68 | 0.98	1.00 | 0.98 | 1.00	0.96 | 0.84 | 0.97	**0.97 | 0.84 | 0.97**
FT w. all data	**FLAN-T5-XXL**	0.94 | 0.72 | 0.94	0.96 | 0.89 | 0.97	0.98 | 0.70 | 0.98	1.00 | 0.97 | 1.00	0.97 | 0.87 | 0.97	0.97 | 0.85 | 0.97
	StarCoder-15B	0.98 | 0.81 | 0.98	0.96 | 0.86 | 0.96	0.98 | 0.67 | 0.98	1.00 | 0.96 | 1.00	0.96 | 0.83 | 0.96	0.97 | 0.82 | 0.97
	LLaMA-2-7B	0.92 | 0.70 | 0.92	0.96 | 0.88 | 0.97	0.97 | 0.65 | 0.97	1.00 | 0.97 | 1.00	0.96 | 0.85 | 0.97	0.96 | 0.83 | 0.97
	Facon-40B	0.90 | 0.67 | 0.90	0.96 | 0.87 | 0.97	0.94 | 0.62 | 0.94	1.00 | 0.94 | 1.00	0.93 | 0.66 | 0.93	0.94 | 0.72 | 0.94
	MPT-30B	0.94 | 0.77 | 0.94	0.97 | 0.90 | 0.97	0.98 | 0.70 | 0.98	1.00 | 0.97 | 1.00	0.97 | 0.87 | 0.97	**0.97 | 0.85 | 0.97**

Table 3: Evaluation Results on **In-Distribution** datasets. Each scores are shown in the following format: **API-F1 | Parameter-F1 | LCS-F1**. The weighted average scores are calculated using the number of test samples in Table 1.

dicted parameters of each API with its gold counterparts. The rationale for using standard metrics such as precision, recall, and F1 scores is that they emphasize exact matches of API and slot names. This decision is based on the fact that APIs are highly specific, and successful execution requires every detail (e.g., name, parameters, input/output format) to align perfectly with the API descriptions. Similarly, to evaluate the model's ability to adhere to the *sequence* of API calls necessary for responding to the given natural language query, we compute the LCS metric to capture the overlap between the gold and predicted sequences of APIs. For each utterance in the test set, first, we identify the longest common subsequence (LCS) between the sequence of APIs in the ground truth and the model's output sequence, and then calculate sequence-level precision and recall. Precision is the length of the LCS divided by the total number of APIs in the model's output sequence, while recall is the length of the LCS divided by the total number of APIs in the ground truth sequence. We then calculate average precision (LCS-Precision) and recall (LCS-Recall) across all sequences, using these values to compute an average F1-score (LCS-F1).

5.5 In-Distribution Evaluation Results

No Fine-tuning evaluation: The first experiment we did was to check how the state-of-the-art open LLMs perform in such a setting. In particular, we evaluated LLaMA-2-70B and Falcon-180B using 3-shot prompting. We also considered ToolLLaMA-2-7B (Qin et al., 2023); a LLAMA-2-7B based model trained on API datasets generated using ChatGPT based on specifications from RapidAPIs. Table 3 shows the evaluation results on five in-distribution datasets: SeqATIS, SeqSNIPS, SeqSGD, SeqMultiWoz, and SeqTopV2. On all datasets, all three non-fine-tuned models seem to get some of the APIs correctly but fail to get the parameters to call such APIs.

Fine-tuning on One Dataset: In this experiment, we fine-tune the baselines discussed above on each dataset and test it on the corresponding test split. We have evaluated four models here: FLAN-T5-XXL, StarCoder-15B, Falcon-40B, and MPT-30B. Ideally, this setting should give the best performance since the training data is focused towards one dataset and its APIs. As shown in Table 3, all models achieved very good performance ($> 90\%$) detecting the right APIs with the performance of all models reaching 100% API-F1 scores for SeqMultiWoz dataset. We hypothesize that this high performance is because the number of APIs in most datasets is not very large (e.g., 12 APIs for SeqMultiWoz). Detecting the correct set of parameter names and values is a more challenging problem for most models with performance being the lowest for the SeqSGD dataset. The weighted average number suggests that the MPT-30B model is doing slightly better than the other models.

