Teaching Models new APIs Quickly: Domain Agnostic Simulators for Task Oriented Dialogue

Anonymous ACL submission

Abstract

We investigate the ability of large language models to neurally generate Task Oriented Dialogues in novel domains, provided only with an API implementation and a list of goals.

We show these simulations formulate online, automatic metrics that correlate well with human evaluations. Furthermore, by filtering for dialogues where goals are met, we can use simulation to repeatedly generate training data and improve the quality of the dialogues themselves. With no human intervention or domain-specific training data, our simulations bootstrap end-to-end models which achieve a 37% error reduction over baseline in previously unseen domains. By including as few as 32 domain-specific conversations bootstrapped models can match the performance of a fully-supervised model with $10\times$ more data.

1 Introduction

Virtual Assistants have become ubiquitous in modern life (Acharya et al., 2021). However, building these Task Oriented Dialogue (TOD) systems is laborious, requiring significant data collection and engineering resources to add support for a novel domain. As such, methods which can generalize, learn from limited examples, and require fewer engineering resources are highly desirable (Shi et al., 2019; Shah et al., 2018; Acharya et al., 2021).

To this end, works have previously identified User Simulators, wherein a model is used to emulate a human user in place of a real one, as a means of addressing these problems. User Simulators have been used to evaluate (Walker et al., 1997, 2000; Schatzmann et al., 2005) and improve Assistant models by providing additional training data (Shah et al., 2018; Acharya et al., 2021) and reward signals for Reinforcement Learning methods (Fazel-Zarandi et al., 2017; Su et al., 2018; Shi et al., 2019). Typically, these User Simulators are either limited to enhancing existing domains (Fazel-Zarandi et al., 2017) or utilize specialized and manually engineered rules or templates for novel domains (Shah et al., 2018; Shi et al., 2019). User Simulators have often required post-hoc human intervention to ensure quality (Shah et al., 2018).

In this work, we show that modern Large Language Models (Radford et al., 2018, 2019; Lewis et al., 2020) generate reasonable dialogues when equipped with an API implementation and prompted with a goal. We observe the quality of these dialogues increases with the power of the base models. Furthermore, we observe that simulation success is a strong discriminator of Assistant performance and dialogue quality.

We describe a method for bootstrapping User and Assistant models for previously unseen dialogue domains. We use Task Success, which can be automatically measured in fully synthetic dialogues, to discriminate between high- and low-quality dialogues. By adding successful dialogues back into the training set and retraining the model, we bootstrap an Assistant model without the use of any domain-specific training data, hand-engineered rules, Natural Language templates, or humans-in-the-loop. Our methodology shows improvements in both zero-shot and full-shot settings.

Furthermore, we show that we can use Task Success as a method for automatically identifying the weakest areas of our model, and employ Active Learning (Tur et al., 2003; Olsson, 2009) to enhance performance. By additionally including as few as 32 domain-specific training examples, we can match the performance of a fully-supervised baseline provided with $10\times$ more data.

We open source our simulation infrastructure — including processing for all public datasets used and scripts for both training models and generating bootstrapped conversations — as part of the Anonymous framework (Anon, 202X).[1]

[1] Code already open sourced online at: Anonymous URL

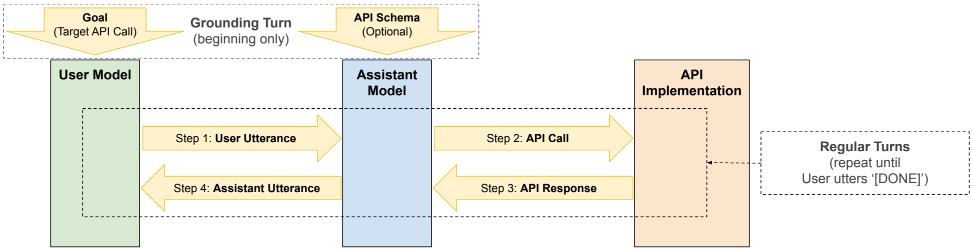

Figure 1: Illustration of our Simulation system. See Section 2 for a description of functionality. In-arrows designate inputs to an entity; entities do not see data where there is no in-arrow. Out-arrows designates generations.

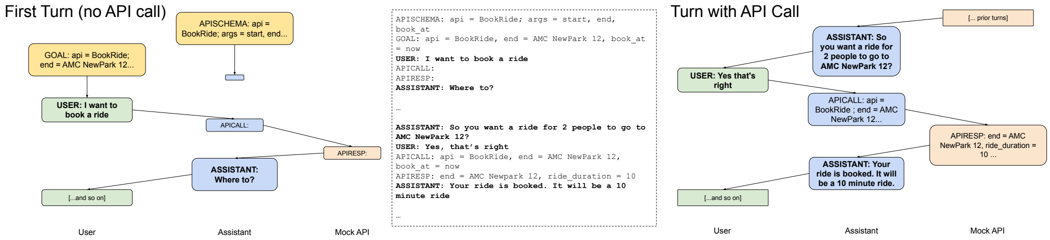

Figure 2: Concrete example illustrating the system described in Fig 1. A linear version of the conversation is in the middle; **bold** denotes **utterances**. Diagrams on either side illustrate how generations are passed between entities.

2 End-to-End TOD Conversation Setup

A high-level illustration of our simulation system is shown in Figure 1 with an example in Figure 2. Our simulation system consists of three main components: a *User* model, an *Assistant* model, and an *API Implementation*. While traditional TOD systems model conversations with a combination of intent detection, belief state tracking, and policy (Jurafsky and Martin, 2009), we employ a more modern setup (Rastogi et al., 2019) where the Assistant must both generate API calls and translate API responses into Natural Language utterances for the User at the right time. This is particularly amenable to modern End-to-End (E2E) approaches based on pretrained Language Models (Ham et al., 2020; Peng et al., 2020; Hosseini-Asl et al., 2020).

To guide the conversation, the User is given a *Goal* as its first turn. The Goal consists of a complete API call (e.g. intent, slot names, and slot values) serialized as a string. The User model uses this Goal to ground natural language utterance generation to the Assistant. The Assistant optionally generates a serialized API call string that is sent to the API Implementation. The API Implementation returns a serialized API response back to the Assistant based on the call, including a sentinel value for failed calls. The Assistant generates a natural language utterance to the User with this response. Entities keep track of their own generations as well as previously seen turns when making generations.

