

# CHARTDIALOGS: Plotting from Natural Language Instructions

Yutong Shao and Ndapa Nakashole

Computer Science and Engineering

University of California, San Diego

La Jolla, CA 92093

{yshao, nnakashole}@eng.ucsd.edu

## Abstract

This paper presents the problem of conversational plotting agents that carry out plotting actions from natural language instructions. To facilitate the development of such agents, we introduce CHARTDIALOGS, a new multi-turn dialog dataset, covering a popular plotting library, `matplotlib`. The dataset contains over 15,000 dialog turns from 3,200 dialogs covering the majority of `matplotlib` plot types. Extensive experiments show the best-performing method achieving 61% plotting accuracy, demonstrating that the dataset presents a non-trivial challenge for future research on this task.

## 1 Introduction

Advances in machine language understanding (Hirschberg and Manning, 2015) have sparked interest in using artificial intelligence to address difficult problems involving language. In this work, we are interested in the problem of plotting via natural language instructions. Plotting is a method for visualizing data and mathematical functions. Plotting libraries such as `matplotlib` support functionality on a range of levels, from general, “change the X-axis from *linear* to *log* scale”, to specific, “color this screen pixel red”. Yet, using such libraries can be difficult for novice users and time consuming even for experts. This obstacle, coupled with the increasing popularity of the scientific method of gleaning information from data (Hey et al., 2009; Dhar, 2013), motivates our objective of designing natural language interfaces (NLIs) for plotting.

NLIs for plotting can be organized into three categories based on what the user is expected to describe: *the data, the function, or the plot*.

**Describing the Data or the Function.** In the first category of plotting NLIs, users are expected to describe the data they would like to visualize, by posing queries such as: “Show me medals for hockey

and skating by country.” Queries may involve simple data analysis: “Is there a seasonal trend for bike usage?” The system retrieves the relevant data, performs simple data analysis, and produces a visualization. This category of NLIs has been studied in Human Computer Interaction and related areas (Gao et al., 2015; Setlur et al., 2016; Srinivasan and Stasko, 2017; Yu and Silva, 2019; Sun et al., 2010).

In the second category of plotting NLIs, users specify the function they would like to visualize. In this category, commercial products such as *wolframalpha.com* yield results for queries such as “plot the tangent to  $x^2$  at  $x = 0.5$ ”. The system processes such queries by leveraging knowledge of functions and mathematical principles.

**Describing the Plot.** In the two categories we have discussed, users only describe what data or function they would like to visualize without describing how to visualize it. The system is in charge of all plotting details, which are not accessible to users. We can think of a third, less explored, category of plotting NLIs, in which the user instructs the system on how they would like to manipulate a plot. As an example, consider the following questions from a community question answering forum for `matplotlib`<sup>1</sup>:

(Q1): “How does one change the **font size** for all elements (ticks, labels, title) on a `matplotlib` plot?”

(Q2): “I have a scatter plot graph . . . I would like the **Y-Axis** to start at the max value and go up to 0.”

(Q3): “Given a signal plot with time index ranging from 0 to 2.6(s), I want to **draw vertical red lines** indicating corresponding time index for the list.”

