TicketTalk: Toward human-level performance with end-to-end, transaction-based dialog systems

Bill Byrne*, Karthik Krishnamoorthi*, Saravanan Ganesh*, Mihir Kale
Google, Mountain View, CA
{billb,krishnamoorthi,srrvnn,mihirkale}@google.com

Abstract

We present a data-driven, end-to-end approach to transaction-based dialog systems that performs at near-human levels in terms of verbal response quality and factual grounding accuracy. We show that two essential components of the system produce these results: a sufficiently large and diverse, in-domain labeled dataset, and a neural network-based, pre-trained model that generates both verbal responses and API call predictions. In terms of data, we introduce TicketTalk, a movie ticketing dialog dataset with 23,789 annotated conversations. The movie ticketing conversations range from completely open-ended and unrestricted to more structured, both in terms of their knowledge base, discourse features, and number of turns. In qualitative human evaluations, model-generated responses trained on just 10,000 TicketTalk dialogs were rated to "make sense" 86.5% of the time, almost the same as human responses in the same contexts. Our simple, API-focused annotation schema results in a much easier labeling task making it faster and more cost effective. It is also the key component for being able to predict API calls accurately. We handle factual grounding by incorporating API calls in the training data, allowing our model to learn which actions to take and when. Trained on the same 10,000-dialog set, the model's API call predictions were rated to be correct 93.9% of the time in our evaluations, surpassing the ratings for the corresponding human labels. We show how API prediction and response generation scores improve as the dataset size incrementally increases from 5000 to 21,000 dialogs. Our analysis also clearly illustrates the benefits of pre-training. To facilitate future work on transaction-based dialog systems, we have published the TicketTalk dataset at https://git.io/JL8an.

*Equal contribution

1 Introduction

Building a dialog system that handles human conversational behavior is challenging because it must respond sensibly and relevantly to a wide variety of context-sensitive user input over multiple conversation turns. Task-based systems, e.g. those used for ticket booking, food ordering, etc., face further hurdles to incorporate ever changing, real-world knowledge into the dialog and execute transactions. Recently, there has been growing interest in the so-called end-to-end approach to task-based dialog systems (Peng et al., 2020; Hosseini-Asl et al., 2020; Lin et al., 2020; Wen et al., 2017; Bordes et al., 2016) due to its relatively simple and scalable architecture, and promising results in chatbot applications (Vinyals and Le, 2015; Serban et al., 2015b). Inspired by sequence-to-sequence learning (Sutskever et al., 2014), this approach trains a single model on a dialog dataset to form the basis for a given application. For each dialog turn, the model effectively takes the conversation history as its input and generates an appropriate response.

To gain wider adoption, the end-to-end approach must overcome challenges with respect to training data and factual grounding. In terms of training data, there is already general concern in the NLP community about the lack of quality, task-oriented dialog datasets, especially domain-specific collections (Wen et al., 2017; Bordes et al., 2016). This problem is compounded for end-to-end approaches since they typically require a large amount of in-domain data to generate competitive results. With respect to grounding, since the end-to-end approach is based on a single neural network, it must either incorporate the knowledge base (KB) into the model itself, or the model must be able to accurately predict which API calls to make and when. In addition, details returned from the API calls must be accurately incorporated in conversational

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 671–680
August 1–6, 2021.

responses. This is contrasted with modular architectures where the user's intent is derived from a structured representation and then used to determine which API calls to make such as in Rastogi et al. (2020) and Madotto (2020).

In this work we promote an end-to-end approach to single-domain, transaction-based dialog systems and describe how we overcome both data and grounding challenges described above. In qualitative evaluations, our models perform on par with humans in generating verbal responses as well as predicting API calls. Just two components form the basis for this system: a sufficiently large, in-domain, labeled dataset and a pre-trained transformer model. Combining natural language output and structured API calls into a unified text-to-text-format allows us to leverage general purpose text-to-text transformers to train models. Specifically, we use the T5 infrastructure (Raffel et al., 2019) and show that its pre-training feature has a significant impact on evaluations, boosting scores by 30 percent.