Conversations continue in this repeated fashion until the User generates a '[DONE]' token. A conversation is said to be *successful* if the Assistant generates an API call equal to the Goal given to the User. We later show that simulations' *Task Success Rate* (TSR), the success averaged over a large number of goals, is a strong proxy for the quality of dialogues generated.

Our system optionally allows grounding the Assistant with an *API Schema* on the first turn; we use this in Sec 5. An API Schema consists of the signature of the Goal (e.g. intent and slot names) without slot values. As we will later see, Schemas (combined with Task Success) enable us to bootstrap models in unseen domains. We take care to *not* use Schemas for evaluations unless explicitly stated, however, since this implies a precursor intent detection step to select the Schema.

3 Related Work

User Simulators have a long history. They have been used for both evaluation and Assistant improvement. Formulations have varied across Rule-based, Agenda-based, and End-to-End approaches.

Works have explored using User Simulators as a proxy for Assistant evaluation (Schatzmann et al., 2005) or predicting user satisfaction (Walker et al., 2000; Ai and Weng, 2008; Jung et al., 2009; Li et al., 2016; Crook and Marin, 2017). Simulators have been measured via Task Success Rate (Gür et al., 2018; Kreyssig et al., 2018) and cross-examining them against a wide-variety of Assistants (Schatzmann et al., 2005). Older works explored using Language Models (LMs) as User Simulators, but observed that the LMs had poor adherence to goals (Georgila et al., 2006; Crook and Marin, 2017). Accordingly, simulators of that time were often Agenda-based (Schatzmann et al., 2007; Shah et al., 2018). In contrast, we find that modern, large, pretrained LMs ground on goals well.

Works have also used User Simulators and Assistants to optimize Reinforcement Learning policy (Schatzmann et al., 2007; Fazel-Zarandi et al., 2017; Peng et al., 2017; Su et al., 2018; Gür et al., 2018; Kreyssig et al., 2018). These often optimized the policy component of pipeline-based systems (Fazel-Zarandi et al., 2017) and frequently relied on the Natural Language Generation (NLG) templates over dialogue acts (Fazel-Zarandi et al., 2017; Shah et al., 2018; Shi et al., 2019; Kreyssig et al., 2018; Acharya et al., 2021). Our work instead utilizes fully lexicalized, E2E models for both the User and the Assistant models, without the need for agendas, dialogue acts, or NLG templates.

Some works use Schemas, neural User Simulators, and neural Assistants for generating synthetic data to add to a training dataset (Shah et al., 2018; Acharya et al., 2021; Campagna et al., 2020; Kim et al., 2021; Mohapatra et al., 2021); these are most similar to our bootstrapping. Shah et al. (2018) uses a stage of human annotation to identify failed dialogues and paraphrases; we show that Task Success can successfully identify failures without human involvement. Campagna et al. (2020) uses detailed schema and rule-based templates for NLG; our schemas are much simpler and we use no rules. Acharya et al. (2021) few-shots using a similar bootstrapping process as ours, but relies on templates for NLG and human paraphrases; we use sampling to achieve diversity (Holtzman et al., 2020). The papers Kim et al. (2021); Mohapatra et al. (2021) use pre-trained language models to generate synthetic dialogues given schemas. Mohapatra et al. (2021) uses a complex set of models to generate each side of the conversation, then filter and label; they also do not attempt to solve the zero-shot learning task. Kim et al. (2021) use a BART-large model to generate dialogues given goal instructions and API results, and then RoBERTa to label the generated dialogues with state tracking labels. They use a pre-generated list of API results for fine-grain direction of the generated conversation. In comparison our dialog uses only goals for directional grounding. We additionally automatically self-improve our generation in a loop using the models generated, successful, dialogues.

In contrast to prior methods that require human selection, template generation, or paraphrasing, we both identify failures and generate conversational variety automatically.

End-to-End systems for TOD have had a recent surge in interest (Asri et al., 2016; Bordes et al., 2016; Liu and Lane, 2018; Rastogi et al., 2019; Ham et al., 2020; Hosseini-Asl et al., 2020; Peng et al., 2020; Lin et al., 2020), thanks to the success of pretrained models (Devlin et al., 2019; Radford et al., 2019; Lewis et al., 2020). E2E models promise to lower the cost of annotation by replacing traditional pipeline models with text-in-text-out and adjacent external API calls (Rastogi et al., 2019; Byrne et al., 2021). Compared to these works, we leverage pretrained models for User models in order to produce simulations, rather than only modeling Assistants. We also leverage Schema-based grounding techniques (Rastogi et al., 2019; Balaraman and Magnini, 2021), which may be viewed as a form of in-context prompting methods (Brown et al., 2020; Schick and Schütze, 2020; Wang et al., 2021; Wei et al., 2021).

4 Evaluating Synthetic Dialogue

We generate synthetic conversations with a variety of different models and validate that traditional automated offline metrics, our TSR metric, and human evaluation all correlate with expectations. Note that our objective is explicitly not to achieve State-of-the-Art performance but to validate directional correlation; namely, we expect more 'powerful' models to outperform weaker ones.

4.1 Experimental Setup

Dataset We focus our efforts on the Google Schema Guided Dialogue (Google SGD) dataset (Rastogi et al., 2019). Google SGD is a large TOD dataset with emphasis on zero-shot incorporation of new skills. Assistants make Service Calls (API Calls) which return responses. For these experiments, we do not use dataset-included API Schemas; we force our Assistant models to learn the underlying Schemas directly from the data.

Models We experiment with four model architectures: LSTM (Hochreiter and Schmidhuber, 1997), LSTM with Attention (Bahdanau et al., 2014), GPT2 (Radford et al., 2019), and BART (Lewis et al., 2020) with R3F (Aghajanyan et al., 2021). We expect models later in this list to be more powerful. Our GPT2 implementation closely resembles the setup of SimpleTOD (Hosseini-Asl et al., 2020), while our BART model roughly mirrors the implementation of MinTL (Lin et al., 2020).

We fine-tune on Google SGD, using the original splits from Rastogi et al. (2019). For each model type, we fine-tune separate User and Assistant models. We generate synthetic conversations as described in Sec 2. We seed conversations with single API calls extracted from a specified fold, one call per conversation, as goals. We mock API Implementations via a lookup table with fully realized API calls as keys and corresponding API responses as values; this lookup table is populated directly from the dataset. We return a sentinel failure value if the Assistant makes an invalid API Call.