<sup>1</sup><https://stackoverflow.com/questions/tagged/matplotlib>

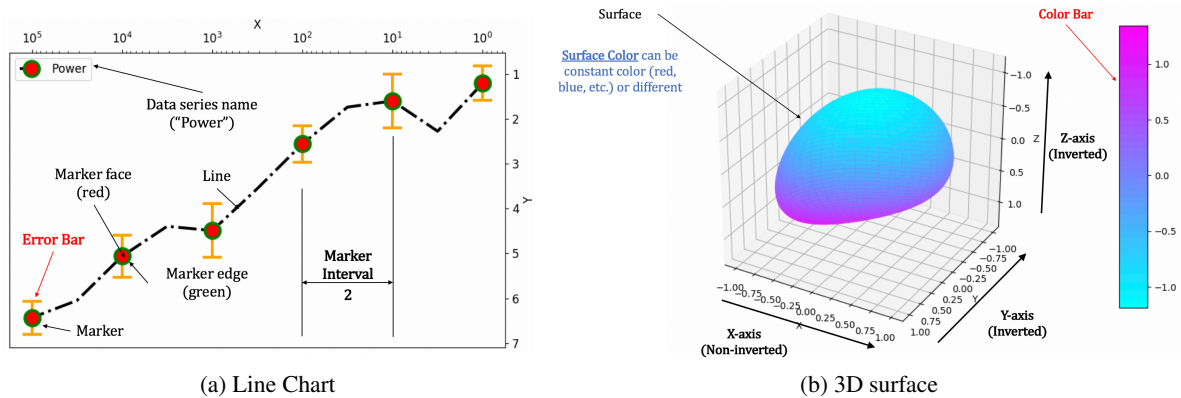


Figure 1: Illustration of two of CHARTDIALOGS plot types. (a) **Line Chart** has slots such as *Line Style*. (b) A **3D Surface** has slots such as *Surface Color*.

For **Q1**, the user’s intent is to change the font size of the elements of a plot; for **Q2**, to invert the Y-axis on the plot; and for **Q3**, to add vertical lines to a plot. All three questions seek to perform an action directly on the plot. The large number of such questions online indicates that direct plot manipulation is a common technical need. Crucially, expressing these intents in natural language is often faster than perusing the documentation of plotting library. Therefore, there is an opportunity to automatically process such intents by mapping natural language to API calls. This problem is the focus of our work.

**Contributions.** The contributions of this work are as follows: 1) We identify and define the problem of *conversational plotting agents*. 2) To facilitate work on this problem, we present a large dataset, CHARTDIALOGS, that consists of written multi-turn plotting dialogs. An in-depth analysis of the data shows that it is linguistically diverse and compares favorably to existing datasets. 3) We conducted extensive experiments in the framework of goal-oriented dialog using various methods. We also collected data on human performance, finding that there is a substantial gap between model and human performance, and therefore room for future work.<sup>2</sup>

## 2 Problem Definition

Our goal is to develop a conversational plotting agent that takes natural language instructions and updates the plot accordingly. The agent is conversational because plots can be complex, making

<sup>2</sup>We have released our dataset and code for experiments: <https://github.com/sythello/ChartDialog>

it difficult to describe everything at once. Users may want to fine-tune the appearance of their plot through multiple turns.

**Goal-Oriented Dialog Problem.** We treat the conversational plotting agent problem as an instance of slot-based goal-oriented dialog. The applicable slots are plot type specific. Figure 1 illustrates example slots for some of the plot types. Different plot types have different slots. However, some slots are shared across plot types. For example, the slot “X-axis scale” is relevant to the x-axis, thus it is applicable in any plot type with an x-axis, including line chart, bar plot, contour plot, etc. This slot can take a value such as “X-axis scale = log”, as a result of a request such as “change the x-axis scale from linear to log”.<sup>3</sup>

Illustrations of all CHARTDIALOGS plot types and their slots are provided in Appendix A.

## 3 Related Work

Goal-oriented dialog datasets largely focus on service domains such as airlines (Hemphill et al., 1990; Seneff and Polifroni, 2000; Bennett and Rudnicky, 2002; Asri et al., 2017; Budzianowski et al., 2018; Wei et al., 2018), restaurant (Henderson et al., 2014; Bordes et al., 2017), bus (Raux et al., 2005; Williams et al., 2013), technical support (Lowe et al., 2015), and car agents (Eric et al., 2017). Recently, a multi-domain goal-oriented dataset covering restaurant, attraction, hospital, police, hotel, taxi and train domains was introduced in (Budzianowski et al., 2018). Our dataset is focused

<sup>3</sup>We wrote a simple script to take as input the plot type (as a special slot) and other slot-value pairs, to generate the actual plot image using matplotlib. This script is included in the released dataset.

on a new domain, which is data plots.