Models were trained on our TicketTalk dataset (aka Taskmaster-3), a movie ticketing dialog corpus with 23,789 conversations labeled with a simple yet unique API-based annotation schema. This makes it one of the largest single-domain datasets to date. A public release of the dataset accompanies this paper. We chose movie ticketing since it is both transaction-based and relatively complex, but our overall approach to dialog systems applies to any task-based domain. While there is a lot of recent work on multi-domain task-based dialog systems, human-like interaction for even single-domain tasks has yet to be demonstrated. By first solving the problem for a single domain, we argue that replicating the process for multiple domains will be achievable by simply training on additional high-quality datasets labeled with the same API-focused strategy.

2 Related work and background

2.1 Datasets

Over the past few years the NLP community has responded to the lack of dialog data with larger, publicly released task-oriented datasets spanning multiple domains (Wu et al., 2020; Budzianowski and Vulić, 2019). This underscores the crucial role data plays in any approach to task-based dialog systems. MultiWOZ (Budzianowski et al., 2018) consists of 10,420 dialogs in multiple domains and has become a popular benchmarking corpus for state tracking. It has also undergone a series of subsequent refinements. MSR-E2E, featured in the Microsoft dialog challenge (Li et al., 2018), has 10,087 dialogues in three domains, movie-ticket booking, restaurant reservation, and taxi booking. Taskmaster-1 (Byrne et al., 2019) offers 13,215 dialogs in six domains and has been updated with a second installment, Taskmaster-2 (Byrne et al., 2020), which adds 17,289 more dialogs totalling over 30,000. The Schema Guided Dialogue dataset (Rastogi et al., 2020) has 22,825 dialogs in multiple domains. MetaLWOZ (Lee et al., 2019) has 37,884 dialogs in 227 domains and is aimed at helping models more accurately predict user responses in new domains. Both Schema and MetaLWOZ are used in DSTC8 (Kim et al., 2019). In addition to these, Serban et al. (2018) provides a thorough survey of dialog corpora released in previous years.

2.2 Modular vs. end-to-end architectures

In contrast to the end-to-end [1] approach, traditional, modular strategies employ a division of labor among the components, e.g. understanding, state tracking, dialog policy, generation, etc., which are either largely hand-crafted or derived from training individual models on labeled datasets (Wen et al., 2017; Young et al., 2013). This architecture is inherently more complex than the single-model end-to-end strategy we propose and can require significantly more design and engineering. Moreover, since each module requires its own supervised training dataset, it is harder to apply to different domains (Serban et al., 2015a).

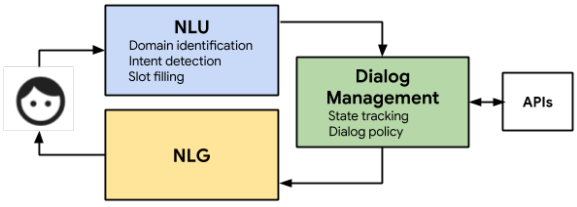

Figure 1: Traditional modular system

However, the separation of functions makes the modular approach more transparent and in some respects easier to debug. It has also been considered by some to be better equipped to interact with external APIs (Sukhbaatar et al., 2015; Wen et al., 2017)

[1] The term "end-to-end" is sometimes also used when describing parts of modular systems (Li et al., 2017; Wen et al., 2017) but it is fundamentally different from the single text-to-text transformer model approach we present here.

and therefore might be better suited for task-based dialogs. As mentioned above, we show that our single model-based approach can accurately generate both the appropriate response as well as predict the correct API call at the right time. Earlier work by Andreas et al. (2020) and Hosseini-Asl et al. (2020) employs a similar modeling approach to predict dialog state in task-based dialogs, which can be seen as a precursor to our API call prediction strategy.