Fine-tuning on All Training Datasets: In this setting, we combine all training datasets and fine-tune the five baseline models on them. The goal of this experiment is to check if we can get a generic model that works well when tested on each individual dataset. We see in Table 3, on SeqATIS and SeqMultiWoz, models trained on the combined training data achieve lower performance compared to models trained on the individual dataset. Performance on SeqSNIPS was similar for both models, while models trained on the combined data achieved better performance on SeqSGD and SeqTopV2. The average scores suggest that all models achieved better performance when trained on the combined datasets compared with the single dataset

Datasets	Fine-Tuned with all API-BLEND data					Tool-Augmented LLMs	
	Falcon-40B	FLAN-T5-XXL	MPT-30B	LlaMA-2-7B	StarCoder-15B	ToolLLaMA-2-7B	Lynx-7B
ToolLLM-G1	0.48 | 0.47	-	0.11 | 0.11	0.07 | 0.07	0.32 | 0.32	0.12 | 0.12	0.43 | 0.44
ToolLLM-G2	0.48 | 0.47	-	0.24 | 0.23	0.09 | 0.08	0.53 | 0.53	0.01 | 0.01	0.33 | 0.34
ToolLLM-G3	0.50 | 0.49	-	0.51 | 0.49	0.19 | 0.20	0.49 | 0.48	0.16 | 0.16	0.50 | 0.50
API Bank-1	0.42 | 0.15 | 0.44	0.57 | 0.12 | 0.59	0.59 | 0.21 | 0.62	0.52 | 0.16 | 0.55	0.49 | 0.15 | 0.55	0.11 | 0.04 | 0.12	0.31 | 0.11 | 0.33
API Bank-2	0.37 | 0.16 | 0.39	0.51 | 0.11 | 0.53	0.49 | 0.20 | 0.52	0.38 | 0.14 | 0.40	0.45 | 0.13 | 0.48	0.05 | 0.03 | 0.05	0.19 | 0.09 | 0.20
ToolBench-HS	0.95 | 0.77 | 0.92	0.42 | 0.16 | 0.43	0.91 | 0.76 | 0.80	0.69 | 0.41 | 0.54	0.90 | 0.81 | 0.89	0.59 | 0.35 | 0.56	0.77 | 0.55 | 0.76
ToolBench-B	0.89 | 0.76 | 0.76	0.36 | 0.09 | 0.33	0.85 | 0.72 | 0.78	0.76 | 0.49 | 0.56	0.44 | 0.34 | 0.40	0.73 | 0.52 | 0.63	0.57 | 0.47 | 0.48
SeqToolQA	0.27 | 0.02 | 0.28	0.18 | 0.00 | 0.19	0.48 | 0.02 | 0.51	0.50 | 0.00 | 0.53	0.24 | 0.01 | 0.26	0.14 | 0.00 | 0.14	0.27 | 0.02 | 0.29
ToolAlpaca	0.41 | 0.11 | 0.41	0.54 | 0.13 | 0.55	0.51 | 0.12 | 0.52	0.46 | 0.13 | 0.46	0.47 | 0.13 | 0.49	0.18 | 0.02 | 0.20	0.32 | 0.04 | 0.34
Weighted Avg.	0.47 | 0.21 | 0.46	0.42 | 0.09 | 0.43	**0.49 | 0.23 | 0.49**	0.41 | 0.16 | 0.40	0.44 | 0.17 | 0.46	0.18 | 0.09 | 0.18	0.37 | 0.14 | 0.38

Table 4: Evaluation Results on **Out-of-Distribution (OOD)** dataset. Each scores are shown in the following format: **API-F1 | Parameter-F1 | LCS-F1**, except the ToolLLM datasets, which are API-only, so they do not have the Parameter-F1 score. All models are prompted with 3-shot examples. The weighted average scores are calculated using the number of test samples in Table 1.

training.