Natural language interfaces to structured languages such as SQL have been explored in Databases (DB) (Li and Jagadish, 2014), Programming Languages (PL) (Yaghmazadeh et al., 2017), and NLP (Zelle and Mooney, 1996). While the problem of language to SQL is different from language to plots, both problems need to deal with the difficulty of automatically interpreting natural language and mapping it to an unambiguous structured representation.

Closer to our work is the task of conversational image editing (Manuvinaurike et al., 2018b,a), whose aim is to enable queries like “Can you please fix the glare on my dog’s eyes”. Although both focus on image manipulation, the images and manipulations are different in the two domains. Additionally, we provide structured representations from which the plot images are generated. Our experiments show that such representations provide useful information for model training. In contrast, the structured representation is not available in conversational image editing. Furthermore, our dataset contains over 3,200 dialogs in comparison to the 129 dialogs for image edits.

Lastly, our task is different from full-fledged program synthesis, which takes natural language as input and produces computer programs in a language such as Python (Church, 1957; Solar-Lezama and Bodik, 2008). Our task is simpler and more structured.

## 4 Data Collection

To facilitate data collection, we make use of structured representations which we call *text plot specifications*.

### Definition 1 (Text Plot Specification, TPSpec)

Let  $S^t$  be the set of all relevant slots for a given plot type,  $t$ , where  $t$  takes on plot type values such as *histogram*, *scatter*, etc. For each slot  $s_i \in S^t$ , let the set of values it can take be  $\mathcal{V}_i^t$ . A TPSpec of plot type  $t$  is given by:  $\mathcal{TP}^t = \{(s_1 : v_1, s_2 : v_2, \dots) : s_i \in S^t; v_i \in \mathcal{V}_i^t\}$

Thus a TPSpec is a sequence of tokens and can be considered as a structured text representation of a plot. This representation is invertible, i.e. a TPSpec can be mapped back to its corresponding slot-value pairs in a deterministic way. The design of TPSpecs is similar to how structured representations are used for dialog state tracking (Kan et al., 2018). We

leverage TPSpecs in our data collection pipeline, which consists of two steps.

**Step 1: Plot Generation.** The first step consists of generating a set of `matplotlib` plots. Since there is a one-to-one mapping between Text Plot Specifications (TPSpecs) and plot images, we only need to generate TPSpecs. Specifically, for each plot type  $t$  and all relevant slots  $s_i \in S^t$ , we design a value pool  $\mathcal{P}_i^t \subseteq \mathcal{V}_i^t$ , from which we randomly sample slot values to generate TPSpec samples.

**Step 2: Dialog Collection.** The second step involves collecting dialogs about the plots we generate in Step 1. A widely-used dialog collection scheme is the Wizard-of-Oz (WOZ) (Kelley, 1984), in which one worker plays the user and another worker plays the computer. Successful dialog datasets have been collected using Wizard-of-Oz approach, including the Air Travel Information System (ATIS) corpus (Hemphill et al., 1990), and others (Budzianowski et al., 2018; Rojas-Barahona et al., 2017; Asri et al., 2017).

We designed Wizard-of-Oz<sup>4</sup> Mechanical Turk (MTurk) tasks to have a *Describer* worker, who plays the role of the user; and an *Operator* worker, who plays the role of the plotting agent<sup>5</sup>. The *Describer* has access to a *target plot* which is the goal plot for the *Operator* to achieve, but it is not directly visible to the *Operator*; the *Operator* has access to an *operation panel* which consists of a changeable field for each slot. The *Operator* can use this panel to execute a plot function on a server. Both workers have access to the *working plot* which is the plot that the *Operator* has generated based on the *Describer*’s requests. It is initialized to a placeholder empty plot.