Figure 2: Simplified end-to-end system

3 The TicketTalk dataset

3.1 Overview

The TicketTalk movie ticketing dataset was created using the self-dialog collection method (Krause et al., 2017; Moghe et al., 2018; Byrne et al., 2019) in which a paid crowd-sourced worker writes both sides of the dialog (i.e. both customer and ticketing agent turns) based on a particular scenario and set of instructions. Following the annotation strategy used for Taskmaster-1 (Byrne et al., 2019), labels are limited to basic entities and events (i.e. API calls). The dataset was created by over 4000 unique, native or near-native US English speakers. Further demographic information (e.g. gender, dialect, etc.) is not known, and no personal identifiable information was gathered.

STAT TYPE	VALUE
Dialogs	23,789
Total turns	481,632
Unique tokens	62,868
Avg. turns per dialog	20.25
Avg. tokens per turn	10.35
Unique named entities	57,285

Table 1: TicketTalk Dataset Statistics

The rationale for limiting dialogs to a single domain (movie ticketing) is based on our hypothesis that human-level performance in terms of both response generation and API call prediction for a particular task requires larger (i.e. 10,000+), more diverse datasets than are currently available. In other words, carefully curated, annotated datasets that cover all the idiosyncrasies of a single task or transaction are a key factor in model performance. Concern about the cost and efficiency of creating these larger corpora has led some researchers to look for approaches that alleviate dependencies on annotated data (Budzianowski and Vulić, 2019; Wen et al., 2017). However, significant time and expense can be saved when assembling these corpora by simplifying the collection and annotation procedures. In addition, little to no training is required for workers to be able to perform consistently well.

3.2 Collection methodology

Using self-dialogs (where a worker creates the whole conversation, both user and agent turns) facilitates building large and linguistically rich datasets since it is both simple and cost effective, and allows users to draw on their lifetime of conversational experiences. This in turn ensures the model can handle the wide range of human conversational behaviors that emerge in natural dialog. For this project we extended the self-dialog to include over three dozen sets of user instructions to generate a wider variety of conversations, from open-ended prompts to more specific instructions that require specific types of exchanges. For example, one set simply instructs workers to "write the transcription of a conversation" in which a person makes a successful ticket transaction with a booking agent. This allows dialog creators to express their unique view of what a typical movie ticketing transaction would be, structuring each conversation how they see fit. They are also instructed to find real values for required details (i.e. slots) such as time, date, theater, movie, etc. using a movie or theater site of their choice for a specific location. This ensures the dataset has a large and diverse KB. In contrast, the more restrictive sets of instructions focus on specific sub-dialogs for error handling, changing a detail, entity resolution, and the like. In such cases we often provide a limited KB with one or more values for all the details so the worker can focus on the primary task of creating a realistic set of exchanges for this type of interaction. In a third type of scenario, the conversation is partially completed and the user's task is focused on a very specific part of the exchange. This allows us to "fill holes" in the data quickly and cost effectively. That is, we can create large numbers of short, conversational examples that the model does not handle

adequately and then retrain for better results.

3.3 Annotation

Dialog data annotation can be complex and time consuming even for trained linguists as it typically involves carefully and consistently labeling dialog states, user intents, and dialog acts, among others (Henderson et al., 2013; Wen et al., 2017; Budzianowski et al., 2018). The API-targeted approach is far more straightforward since only basic entities (e.g. name, time, number of tickets, theater, movie attributes, etc.) and API calls (e.g. to find theaters, movies, and show times, book tickets, etc.) are labeled. The task is therefore easier to learn, faster to complete, and cheaper to run. Moreover, as we discuss below, it fits well with the text-to-text format we use in our approach to transaction-based dialog systems. Fifteen workers performed the annotations using a web-based tool that allows for only well-formed labels. To label an API call, the API name is first selected which in turn creates the correct set of possible (arg_name, arg_value) pairs to choose from, both for inputs and responses. This ensures that the model is trained on syntactically well formed API calls. No annotations were removed from the dialogs. The full annotation schema is included with the dataset release at `https://git.io/JL8an`.