Evaluation Metrics We report two metrics for the Assistant: Joint Goal Accuracy (JGA)[2] and BLEU score. We additionally evaluate our simulation quality by Task Success Rate over the goals of the Valid and Test sets. In Google SGD, the Test set contains Out-of-Domain (OOD) examples that do not appear in the Train or Valid sets. As such, considering both Valid and Test gives us some estimate of OOD performance. While we aim to ensure that the models are well trained, e.g. comparable to results in the literature, our main objective to examine the correlation between Task Success measured on synthetic data and established offline metrics.

Human Evaluation We use ACUTE-Eval (Li et al., 2019) for human evaluation. ACUTE-Eval is a pairwise evaluation in which an annotator is shown two dialogues and asked a question ("Which Assistant would you rather use yourself?") about which they prefer. As recommended by Li et al. (2019), we use a manually-curated control pair comparing an artificially repetitive dialogue with a gold dialogue from Google SGD; annotators who failed to identify the gold dialogue were removed. We select a random subset of 400 goals derived from the Test set of Google SGD and generate synthetic conversations using a fixed BART model for the User and the different model architectures for the Assistant. We only present User and Assistant utterance turns to annotators (hiding any API calls). We collect pairwise annotations between each Assistant model architecture described above, as well as the gold dialogues from the original dataset (Human). Annotators were presented with conversations with the same goal when comparing model-generated conversations. We measure the fraction of times each model architecture (or Human) was preferred in its pairwise match up, and compare all possible pairs of model architectures. The annotators were also asked to provide justification for their selections. An image of the annotator UI is included in Appendix A.1.

[2]We deviate from original Google SGD evaluation here and report JGA on the *API calls* rather than the *belief state*. An example receives a JGA score of 1 iff it generates the exact API call perfectly. Note that the majority of turns do not have API calls, so majority baseline is about 0.71.

	User	Assistant		Simul. TSR	
Model	BLEU	JGA	BLEU	Valid	Test
LSTM	.058	.777	.123	.042	.042
LSTM+Attn	.078	.833	.183	.302	.169
GPT2	.093	.869	.223	.474	.307
BART	.116	.897	.252	.583	.352

Table 1: **Automatic Metrics of varied Model Architectures** tested on the original Google SGD split. TSR increases along with offline metrics, but shows greater discrimination than offline metrics.

4.2 Results

Automatic and Qualitative Evaluation See Table 1. We find that performance of all metrics improves monotonically in the direction we expect: more modern models with better pre-training and regularization do better. Additionally, the magnitude of improvements from better modeling is more visible using either measure of TSR compared to using more traditional offline metrics.

Reading simulations seeded by goals from the Test set, we observe that LSTMs generated few unique dialogues and generally ignored goals, preferring to replace slots and intents with ones more frequently seen in training. Though it had very low TSR, LSTM utterances often declared successful task completion regardless. On the other hand, GPT2 and BART models ground strongly on given goals. In particular, we observe these stronger models are able to generate plausible dialogues even on goals from domains present only within the Test (but not Train or Valid) sets of Google SGD. This ability to generalize is tested rigorously in Sec 5.

Lose % \ Win %	L	A	G	B	H
LSTM		*.48*	.55	.57	**.80**
Attn	*.52*		**.60**	**.63**	**.83**
GPT2	.45	**.40**		.58	**.81**
BART	.43	**.37**	.42		**.75**
Human	**.20**	**.17**	**.19**	**.25**	

Figure 3: **Pairwise Human Evaluations of Simulations**. Blue entries agree with automatic metrics; red italic entries disagree. Bold numbers indicate statistical significance ($p < .05$, binomial test).

Fold	No. Diag.	No. Domains
Pretraining (train)	119,677	29
In-Domain train	13,888	16
In-Domain valid	1,966	16
In-Domain test	3,132	16
Out-of-domain train	2,303	4
Out-of-domain valid	768	4
Out-of-domain test	768	4

Table 2: **Statistics of Datasets** for pretraining and bootstrapping. In-Domain and Out-of-Domain refer to new splits; see Sec 5.1. Pretraining statistics include those of In-Domain. Out-of-Domain train fold is used to sample goals for bootstrapping, but not for pretraining.

Human Evaluation The results of our human evaluation are shown in Figure 3. We label the scores depending on whether they agree or disagree with our expectations from automatic metrics. We find only the LSTM v. LSTM-Attn pair disagrees with our expectations; all other pairwise evaluations agrees. Humans are greatly preferred over all the simulations, indicating none of our simulations are at human-level performance. However, the preference for gold data roughly decreases as the quality of the system improves.

We read training logs and reasons for human preferences. As would be expected, successfully helping the User was a frequent justification for preferring one Assistant over another; conciseness, naturalness, and brevity were also mentioned.

5 Bootstrapping Novel Domains

In this section, we consider whether neural generations can *bootstrap* models on completely novel domains. At a high level, our approach generates synthetic data, filters the synthetic dialogues using Task Success, and re-trains simulation models using the synthetic dialogue. This process is repeated for multiple iterations to form a feedback loop.

5.1 Experimental Setup

Pretraining & Data setup Following the work of Soloist (Peng et al., 2020) and TOD-BERT (Wu et al., 2020), we pretrain a BART model on a large number of Task Oriented Dialogue datasets.

As before, we use Google SGD as our primary dataset. To ensure bootstrapping is truly out of domain relative to pretraining, we build two custom splits for Google SGD. We analyze all datasets and select four holdout domains unique to Google SGD. Google SGD conversations that use the holdout domains are *Out-of-Domain* and all remaining conversations are *In-Domain*. Since Google SGD has multiple domains in conversation, some non-holdout dialogue appear as part of conversations in the Out-of-Domain split. Only In-Domain Google SGD is used for pretraining. Final statistics of pretraining and evaluation data are provided in Table 2. Full descriptions of datasets, domains, and relevant preprocessing may be found in Appendix A.2.

Bootstrapping Procedure We use a *Schema-Aware* models in order to generate synthetic training data, as we find that Schema Aware models are necessary for zero-shot and few-shot domain generalization. We use goals from the Train split of Out-of-Domain Google SGD, to generating grounding for synthetic training data. In order to increase data diversity and prevent overfitting, we use Nucleus generation (Holtzman et al., 2020; $p = 0.9$). We generate 20 synthetic conversations for a given goal and retain only successful conversations. These successful conversations make up the synthetic data that we use for fine-tuning, with 10% withheld and used for model selection for early stopping.