The *Describer* begins the conversation by writing a message in natural language, describing to the *Operator* a request that would take them closer to their goal of matching the working plot with the target plot. The *Describer* could say “invert the Y-axis”. The *Operator* can respond in natural language to ask clarification questions, or fill out slots in the operation panel and show the resulting plot to the *Describer*. For example, the operator might select the slot corresponding to “invert Y-axis=True” and the working plot is updated for both workers to see. The *describer* would continue

<sup>4</sup>Our setting is slightly different from the usual Wizard-of-Oz in that users were informed that they were conversing with fellow humans.

<sup>5</sup>Multi-worker MTurk tasks are implemented using ParlAI (Miller et al., 2017)

	DSTC2 [2014] (restaurant)	SFX [2014] (restaurant)	WOZ2.0 [2017] (restaurant)	FRAMES [2017] (travel)	KVRET [2017] (car)	M2M* [2018] (movie,rest)	ImageEdits [2018] (images)	CHARTDIALOGS [2019] (plots)
# Dialogues	1,612	1,006	600	1,369	2,425	1,500	129	<b>3,284</b>
Total # turns	<b>23,354</b>	12,396	4,472	19,986	12,732	14,796	8,890	15,754
Total # tokens	199,431	108,975	50,264	<b>251,867</b>	102,077	121,977	59,653	141,876
Avg. turns per dialo.	14.49	12.32	7.45	<b>14.60</b>	5.25	9.86	unk	4.80
Avg. tokens per turn	8.54	8.79	11.24	<b>12.60</b>	8.02	8.24	unk	9.01
Total unique tokens	986	1,473	2,142	<b>12,043</b>	2,842	1,008	2,299	2,652
# Slots	8	14	4	<b>61</b>	13	14	unk	53
# Values	212	1847	99	3871	1363	138	unk	328

Table 1: Comparison of CHARTDIALOGS to other single domain goal-oriented dialog data sets. \*M2M is largely on the restaurant domain but also includes movies

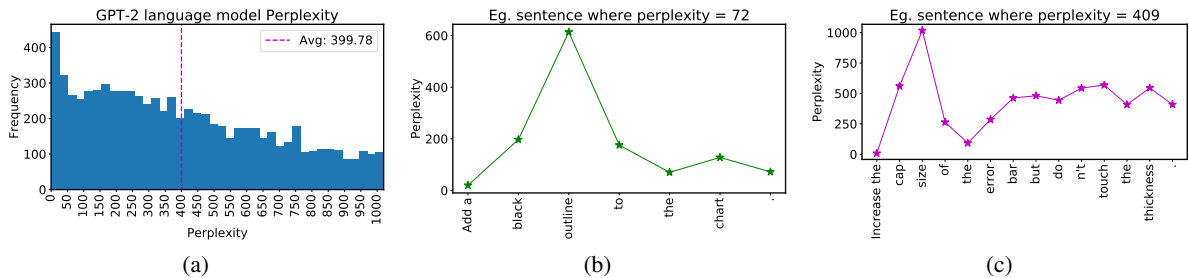


Figure 2: (a): Distribution of perplexity of the utterances. (b) and (c): average per word surprise of a growing sentence as new words are added to the sentence. High perplexity is a result of plot-specific terms ‘outline’, and ‘cap’ arising in unexpected contexts.

by, for example, saying “make the font size larger”. The two workers continue to have a dialog, taking turns until the working plot exactly matches the target plot. Screenshots of our data collection UI are shown in Figure 7 and 8 in the Appendix.

If a pair of workers failed to successfully collaborate to match the target plot, the dialog is still kept in our dataset as negative examples. However, in our exploratory method study in section 6, we skipped them for simplicity.

**Mechanical Turk Cost and Statistics.** The dataset cost \$8,244.18 to collect. The average task completion time was 6 minutes. In total, 419 workers engaged in this task; 338 of them completed at least 1 successful dialog. Workers were provided a tutorial and had to complete a test before joining the task.