4 A novel end-to-end approach

4.1 Overview

We implement a new approach to end-to-end dialog systems by combining natural language output and structured API calls into a unified text-to-text format where the input and output are always text strings. This allows us to leverage widely available, state of the art, general purpose text-to-text transformers as the foundation of our system. Specifically, we used the publicly available Text-To-Text Transfer Transformer (T5) (Raffel et al., 2019) to train our models. The T5 framework was designed specifically to explore transfer learning techniques for NLP and includes pre-training on the Colossal Clean Crawled Corpus (C4), composed of hundreds of gigabytes of web-based English text (Raffel et al., 2019). The original pre-training objective for the C4 corpus in the T5 framework was a denoising task, i.e. recovering missing words from the input. Since this type of task scales well to multiple downstream tasks, we used our custom inputs/targets from the TicketTalk dataset to represent an end-to-end task based dialog system and ultimately achieve positive results.

4.2 Setup

We use T5-Base (Raffel et al., 2019) as our pre-trained model, which follows the transformer architecture (Vaswani et al., 2017) and consists of 220M parameters. It was pre-trained on the large scale C4 dataset mentioned above for 1M steps with a span corruption objective. We fine-tune this model on the Taskmaster-3 dataset for 40000 steps with a constant learning rate of 0.001 using 16 TPU v3 chips. The batch size was set to 131,072 tokens per batch. The maximum input sequence length and output length were set to 1024 and 256 tokens respectively.

4.3 Model and implementation

The goal of our model is to generate a text string that either serves as a verbal response to the user or that contains one or more API calls with the data required at the current stage of the conversation. Verbal responses come in two flavors: those that depend on a particular API call details and those that do not. For example, when an API is invoked to find theater names for a given movie and location, the details returned from the API call must be correctly incorporated into the system's next response, e.g. "I found two theaters, AMC 20 and Century City 16." In contrast, other verbal outputs, e.g. "What city do you plan to see the movie in?" are derived from the overall conversation history.

Given the required text-to-text format used in our approach, we identify the type and function of each string by converting the annotations to a set of tokens. As shown in Table 2 and 3, tokens identify the speaker, i.e. user vs. agent, the string type i.e. utterance vs. API call, and the details of each API call, both names as well as input parameters and values, and response parameters and values. We also tag the conversation "context" which separates the most recent turn from previous turns. Our token key is shown in Table 2.

The first step is to use tokens to represent the user and agent interactions, providing speaker information to the model by the use of "<U>" and "<A>". We then convert any API invocations into their text equivalent using tokens for marking API names, argument types and values, i.e. "<PN>", "<PAN>", etc. The results of these two steps are shown in Table 3.

U	user
A	agent
PN	program name
PAN	program argument name
PAV	program argument value
PR	program response
PRAN	program response argument name
PRAV	program response argument value
C	conversation context

Table 2: Tokens identifying string type and function

<**U**>	I'd like to watch a movie.
<**A**>	Sure. I can help you with that.
	What kind of movies are you interested in?
<**U**>	Are there any good action movies?
API call:	<**PN**>find_movies
	<**PAN**>name.genre<**PAV**>action
Response:	<**PR**>find_movies
	<**PRAN**>name.movie
	<**PRAV**>John Wick<**PRAV**>Jack Ryan
<**A**>	I found John Wick and Jack Ryan.

Table 3: Speaker turns and API calls identified with tokens

The next step is to create the model inputs and targets. We use the following algorithm to accomplish this:

1. Initialize conversation context to an empty string.
2. Iterate through the interactions and do the following:
   (a) If the sentence is a user utterance (<U>) or a program response(<PR>), add it to the model input along with the conversation context (if present).
   (b) If the sentence is an agent utterance (<A>) or program invocation (<PN>), add it to the model target.
   (c) If both model input and target have been created, output the (input, target) pair and update the conversation context to reflect this.
   (d) Continue (2) to generate the next input, target pair.

Using the these rules, the model inputs and targets are generated as in Table 4.