We fine-tune both Schema-Aware (for data generation) and Schema-Agnostic (for evaluation) versions of our models on this synthetic data. Recall that Schema-Aware models have the User intent given to them, and therefore only Schema-Agnostic models may be used for evaluation. Models are fine-tuned incrementally – the best model from the previous iteration acts as initialization for the next iteration – and synthetic data is accumulated across iterations. During early experimentation, we found that fine-tuning a single model on both User and Assistant roles generally performed better than fine-tuning separate models and use this multitask setup for all of our experiments. Additionally, we find that multitasking on In-Domain data alongside synthetic data helps prevent overfitting, and include it in all experimental conditions.

Experiments We perform multiple experimental comparisons for models with synthetic data.

In our first experiment, we compare the performance of Schema-Aware and Schema-Agnostic models. This is to demonstrate that providing Schemas boosts performance and raises the Task Success Rate, enabling generation feedback loops.

In our second experiment, we consider how simulations may be used to bootstrap models in a *Zero-shot setting*. In these experiments, we only provide Out-of-Domain information via goals and completely synthetic data. To show the improvement provided by simulations, we compare primarily against the Base pretrained model, which has never seen any Out-of-Domain information. To contextualize the result, we also provide results for a Fully-Supervised model upper baseline, which was fine-tuned directly on all available Out-of-Domain data.

We evaluate on Schema-Agnostic versions of these models, and report offline metrics of Out-of-Domain JGA and Assistant BLEU-4, as well as the online metric TSR. We also report offline In-Domain JGA to ensure the use of synthetic data does not harm prior learned domains.

In our third experiment, we consider how simulations enable a form of *Active Learning*. At each iteration, we evaluate performance of the model across all conversation goals, and identify the 8 schemas with the lowest overall performance. We select 8 conversations from the Out-of-Domain training set with goals matching these schemes and add them into the next iteration's training data (along with synthetic data). We also add 8 samples to the validation data. This may be seen as *Active Learning*, where model performance is used to guide data collection at a lower cost. As a comparison, we evaluate a baseline trained with an equal number of *random* samples; this demonstrates the performance of few-shot modeling without any simulation. To contextualize performance, we also compare to a model which uses 10× more few-shot samples, and a fully supervised model.

We also evaluate all of our models on the held-out Test set, which contains entirely unseen goals and conversations. Here, we evaluate whether our bootstrap procedure can be used in data-rich environments by applying it to a fully-supervised model as well.

	In-Domain		Out-of-Dom	
Model	JGA	TSR	JGA	TSR
Schema-Agnostic	.878	.292	.777	.000
Schema-Aware	**.960**	**.839**	**.880**	**.369**

Table 3: **Task-pretrained BART models with and without Schemas** on validation data. Schemas help Assistant models make correct API calls and are necessary for any successful Out-of-Domain simulation.

5.2 Results

Use of Schemas Results of our first experiment are shown in Table 3. Across both In-Domain and Out-of-Domain, we see a substantial rise in performance in Schema-Aware models. This is unsurprising, as providing Schemas allows the model to bypass Intent detection. Indeed, providing Schemas has a dramatic effect on Out-of-Domain performance, boosting JGA well above baseline performance, and enabling a non-zero Task Success Rate. Such successful conversations form the basis of our synthetic data during bootstrapping, and thus critical to our methodology.

Zero-shot Results for our Zero-shot experiments are shown in Figure 4, with additional metrics provided in the Appendix A.4. Overall, we find that Out-of-Domain JGA performance goes up by a total of 8.3 absolute points, or about a 37% reduction in total errors, despite having no access to domain-specific data. However, JGA plateaus after one iteration of simulation training and further iterations provide marginal negative value.

Meanwhile, TSR continues to improve for 3 iterations before eventually plateauing at approximately the level of the Fully-Supervised model; this suggests that JGA and TSR are no longer coupled, and that our bootstrapping procedure primarily optimizes its selection criteria: Task Success. To ensure the model was not simply making random API guesses to maximize TSR, we counted the number of API calls in simulation, and found the model converged on approximately 1 call per dialogue, matching the desired distribution.

We also find that Assistant BLEU does not change significantly compared to baseline, suggesting that improvements to Task Success do not translate to improvements in Natural Language Generation. Finally, we see that In-Domain JGA remains unchanged relative to the baseline, demonstrating that the addition of synthetic data does not come at the cost of performance in prior learned domains.

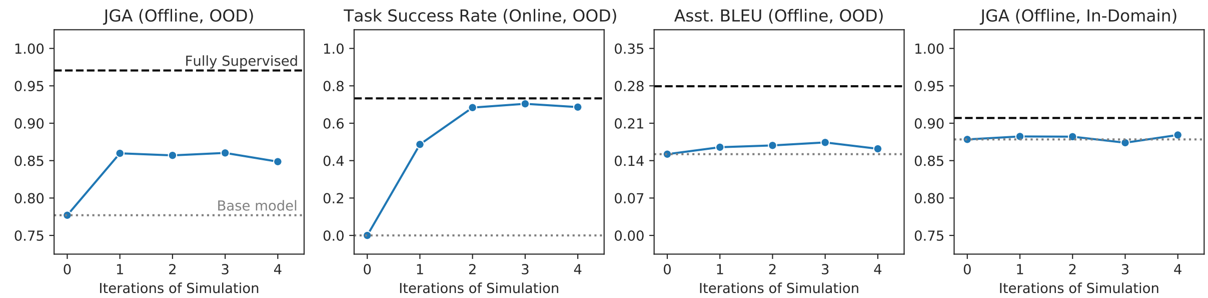

Figure 4: **Bootstrapped Validation Performance, Zero-shot.** Out-of-Domain JGA and TSR both improve through use of only synthetically generated data. BLEU and In-Domain JGA are relatively unaffected.

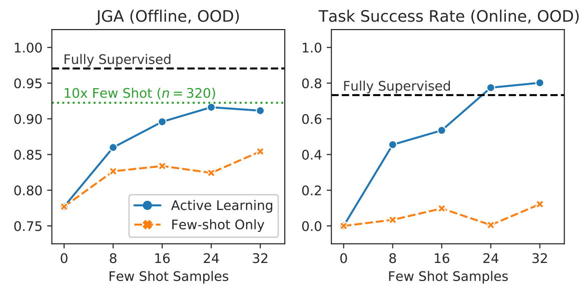

Figure 5: **Bootstrapped Validation Performance with Active Learning.** Active Learning vastly outperforms a model with an equivalent number of few-shot samples. JGA and TSR match performance of a baseline with 10× more domain-specific samples.