## 5 CHARTDIALOGS Statistics

The collected dataset, CHARTDIALOGS, consists of 3, 284 dialogs, 15, 754 dialog turns and 141, 876 tokens in total.

**Comparison to other Datasets.** Table 1 compares our dataset to other goal-oriented datasets that are about a *single* domain, such as travel, restaurant, car, etc., on several key metrics. In particular,

we compare to: DSTC2 (Henderson et al., 2014), SFX (Gašić et al., 2014), WOZ (Wen et al., 2017), FRAMES (Asri et al., 2017), KVRET (Eric and Manning, 2017), M2M (Shah et al., 2018) and ImageEdits (Manuvinakurike et al., 2018b,a). Table 1 shows that our corpus compares favorably to other datasets and is strong on two metrics: number of dialogs, and number of slots. This is a positive indication, given the narrowness of our domain in comparison to other domains.

**Naturalness of Utterances.** We took a pre-trained language model, the Generative Pre-trained Transformer (GPT-2) of OpenAI (Radford et al., 2019), to evaluate the naturalness of utterances in our dataset. Although this language model is trained on Web text, which is different from our domain, it can be a good measure of language naturalness, at least for generic texts. Figure 2a shows GPT-2 perplexity distribution for half of the utterances, 7, 876, in CHARTDIALOGS. This half consists of the utterances with the lowest perplexity. The second half with higher perplexity forms a long-tail distribution and is omitted for plot readability.

As shown in Figure 2a, the dataset contains utterances of varying degrees of naturalness, from pure natural language (“please invert the Y-axis”), to a



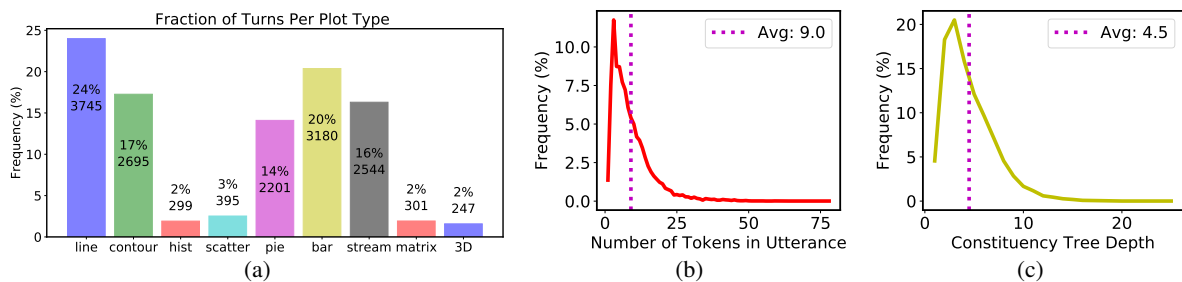


Figure 3: (a) Dialog turns per CHARTDIALOGS plot type. Distributions of (b) Words per utterance, and (c) Constituency tree depth for utterances.

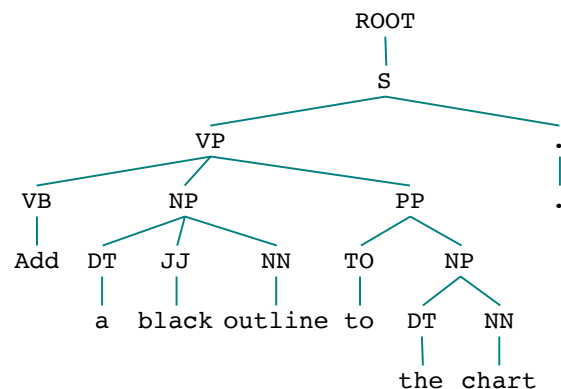
structured code-style language (“Y-axis=inverted”). This is inline with our goal to have a conversational plotting agent that deals with requests with different levels of naturalness. The average perplexity even on the first half is high at 399.77. The second half, not shown, has median perplexity of 3,776.0, and mean perplexity of 77188.58.