Once the model has been trained on inputs and targets, we can use the system to accomplish tasks in the following manner:

INPUTS	**TARGETS**
<**U**>I'd like to watch a movie.	<**A**>Sure. I can help you with that. What kind of movies are you interested in?
<**U**>Are there any good action movies? <**C**> <**U**>I'd like to watch a movie. <**A**>Sure. I can help you with that. What kind of movies are you interested in?	<**PN**>find_movies <**PAN**>name.genre <**PAV**>action
<**PR**>find_movies <**PRAN**>name.movie <**PRAV**>John Wick <**PRAV**>Jack Ryan <**C**> <**U**>I'd like to watch a movie. <**A**>Sure. I can help you with that. What kind of movies are you interested in? <**U**>Are there any good action movies? <**PN**>find_movies <**PAN**>name.genre <**PAV**>action	<**A**>I found John Wick and Jack Ryan.

Table 4: Generating inputs vs. targets

1. Obtain user utterance and format it by adding the speaker token.
2. Provide the formatted utterance to the model.
3. Obtain model prediction
   (a) If the model prediction contains the agent (<A>) token, format it and show it to the user.
      i. Update conversation context and start again from (1).
   (b) If the model prediction contains the program (<PN>) token:
      i. Extract program argument name (<PAN>) and value (<PAV>).
      ii. Issue the API call by providing it to the API adapter.
      iii. Format API results and provide it to the model along with the conversation context.
      iv. Start from (3).

This interaction lifecycle is illustrated in Figure 3.

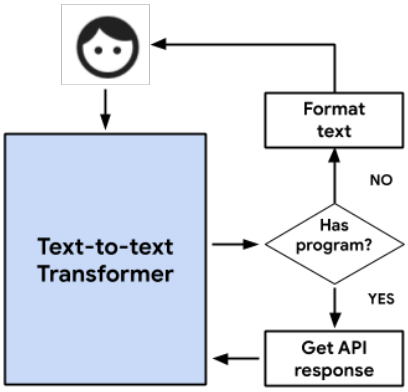

Figure 3: System interaction life cycle

4.4 Invoking APIs

When we detect an API call in the output, we invoke the API, retrieve the results, and embed the responses in the next model input. As shown in Figure 4, each API call predicted by the model typically contains a generic API name, such as "find-movies", or "find-theaters", and a list of key value pairs that detail the specific parameters to be used while invoking the API, as shown in Figure 4.

MODEL OUTPUT	API INVOCATION

<PN>find_theaters
<PAN>location
<PAV>nearby
	GET /my-movie-api/theaters?auth=_key
	
	{
	"location": {
	"zipcode": 94040
	}
	}

Figure 4: Example API invocation (outside model)

The API call, while structured, may still include pronouns or other co-referential phrases as input parameters. For example, the date parameter for an API call might contain the value "tonight", and the location value might be "nearby". The resolution of these entities happens outside the core interaction layer in what can be understood as the "API adapter" (and not the actual API itself). This not only helps simplify annotation, but also helps leverage existing solutions to these well defined problems. This separation of the API layer is also useful for encapsulating all API specific artifacts, like authentication tokens, endpoint addresses and data formatters. In this way, the end-to-end system is able to interact with the user to solicit details relevant to the task, generate API calls to fetch data from external knowledge sources, and use the responses provided by the API call to construct natural language responses.

5 Experiments

5.1 Overview

In this section, we show how our end-to-end approach to transaction-based dialog systems produces verbal responses and predicts API calls with near human-level quality and accuracy. Through human qualitative evaluations, we show that two aspects in particular, dataset size and pre-training, significantly affect performance. Below we describe our evaluation methodology followed by a detailed discussion of the experiment results.

5.2 Evaluation methodology

Dataset size and pre-training are key factors in creating models for end-to-end dialog systems. To understand the amount of data required for our approach, we trained four models, each on a different number of randomly selected subsets of the TicketTalk dataset, namely 5000, 7500, 10,000 and 21,000 dialogs. To measure the effect of transfer learning, we trained a second 10,000-dialog model without the T5 framework's pre-training component, setting up an A-B comparison with the pre-trained model.