	In-Domain		Out-of-Dom	
Model	JGA	TSR	JGA	TSR
Base Model	.829	.394	.770	.000
Zero-Shot Simulation	**.838**	**.454**	**.860**	**.779**
Few-Shot Only ($n = 32$)	**.835**	**.459**	.852	.140
Active Learning ($n = 32$)	.830	.362	**.911**	**.799**
Fully Supervised	.895	.551	.973	.769
Fully Sup. + Simulation	**.895**	**.555**	**.977**	**.847**

Table 4: **Test set results.** Final performance on the held out Test-set for both In-Domain and Out-of-Domain.

Active Learning Results for our Active Learning experiments are shown in Figure 5. Contrary to the Zero-shot models, we see that JGA performance consistently improves for 3 iterations, finishing with a total of 13.4 points over the Zero-shot baseline and matching the performance of Few-shot model with 320 dialogues (in green), 10× the number available to the Active Learning model. The Active Learning model also shows a large gain over the sample-equivalent Few-shot model (dashed orange), and that the gain increases with the number of samples. These results demonstrates that our use of Task Success strongly improves our sample-efficiency. TSR performance continues to improve, and eventually exceeds the fully-supervised model that was trained with 72× more data.

Although not shown, we find that Assistant BLEU, as in our Zero-shot experiments similarly does not significantly improve. This indicates that TSR is more strongly correlated with JGA, and its optimization primarily benefits NLU. In-Domain JGA also remains flat, confirming that synthetic data does not lower existing performance. Additional metrics are provided in Appendix A.4.

Test Set Results To ensure that our methodology did not overfit via leaking of goals, we report final Test Set performance for a fully held-out set of Out-of-Domain data. For Simulation-based models, we evaluated models after 4 iterations. Results are shown in Table 4. We find that results are consistent with our earlier analysis. Zero-shot JGA improves 9 points over the baseline, and Active Learning gains 14 points over the baseline. Task Success Rate shows larger improvements, and matches the Fully-supervised baseline. Both models outperform the Few-shot only baseline. Finally, we see that our simulation procedure remains useful even in data-rich environments: adding simulations to a fully supervised model improves JGA by 0.4 absolute points (15% error reduction).

Human Evaluation We perform a final human evaluation using each the models from our experimental conditions. We repeat the ACUTE-Evals described in Section 4.2, using synthetic dialogues from each condition. Results are shown in Figure 6.

We find that all human judgements are roughly consistent with our expectations from offline evaluation, with clear wins for Active Learning over the Baseline and Few-shot models. The Human gold data outperforms all of our models, indicating further avenues for improvement. Nonetheless,

Lose % \ Win %	B	F	Z	A	H
Base Model		**.60**	**.73**	**.59**	**.72**
Few-Shot only	**.40**		**.60**	**.62**	**.75**
Zero-Shot Sim.	**.27**	**.40**		*.42*	**.66**
Active Learning	**.41**	**.38**	*.58*		**.69**
Human	**.28**	**.25**	**.34**	**.31**	

Figure 6: **Pairwise Human Evaluations of bootstrapped models**. Blue entries agree with automatic metrics; red italics disagree. Bold numbers indicate statistical significance ($p < .05$, binomial test).

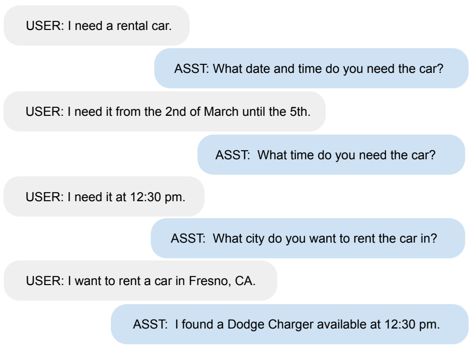

Figure 7: **Synthetic Conversation** from our final Active Learning model with *Greedy* generation. The conversation is successful, but linguistic variation is low.

the win rate of Humans decreases in our models compared to the Baseline and Few-shot models.

In analysis of annotator preferences, we find that annotator selection is generally well-correlated with TSR: Annotators preferred unsuccessful conversations over successful conversations in only about 10% of pairings. Simplicity and clarity were oftentimes given as rationale for preference in these unaligned pairings. Overall, annotators preferred Assistants that had clear communication. Many of the models learned to ask confirmation questions; this was generally liked as long as there was not too much back and forth between the User and Assistant models in doing so. Some annotators even preferred generated conversations with confirmations over the gold, human conversations. While some annotators preferred "friendlier" or "more conversational" Assistants, this was not a consistent preference; some annotators found similar conversations to be "weirdly informal," or to "take too long to get to the point."

5.3 Limitations

While our bootstrapping and Active Learning procedures do significantly improve the robustness of models, they do surprisingly little to affect linguistic diversity, especially when using greedy generation. As a representative example, see Figure 7. In it, we observe that the conversation devolves to a simple slot-filling questionnaire, with the User beginning many utterances with "I need. . ." Manually reviewing simulated conversations found that while hallucination is very low in greedy-generated dialogues, most dialogues form roughly the slot-filling questionnaire pattern. While the synthetic conversations are plausible, their linguistic diversity is extremely low, explaining why our TSR reaches near perfect levels: the User learns to specify things as simply as possible. Furthermore, we find one of our domains (*Make payments*) has multiple instances of infinite-loops being generated, a common issue known in Neural Language Models (Holtzman et al., 2020; Welleck et al., 2019).

These examples, along with the lack of improvements in NLG metrics (BLEU scores), show that Task Success is likely to reduce the linguistic diversity of Assistants or Users, unless generation methods prone to hallucination are used. In the future, identifying automatic-filtering techniques for NLG utterances, similar to Task Success and slot filling, could help with this problem. Other generation methods, like Diverse Beam Search (Vijayakumar et al., 2016), may also be able to perform a better hallucination-diversity trade-off. Nonetheless, the offline metrics on the Test set demonstrate that our methodology does improves robustness of the NLU components of our model, as indicated by the Out-of-Domain JGA metrics.