5.6 Out-Of-Distribution Evaluation

To check the generalizability of the different API models, we measure the performance of all models on five out-of-distribution test sets; ToolLLM, API Bank, ToolBench, our SeqToolQA, and ToolAlpaca. Some test sets have subcategories, such as ToolLLM has G1, G2, and G3; API-Bank has L1 and L2; and ToolBench has HS and B for Home Search and Booking, respectively. In our test-suite, ToolLLM is the API-only test-set that does not have any parameters. In this experiment, we use the 5 models (i.e., FLAN-T5-XXL, StarCoder-15B, Falcon-40B, MPT-30B, and LLaMA2-7B) that are fine-tuned with our combined data. In addition to these models, we have also evaluated ToolLLaMA-2-7B from ToolLLM (Qin et al., 2023) and Lynx-7B from API-Bank (Li et al., 2023b). In this experiment, we compare the performance with 3-shot in-context learning examples for all the models.

Table 4 showcases our OOD evaluation results. Our models that are fine-tuned with API-BLEND data perform better than other Tool/API augmented models. This is because our models achieve better generalizability as they have seen diverse sets of API data from different domains in the fine-tuning process. Falcon-40B and StarCoder-15B are showing better performance on our API-only test-set ToolLLM (we did not evaluate FLAN-T5-XXL on ToolLLM due to max sequence limit), whereas FLAN-T5-XXL and MPT-30B are doing well on API-Bank and ToolAlpaca. Even though ToolLLaMA-2-7B and Lynx-7B are from ToolLLM and API-Bank respectively, still they are performing poorly. In their papers, they used different metrics, e.g., pass rate to determine how many times the model reached an answer, and win rate which uses ChatGPT as a judge. In both the ToolBench datasets, Falcon-40B is outperforming the others. On SeqToolQA, even though the models have scored some points in API detection, however, all the models performed poorly on detecting the parameter names and values, which leads to a low Parameter-F1 score. This is because the parameter values in SeqToolQA contain codes, SQL, math functions, etc., which models have not seen in any training data, and these special slot values are not trivial to generate by seeing only a few ICL examples.

5.7 Qualitative Studies

We also performed extensive studies on the outputs generated by the models in our experiments. In this section, we are going to discuss our findings on the failed cases along with some common mistakes demonstrated by the models. We found, in most of the cases, that parameters names and values are the obvious reason for not doing well on slot detection in both in and out-of-distribution test sets. We provide samples for each case for better understanding. We would like to mention that most of the provided examples contain "gold_output" and "predicted_output" and we have shown only the error part (sub-string) from the original outputs for brevity.

5.7.1 Unnormalized Slot-Values

In an ideal scenario, the parameter values should be extracted exactly from the query by the models while generating the texts. However, sometimes, the model extracts the sub-part of it or represents it in a different form after extracting it. In a human evaluation, we would consider the generated text matches the gold, although while doing it programmatically it's showing a mismatch and we have

not found a good way to normalize the parameter values. The following examples capture some of the unnormalized parameter value mismatches. In the first example, the month and the day in the predicted output are repeated. The predicted output on the second one contains only the city name, whereas the gold contains the city and the state. In the final example, even if the intent and slot values are correct, they have used different parameter formats to represent it. We plan to investigate further these issues, but we keep it for future work.

```
### SeqSGD ###
{  "gold": "March 3rd, this Sunday",
   "predicted": "March 3rd, the 3rd"
}
{  "gold": "NYC, New York",
   "predicted": "NYC"
}
### ToolBench ###
{  "gold": "Date(3, 9, 2023)",
   "predicted": "Date(year=2023, month=3, day=9)"
}
```