As mentioned earlier, our models generate three types of output: API calls, verbal responses based on the results of an API call, and "plain" verbal responses based on the conversation context (i.e. not dependent on a particular API call response). We set up a pair of evaluations for each type. The first evaluation asked human raters to evaluate the model's output given a specific conversation history (i.e. context) while the second asked raters to evaluate the human's response for the same set of contexts. Each experiment included 1000 context-response pairs of varying lengths, i.e. some conversation histories might have just one exchange (a user and agent turn) while others could have up to nine exchanges. We requested three ratings for each question distributed among a pool of about 900 paid raters for a total of 3000 data points per experiment. Table 5 and Table 6 below shows a sample context-response pair presented to human raters for each type of model output.

We use our "makes-sense" metric to evaluate the model-generated responses and API call predictions against the human standard. For verbal responses, we ask one question:

- Does the agent's next response make sense?

CONTEXT	NEXT RESPONSE
Cust: Can you help me book a movie ticket? **Agent:** Yes I can. **Cust:** Can you find tickets for the movie Knives Out? **Agent:** Sure! What time did you want to book? **Cust:** 5 PM would be best.	**Agent:** OK. Do you have any theaters in mind?

Table 5: Context paired with generated verbal response

CONTEXT	ACTION
Cust: I would like to see a movie tonight. **Agent:** Sure. What movie would you like to see? **Cust:** I'm not really sure. Can you help me pick something? **Agent:** No problem. I can give you the names of a couple of movies playing in your area. What city are you going to see the movie in?	FIND_MOVIES location: Oak Valley Arkansas

Table 6: Context paired with predicted API call

For negative answers, we give a list of reasons raters believe it does not make sense (i.e. off topic, repeated information, incorrect details, grammar mistakes, other). For API call predictions there are two questions:

1. Do all the action types, their details, and their order make sense at this point in the conversation?

2. Are there any actions that should be listed here but that are missing (either as additions or replacements)?

Again, raters are given options to choose for negative answers.

The offline evaluation strategy described above offers scalability and minimal rater training. However, an online, interactive setup would further allow us to evaluate the ability of the model to handle errors in its own output (from previous predictions) and its robustness while dealing with novel inputs. We have begun to build an interactive UI to facilitate such evaluations and show promising results of such an interaction in Table 7 below. The authors of this paper played the USER role. The T5 model was trained on the full TicketTalk dataset which includes nearly 24K dialogs. If the model generates an API call, we create a value that mimics the response from the API adapter and provide it to the model before the next prediction. We also provide the model with fake API responses (for calls like find_movies and find_theaters) containing entities that have never been used in the conversations in the TicketTalk dataset.

The conversation in 7 includes the exact API responses with intentionally made up movie theater names that have been provided to the model to ensure they were not part of the training set. The model behaves correctly when provided with the made up API responses that are not in the dataset. When the dialog flow closely matches the dataset flows, which are significantly diverse and varied, we can recreate interactions like this relatively easily. This particular example took two attempts to generate. Future evaluation of our approach will include this type of interactive task where testers rate both individual as well as the overall conversation.

5.3 Results

Comparing the "makes-sense" scores for model-generated vs. human-generated responses, a clear pattern of improvement emerges based on dataset size. Table 8 presents the three types of model-generated responses evaluated: "Plain responses" (not strictly based on API results), "Responses to APIs" (based on API results), and "API calls" themselves. When 5K and 7.5K dialogs are used for the training set, scores for model-generated responses lag behind the human-generated scores by up to 5.5%. At 10K dialogs, the response scores differ by less than 2% and model-generated API predictions outperform human labels by 2.5%. At 21K dialogs, model-generated responses improve to near human-level performance. The 10K model's API call prediction fares better than 21K model for API labeling, which is likely due to the fact that, as more API call combinations are introduced, they are harder for the model to interpret. In contrast, adding general dialog data along with pre-training will improve the model's predictions of English utterances which gives the 21K model an advantage in "plain response" scores. As an automatic metric, we also provide the BLEU score generated for each model. Maximum n-gram order for computing BLEU score was set to 4. The unrestricted nature of the entities in the datasets makes it much harder to create a robust automatic metric for API call