6 Conclusion

We explored the use of pretrained Language Models as User Simulators in order to generate synthetic dialogues, and filter these models for quality using Task Success. We demonstrated our methodology can be used to improve models in zero-shot, few-shot, and full-shot manners. By incorporating Active Learning, we additionally show that our models are able to bootstrap NLU performance to that of a model with 10× more training data. We encourage future work to look for improved generation methods which improve diversity without hallucination, and to find methods for automatically grading the quality of generations. Other improvements, such as the use of Schema Descriptions (Rastogi et al., 2019; Lin et al., 2021), may provide further generalization on unseen domains.

References

Anish Acharya, Suranjit Adhikari, Sanchit Agarwal, Vincent Auvray, Nehal Belgamwar, Arijit Biswas, Shubhra Chandra, Tagyoung Chung, Maryam Fazel-Zarandi, Raefer Gabriel, Shuyang Gao, Rahul Goel, Dilek Hakkani-Tur, Jan Jezabek, Abhay Jha, Jiun-Yu Kao, Prakash Krishnan, Peter Ku, Anuj Goyal, Chien-Wei Lin, Qing Liu, Arindam Mandal, Angeliki Metallinou, Vishal Naik, Yi Pan, Shachi Paul, Vittorio Perera, Abhishek Sethi, Minmin Shen, Nikko Strom, and Eddie Wang. 2021. Alexa conversations: An extensible data-driven approach for building task-oriented dialogue systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 125–132, Online. Association for Computational Linguistics.

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. *ICLR*.

Hua Ai and Fuliang Weng. 2008. User simulation as testing for spoken dialog systems. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 164–171.

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *arXiv preprint arXiv:1607.00070*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Vevake Balaraman and Bernardo Magnini. 2021. Domain-aware dialogue state tracker for multi-domain dialogue systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:866–873.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Bill Byrne, Karthik Krishnamoorthi, Saravanan Ganesh, and Mihir Kale. 2021. TicketTalk: Toward human-level performance with end-to-end, transaction-based dialog systems. 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)*, pages 671–680, Online. Association for Computational Linguistics.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online. Association for Computational Linguistics.

Paul A Crook and Alex Marin. 2017. Sequence to sequence modeling for user simulation in dialog systems. In *INTERSPEECH*, pages 1706–1710.

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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Maryam Fazel-Zarandi, Shang-Wen Li, Jin Cao, Jared Casale, Peter Henderson, David Whitney, and Alborz Geramifard. 2017. Learning robust dialog policies in noisy environments. *arXiv preprint arXiv:1712.04034*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: learning and evaluation. In *INTERSPEECH*.

Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. 2018. User modeling for task oriented dialogues. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906. IEEE.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Sangkeun Jung, Cheongjae Lee, Kyungduk Kim, Minwoo Jeong, and Gary Geunbae Lee. 2009. Data-driven user simulation for automated evaluation of spoken dialog systems. *Comput. Speech Lang.*, 23(4):479–509.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Sungdong Kim, Minsuk Chang, and Sang-Woo Lee. 2021. NeuralWOZ: Learning to collect task-oriented dialogue via model-based simulation. 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)*, pages 3704–3717, Online. Association for Computational Linguistics.

Florian Kreyssig, Iñigo Casanueva, Paweł Budzianowski, and Milica Gasic. 2018. Neural user simulation for corpus-based policy optimisation of spoken dialogue systems. *ArXiv*, abs/1805.06966.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: Improved dialogue evaluation with optimized questions and multi-turn comparisons. In *NeurIPS workshop on Conversational AI*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, et al. 2021. Zero-shot dialogue state tracking via cross-task transfer. *arXiv preprint arXiv:2109.04655*.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. MinTL: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405, Online. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2018. End-to-end learning of task-oriented dialogs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 67–73, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Biswesh Mohapatra, Gaurav Pandey, Danish Contractor, and Sachindra Joshi. 2021. Simulated chats for building dialog systems: Learning to generate conversations from instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1190–1203, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fredrik Olsson. 2009. A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240, Copenhagen, Denmark. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 45–54, Lisbon, Portugal. Special Interest Group on Discourse and Dialogue (SIGdial).

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152, Rochester, New York. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Pararth Shah, Dilek Hakkani-Tür, Bing Liu, and Gokhan Tür. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51, New Orleans - Louisiana. Association for Computational Linguistics.

Weiyan Shi, Kun Qian, Xuewei Wang, and Zhou Yu. 2019. How to build user simulators to train RL-based dialog systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1990–2000, Hong Kong, China. Association for Computational Linguistics.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. *arXiv preprint arXiv:1808.09442*.

Gokhan Tur, Robert E Schapire, and Dilek Hakkani-Tur. 2003. Active learning for spoken language understanding. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 1, pages I–I. IEEE.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Marilyn Walker, Ace Kamm, and Diane Litman. 2000. Towards developing general models of usability with paradise. *Natural Language Engineering*, 6.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. PARADISE: A framework for evaluating spoken dialogue agents. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 271–280, Madrid, Spain. Association for Computational Linguistics.

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards zero-label language learning. *arXiv preprint arXiv:2109.09193*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929, Online. Association for Computational Linguistics.

A Appendix

A.1 Screenshot of Annotator UI

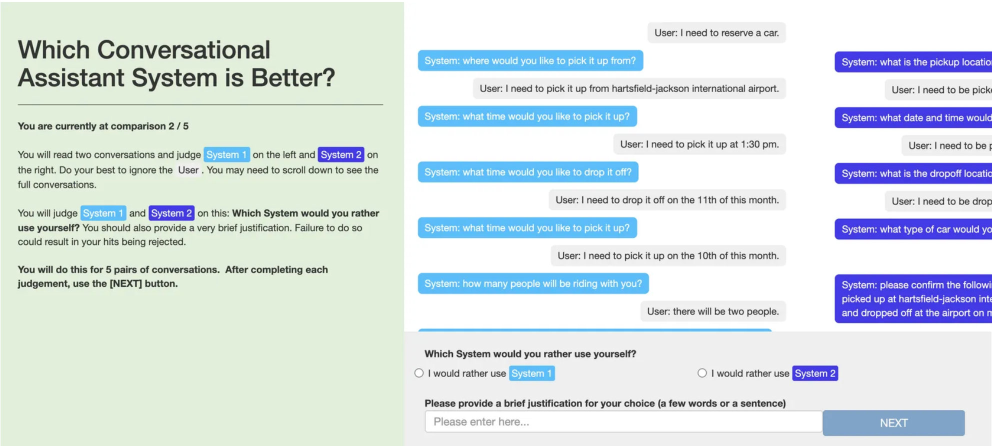

Figure 8: **Screenshot of Annotator UI**. Annotators are asked to evaluate "Which Conversational Assistant System is better" by pressing radio buttons corresponding to two presented conversations.