Figures 2b and 2c show the perplexity behavior for two utterances. The figures show the average per-word surprise of a growing sentence as new words are added to the sentence. For example, in Figure 2b, the perplexity for “add a” is low, increases for “add a black”, increases even more for “add a black outline”, and decreases for “add a black outline to”. It is clear that high perplexity of the dataset is a result of plot-specific terms like ‘outline’ in Figure 2b and ‘cap’ in Figure 2c, arising in unexpected contexts in Web text.

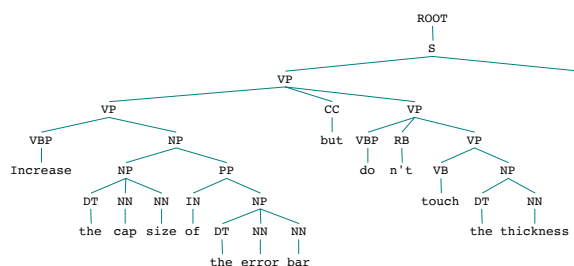
**Turns Per Plot Type.** Figure 3a shows the fraction of dialog turns per plot type. Some plot types have more dialogs and more turns than others, which is a design choice we made in collecting the dataset. Although not the subject of the current paper, we would like the plotting agent to generalize to plot types with few data points, and potentially, to plot types that were never seen before, as a challenge for few-shot or zero-shot learning methods.

**Utterance Length.** Figure 3b shows that our dataset has utterances of varying lengths in terms of tokens. The average number of tokens per utterance is 9.01, which is comparable to the average among all the datasets reported in Table 1, which is 9.57.

**Utterance Syntactic Depth.** Figure 3c shows the distribution of constituency parse tree depths from the Stanford Parser. The average tree depth is 4.5. Figure 4 shows two parse trees of different depths. The parse tree in Figure 4a for the utterance “Add a black outline to the chart” has a tree depth of 4, and reflects the nature of the average utterance. On the



(a) Constituency tree depth = 4



(b) Constituency tree depth = 8

Figure 4: Two CHARTDIALOGS utterances with different constituency tree depths. The average tree depth in the dataset is 4.5.

other hand, the parse tree in Figure 4b for “Increase the cap size of the error bar but don’t touch the thickness” shows a more complex utterance with a tree depth of 8. We also show the most common top-level constituent combinations in Figure 5 in the Appendix.

## 6 Methods

To study the feasibility of developing conversational plotting agent using CHARTDIALOGS, we assess the performance of various methods.

The main methods we evaluate build on the sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014; Vinyals and Le, 2015).

Seq2seq models employ two components: an encoder and a decoder. The encoder produces hidden states of the input. Attention is used to produce a weighted sum of the encoder hidden states, known as the *context vector*  $c_t^*$ . The decoder defines the joint probability of an output sequence  $\mathbf{y} = (y_1, \dots, y_{n_y})$  as:

$$p(\mathbf{y}) = \prod_{t=1}^T p(y_t \mid \{y_1, \dots, y_{t-1}\}, c_t^*)$$

**Input.** We treat each plot update as a separate datapoint. For each datapoint, the input comes from three available sources: i) current state as represented by the text plot specification (TPSpec), ii) current state as represented by the plot image, and iii) the dialog history. In principle, the entire dialog history can be considered. In our experiments, we consider all utterances from the last plot update to the current one from both interlocutors. In other words, starting from the last plot update, the Describer’s instruction and all the clarification questions and responses are concatenated and provided as the dialog history.

**Output.** We formulate the model output as the update needed from the current TPSpec to the next TPSpec. We denote such an update as  $\Delta TPSpec$ . For example, if the current TPSpec is  $\{('line\_width': 'thin'), ('line\_color': 'black')\}$  and the next TPSpec is  $\{('line\_width': 'thin'), ('line\_color': 'red')\}$ , the corresponding  $\Delta TPSpec$  is  $\{('line\_color': 'red')\}$ . As discussed below, the output module can be a sequence decoder, in which the  $\Delta TPSpec$  is predicted as a sequence; or a set of classifiers, each of which predicts the new value of a different slot.