USER	I'd like to book some tickets.
MODEL	Okay, what movie would you like to see?
USER	Whats in theaters?
API call	find_movies(location: 'nearby')
API response	name.movie: [NewMovie 2021, OldMovie 2020]
MODEL	NewMovie 2021 and OldMovie 2020
USER	What's the rating for NewMovie?
API call	get_movie_attribute(name.movie: 'NewMovie 2021', attribute: 'rating.movie')
API response	rating.movie: [7.7]
MODEL	7.7
USER	Ok wheres it playing?
API call	find_theaters(name.movie: 'NewMovie 2021', location: 'nearby')
API response	name.theater: [ClassicTheater 2000]
MODEL	ClassicTheater 2000
USER	Are there any shows this evening?
API call	find_showtimes(name.movie: 'NewMovie 2021', name.theater: 'ClassicTheater 2000', date.showing: 'today', time.preference: 'this evening')
API response	time.showing: [7:10 PM, 9:40 PM]
MODEL	Yes, there is a 7:10 PM and a 9:40 PM.
USER	Book two tickets for the 7 o clock show.
MODEL	Okay, I have you down for two tickets for the 7:10 PM show of NewMovie 2021 at ClassicTheater 2000. Is that correct?
USER	Yes, that is correct.
MODEL	Okay, would you like to use your credit card on file?
USER	yeah
API call	book_tickets(name.movie: 'NewMovie 2021', name.theater: 'ClassicTheater 2000', num.tickets: 'two', time.showing: '7:10 PM', date.showing: 'today', type.screening: 'any')
API response	booking.status: [success]
MODEL	Your tickets are confirmed and they've been sent to your email address.

Table 7: Model performance in an online interactive session.

predictions. This is compounded by the fact that any given dialog context may allow for different sets of API calls.

The effect of pre-training is also very clear. After training a fifth model, this time without the T5 framework's pre-training feature, we see a huge drop in evaluation scores. As shown at the bottom of Table 8, we see a decrease of 30% in model performance for verbal responses and about a 25% drop in API call prediction accuracy.

Finally, the quality of the model's prediction stays on par with human scores throughout the

Size	Plain Resp.	Resp. to APIs	API calls
5K			
model:	86.9% **-5.5%**	92.3% **-3.9%**	95.2% **-2.2%**
human:	92.4%	96.2%	97.4%
BLEU:	56		
7.5K			
model:	87.8% **-3%**	93.8% **-2.4%**	95.2% **-2.3%**
human:	90.8%	96.2%	97.7%
BLEU:	59		
10K			
model:	86.5% **-1.9%**	91.8% **-1.4%**	97.1% **+2.5%**
human:	88.4%	93.2%	94.6%
BLEU:	61		
21K			
model:	89.8% **-1.4%**	95.3% **-0.3%**	93.9% **+0.3%**
human:	91.2%	95.6%	93.6%
BLEU:	60		
	No Pre-training		
10K			
model:	55.8% **-32.6%**	63.1% **-30.1%**	72.8% **-21.8%**
BLEU:	51		

Table 8: Effects of training set size and pre-training on model accuracy

conversation as the context grows. Figure 5 shows how the model's "makes sense" score stay on the same path after each exchange.