A.2 Pretraining Datasets

Dataset	Dial.	Dom.	Overlap with Google SGD
GoogleSGD In-Domain	18,986	16	Alarm, Banks, Buses, Calendar, Events, Flights, Hotels, Media, Movies, Music, Restaurants, Rideshare, Services, Travel, Trains, Weather
GoogleSGD Out-of-Domain	3,839	4	Home Search, Messaging, Payment, Rental Cars
MetaLWoZ	37,884	47	Banks, Buses, Events, Movies, Music, Restaurants
MSR-E2E	10,087	3	Movies, Restaurants, Taxis
MultiDoGo	19,522	6	Calendar, Flights, Media, Weather
MultiWoz	10,438	7	Attractions, Hospitals, Hotels, Restaurants, Taxis, Train
Taskmaster-1	13,215	6	Restaurants, Rideshare
Taskmaster-2	17,289	7	Flights, Hotels, Movies, Music, Restaurants
TicketTalk	23,789	1	Movies

Table 5: **Detailed statistics of datasets used in our work**. All datasets except for Google SGD Out-of-Domain used in pretraining.

We describe the different datasets that we use for pretraining. In general, we attempt to make these datasets be structured as similarly as possible to the conversations format as described in Sec 2 and generate data for separate User and Assistant models once formatted. For datasets with no API Call or Response labels, we imitate these values by accumulating dialogue state across user and assistant responses, respectively, and presenting these on appropriate turns. Other exceptions are described inline.

See Table 5 for dataset statistics.

Google SGD We describe Google SGD in 4.1. We describe our method for splitting Google SGD into In-Domain and Out-of-Domain splits in 5.1.

For our Out-of-Domain split, we used Home Search, Messaging, Payment, Rental Cars as 4 holdout domains that did not have analogues in any of the other datasets. Though the "Services" and "Travel" domains do not occur explicitly in the other datasets, we do not include them in our holdout since they include semantically similar information to the "Hospital" and "Attractions" domains of MultiWoz, respectively. For our In-Domain split, we include solely the 16 other domains of the dataset.

As also mentioned in 5.1, since Google SGD is a dataset that contains both single-goal and mulit-goal conversations, some domains of the In-Domain split are present as goals in multi-goal conversations of the Out-of-Domain split.

MetalWoz MetalWoz is a dataset constructed in a Wizard of Oz fashion across 227 tasks and 47 domains. Given a domain and a task, conversing pairs were asked to chat for 10 turns to satisfy the user's queries.

As this dataset does not include any annotations about API Calls, API Responses, or belief state, we pretrain on this dataset as-is and do not attempt to transform it into the format described in Sec 2. We do however split this dataset into separate User and Assistant versions.

MultiDoGo MultiDoGo is a large task-oriented dataset collected in a Wizard of Oz fashion, using both crowd and expert annotators with annotations at varying levels of granularity. We use only the data available publicly on this dataset's open-source repository (about 20k dialogues total.)

MultiWoz MultiWoz is a dataset of single and multi-goal human-human conversations collected in a Wizard of Oz fashion. Validation and test sets contain only successful conversations while the train set include some that are incomplete. Data of the original dataset is labelled with belief states.

MSR-E2E MSR-E2E is a dataset of human-human conversations in which one human plays the role of an Agent and the other one plays the role of a User. Data is collected from Amazon Mechanical Turk.

Taskmaster 1 Conversations in Taskmaster 1 were collected in one of two ways: spoken Wizard of Oz conversations between humans (transcribed to text) as well as written conversations from a single human in a self-dialog method. Similar to our conversations format, rather than being labelled with intents and dialog acts, conversations of this dataset are labelled with simple API arguments.

Taskmaster 2 Taskmaster2 is a dataset of entirely spoken two person dialogues collected in a Wizard of Oz manner where Assistant utterances were typed by a human and then "spoken" using a text-to-speech service. Dialogues in this dataset includes those that are search and recommendations oriented, rather than purely task execution.

TicketTalk TicketTalk (or Taskmaster 3) is a dataset of movie ticket dialogues collected in a self-chat manner. To induce conversational variety, crowd workers were asked to generate conversations given dozens of different instructions of different level of specificity, some purposefully including conversational errors.

A.3 Hyperparameter Tables

We include hyperparameter tables for each of our models used in this paper. All models were trained using the ADAM optimizer. All models were optimized using Token Exact Match (examples with perfect greedy decoding) as an early stopping criteria.

Evaluating Synthetic Dialog Generation and its Metrics Note that as described in Sec 4, we aim to look for reasonable correlations between metrics and model architectures rather than absolute performance.

LSTM & LSTM with Attention: (1 GPU per run)

Hyperparameter	Swept Values
Learning Rate	1e{-3, -4}
Number of Layers	{1, 2, 4}
Embedding size	{256, 384}
Hidden size	{1024, 2048}
Batch size	64
Embedding Init	FastText

GPT2: (8 GPUs per run)

Hyperparameter	Swept Values
Learning Rate	1e{-5, -6}
LR Scheduler	Reduce on Plateau, Invsqrt
Model Size	124M
Text Truncate	512
Warm-up Updates	100
Batch size	4
Update Frequency	2
Gradient Clip	1

BART: (8 GPUs per run)

Hyperparameter	Swept Values
Learning Rate	1e{-5, -6}
LR Scheduler	{Reduce on Plateau, Invsqrt}
Model Size	400M
Text Truncate	512
Warm-up Updates	100
Batch size	4
Update Frequency	2
Gradient Clip	1

Bootstrapping Novel Domains Once we early stopped models on Token Exact Match, we used Task Success Rate on validation goals of Google SGD to select the best model out of a given hyperparameter sweep. However, a post hoc analysis suggests that using Token Exact Match on synthetic data fine-tunes would have worked approximately as well for the goal of improving JGA.