5.7.2 Semantically similar slot-names in API Specification

In our instructions, we provide the possible list of APIs and parameters to be used by the model while answering the queries. We extract these APIs and parameters from the original dataset's API specifications, if any. However, we found in some cases the parameter names are semantically very similar across the datasets. Here are some examples from the SeqSGD dataset: (1) *leaving_date* and *departure_date*; (2) *leaving_time* and *departure_time*; (3) *origin*, *from_city*, and *from_location*; and (4) *destination*, *to_city*, and *to_location*. Now, it often happens that the parameter values are correct in the generated text but the parameter names do not exactly match with the gold, even if they are very close. Following are some examples of such cases.

```
### SeqSGD ###
{ "gold": "destination_airport = ATL",
  "predicted": "destination = ATL"
}
{ "gold": "show_type = imax",
  "predicted": "theater_name = imax"
}
### SeqATIS ###
{  "gold": "cuisine = souvlaki",
   "predicted": "served_dish = souvlaki"
}
```

6 Conclusion

This paper presents API-BLEND, a large corpora for training and evaluation of tool-augmented LLMs, curated from real-world datasets of different domains such as dialog, semantic parsing, digital assistants, etc. API-BLEND consists of 10 datasets (5 in-distribution and 5 out-of-distribution) comprising over 190k instances (including train, development, and test). We have demonstrated that the models fine-tuned with API-BLEND generalize better than the other tool-augmented LLMs on OOD experiments. Our findings not only substantiate the importance of API-BLEND in training and benchmarking tool-augmented LLMs but also highlight the necessity of generalizability to improve the API usage capability of LLMs.

7 Limitations and Risks

A limitation of our benchmark, API-BLEND, is that it does not deal with environment interactions for an API agent. In future work, it will be interesting to explore this setting of an embodied agent, where the API calls effect changes in the grounded environment. Further, our benchmark is focused on English API commands, and in the future, it will be interesting to develop a multilingual API-BLEND. Also, in a real-world scenario, for a query, multiple correct solutions with different sequences of APIs are possible, however, the API-Blend dataset does not have such cases. For the multiple correct solutions, we can evaluate the predicted solution using one of the following approaches: (i) if the ground truth consists of all the alternate API sequences, then we can use our existing metrics by comparing the predicted sequence with each alternate ground truth; (ii) if the APIs are executable, then we can use accuracy-based metrics over the final model response, and (iii) otherwise we can leverage an LLM evaluator (as a judge) to calculate the win/pass rate (Qin et al., 2023). We do not perceive any risks associated with our work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Nicholas Farn and Richard Shin. 2023. Tooltalk: Evaluating tool-usage in a conversational setting. *arXiv preprint arXiv:2311.10775*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Himanshu Gupta, Kevin Scaria, Ujjwala Anantheswaran, Shreyas Verma, Mihir Parmar, Saurabh Arjun Sawant, Swaroop Mishra, and Chitta Baral. 2023. Targen: Targeted data generation with large language models. *arXiv preprint arXiv:2310.17876*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *arXiv preprint arXiv:2305.11554*.

Joy He-Yueya, Gabriel Poesia, Rose E Wang, and Noah D Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. *arXiv preprint arXiv:2304.09102*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2022. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8460–8478.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. Mlqa: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330.

Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023a. Api-bank: A benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. Api-bank: A benchmark for tool-augmented llms.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023c. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, pages 2080–2094, Online. Association for Computational Linguistics.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020. Agif: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1807–1816.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Tanik Saikh, Tirthankar Ghosal, Amish Mittal, Asif Ekbal, and Pushpak Bhattacharyya. 2022. Scienceqa: A novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23(3):289–301.

Scarton Scarton, Mikel L Forcada, Miquel Esplà-Gomis, and Lucia Specia. 2019. Estimating post-editing effort: a study on human judgements, task-based and reference-based metrics of mt quality. In *Proceedings of the 16th International Conference on Spoken Language Translation*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *ArXiv*, abs/2302.04761.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023a. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023b. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

MN Team et al. 2023. Introducing mpt-7b: a new standard for open-source, commercially usable llms.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models, 2023. *URL https://arxiv.org/abs/2307.09288*.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023a. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023b. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2021. Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. *arXiv preprint arXiv:2104.00773*.

Yu Yu, Shahram Khadivi, and Jia Xu. 2022. Can data diversity enhance learning generalization? In *Proceedings of the 29th international conference on computational linguistics*, pages 4933–4945.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted

gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *arXiv preprint arXiv:2306.13304*.