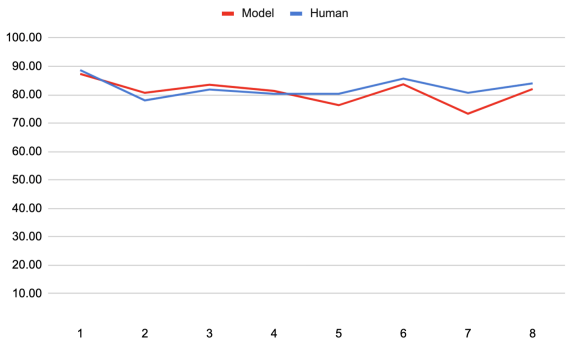

Figure 5: Model accuracy per dialog exchange

6 Conclusion

We have described an end-to-end dialog system approach that shows promising potential for transaction-based dialog applications. In offline human evaluations, our single-domain models trained on just 10,000 dialogs generate responses and predict API calls with near-human level accuracy. A

key aspect of this strategy is combining natural language output and structured API calls into a unified text-to-text format in order to leverage general purpose text-to-text transformers, such as the T5 framework. In this way, predicting which API call to make and when is essentially the same as generating the appropriate utterance at a given point in the conversation. The pre-training component significantly boosts performance on our downstream task of fine tuning models on the our datasets. These carefully curated and sufficiently large datasets are also core to this strategy, and creating them is straightforward using the self-dialog technique and simple, API-focused annotation. The TicketTalk dataset released with this paper is one such example. When compared with more traditional, modular system architectures, our end-to-end approach should significantly reduce design and engineering time and resources needed to build task-based dialog systems. Future work will include interactive evaluation of current models as well as an application of this approach to multiple-domain systems.

Acknowledgments

We would like to thank our colleagues Daniel De Freitas Adiwardana, Noam Shazeer, Filip Radlinksi, and Pedro Moreno for their discussion and insights through several iterations of this paper. We thank Hadar Shemtov for his guidance and support of the overall project.

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, et al. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

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

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's gpt-2–how can i help you? Towards the use of pretrained language models for task-oriented dialogue systems. *arXiv preprint arXiv:1907.05774*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Bill Byrne, Karthik Krishnamoorthi, Saravanan Ganesh, Amit Dubey, Andy Cedilnik, and Kyu-Young Kim. 2020. Taskmaster-2. https://github.com/google-research-datasets/Taskmaster/tree/master/TM-2-2020. Second dataset in series of three.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Andy Cedilnik, and Kyu-Young Kim. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. *arXiv preprint arXiv:1909.05358*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471.

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*.

Seokhwan Kim, Michel Galley, Chulaka Gunasekara, Sungjin Lee, Adam Atkinson, Baolin Peng, Hannes Schulz, Jianfeng Gao, Jinchao Li, Mahmoud Adada, Minlie Huang, Luis Lastras, Jonathan K. Kummerfeld, Walter S. Lasecki, Chiori Hori, Anoop Cherian, Tim K. Marks, Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, and Raghav Gupta. 2019. The eighth dialog system technology challenge.

Ben Krause, Marco Damonte, Mihai Dobre, Daniel Duma, Joachim Fainberg, Federico Fancellu, Emmanuel Kahembwe, Jianpeng Cheng, and Bonnie Webber. 2017. Edina: Building an open domain socialbot with self-dialogues. *arXiv preprint arXiv:1709.09816*.

S Lee, H Schulz, A Atkinson, J Gao, K Suleman, L El Asri, M Adada, M Huang, S Sharma, W Tay, et al. 2019. Multi-domain task-completion dialog challenge. *Dialog System Technology Challenges*, 8.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Xiujun Li, Sarah Panda, JJ (Jingjing) Liu, and Jianfeng Gao. 2018. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. In *SLT 2018*.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. Mintl: Minimalist transfer learning for task-oriented dialogue systems. *arXiv preprint arXiv:2009.12005*.

Andrea Madotto. 2020. Language models as few-shot learner for task-oriented dialogue systems. *arXiv preprint arXiv:2008.06239*.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards exploiting background knowledge for building conversation systems. *arXiv preprint arXiv:1809.08205*.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

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.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015a. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015b. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*, 7(8):434–441.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. 2020. Tod-bert: Pre-trained natural language understanding for task-oriented dialogues. *arXiv preprint arXiv:2004.06871*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.