BART: (8 GPUs per run)

Hyperparameter	Swept Values
Learning Rate	{1e-4, 5e-5, 1e-5, 5e-6}
Model Size	400M
Batch size	4
Update frequency	8
LR Scheduler	Invsqrt
Warm-up updates	1000
Text truncate	512
Label truncate	512
Gradient Clip	0.1
Multitask Weights	1
Validation Steps	100

A.4 Additional Results

We report additional metrics on each of our models, for both offline (static, held-out data) and online (during simulation) settings.

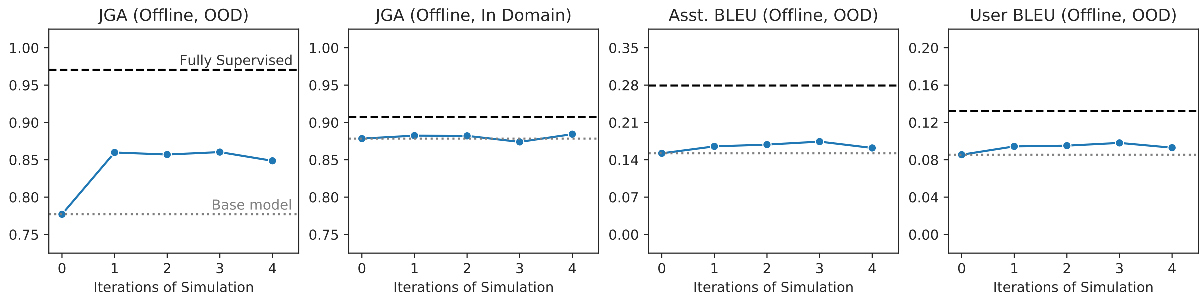

Figure 9: **Offline Bootstrapping Results.** Results of Bootstrapping on a static (held-out) offline dataset.

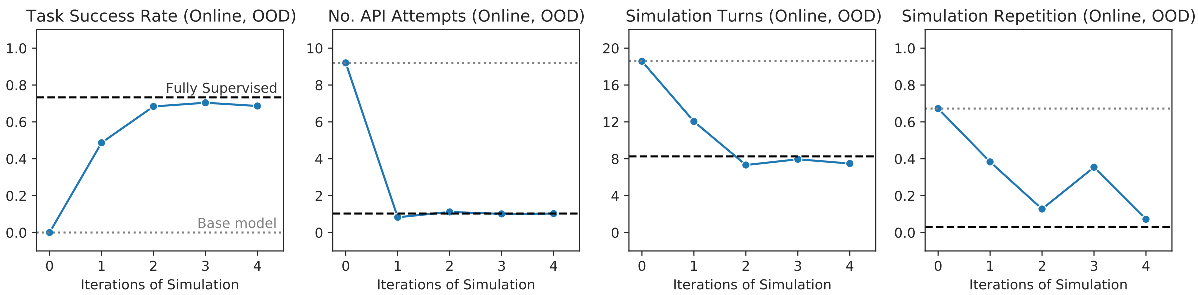

Figure 10: **Online Bootstrapping Results.** Results of Bootstrapping models on during simulations.

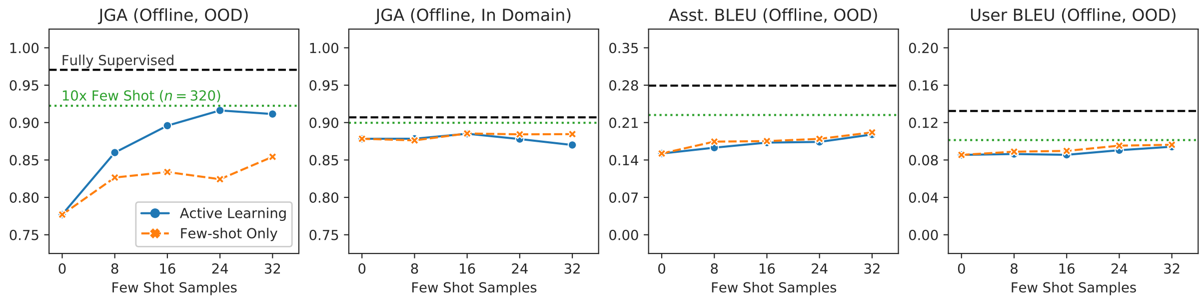

Figure 11: **Offline Active Learning Results.** Results of Active Learning on a static (held-out) offline dataset.

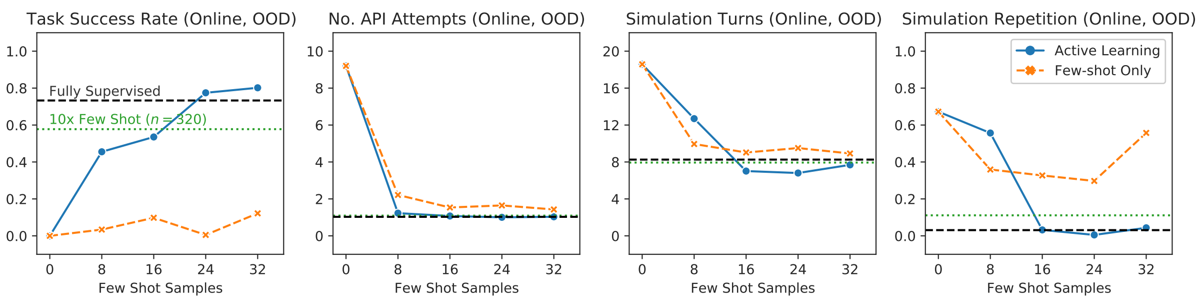

Figure 12: **Online Active Learning Results.** Results of Active Learning model during simulations.

A.5 Holdout API Analysis of Bootstrap Procedure

We take the models generated from Sec 5.1 and evaluate the models for JGA, limiting only turns to which include API calls, over each of the holdout domains. Results of this are shown in Figure 13. We observe that all holdout domains have a JGA value of zero for the Base model; this validates our selection of holdout domains. We also observe that compared to the other models, Active Learning performs much better across all domains.

	Domain			
Model	Find Homes	Payment	Rental Cars	Messaging
Base Model	.000	.000	.000	.000
Few-Shot Only	.603	.022	.411	.768
Zero-Shot Sim.	.876	.022	.266	.929
Active Learning	**.882**	**.870**	**.623**	**.946**

Figure 13: **JGA of individual Holdout Domains** (limited to API Call Turns only).