**[M1] S2S-PLOT+TXT.** The first method is a seq2seq method whose input consists of the current state as represented by both TPSpec and plot image, and the dialog history. The TPSpec and dialog history are concatenated and fed to a seq2seq model. For all methods involving a seq2seq model, we use a 2-layer Bi-LSTM for the text encoder and another 2-layer Bi-LSTM for the decoder. To encode the plot image, we used a CNN followed by a row-wise LSTM. The final representation of an image is a sequence of vectors and are concatenated with the text representations on the temporal dimension before they are fed to the decoder. More details are provided in Appendix B.

**[M2] S2S-TXT.** The second method is another seq2seq model, but we omit the plot image from the

input. We consider this version in order to assess the role of the vision modality in the task.

**[M3] S2S-NoState.** This is a seq2seq model whose input consists only of the dialog history. The state in the form of current TPSpec or plot image is completely omitted. The goal is to assess if the state is actually taken into account by the model.

**[M4] S2S-NoUtterance.** This is a seq2seq model whose input consists only of the current state as represented by TPSpec. The dialog history is completely omitted. The goal is to assess if the dialog history is actually taken into account by the model.

**[M5] MaxEnt.** We trained a logistic regression classifier to take as input the TPSpec and dialog history. They are represented jointly as bag-of-words. Classification predictions are made for each slot separately. For each slot, the candidate label space is all possible labels that appeared in our dataset, along with a special label *[unchanged]* indicating not to change the value of this slot, i.e. using the value from current state. Notice that bag-of-words features have a critical problem of ignoring word ordering. For example, it cannot distinguish between “red line with blue markers” and “blue line with red markers”.

**[M6] RNN + MLP.** This model is similar to MaxEnt except that features are extracted by an LSTM encoder, which considers word ordering. It differs from the seq2seq models in that the prediction is made with MLP classifier heads for each slot separately, instead of an LSTM decoder for the whole output. This exempts the model from the burden of generating a structured sequence; on the other hand the model is no longer equipped to learn the dependencies between different slots. We use a 2-layer Bi-LSTM encoder for the input representation. Each MLP consists of 2 fully-connected layers.

**[M7] Transformer + MLP.** We consider another alternative where instead of an RNN, we use a transformer encoder, in particular, BERT (Devlin et al., 2019). The final layer output of the special BERT token “[CLS]” is used as the input representation and fed to MLP classifier heads. The structure of MLP classifier heads is the same as in RNN+MLP.

## 7 Experiments

We conducted experiments for the following purposes: **(P1)** to evaluate the performance of the above-mentioned methods; **(P2)** to establish the quality of our dataset; and **(P3)** to establish a gold

Methods	SPLIT	SINGLE	PAIR
S2S-PLOT+TXT	0.585	<b>0.613</b>	0.594
S2S-TXT	0.601	<b>0.613</b>	0.591
S2S-NoState	0.525	0.549	0.535
S2S-NoUtterance	0.060	0.047	0.046
MaxEnt	0.196	0.265	0.422
RNN+MLP	0.328	0.324	0.325
Transformer+MLP	0.311	n/a <sup>6</sup>	n/a <sup>6</sup>

Table 2: Exact match plotting performance.

Methods	SPLIT	SINGLE	PAIR
S2S-PLOT+TXT	0.871	<b>0.890</b>	0.888
S2S-TXT	0.874	<b>0.893</b>	0.885
S2S-NoState	0.847	0.866	0.863
S2S-NoUtterance	0.316	0.306	0.155
MaxEnt	0.677	0.734	0.806
RNN+MLP	0.714	0.712	0.724
Transformer+MLP	0.723	n/a <sup>6</sup>	n/a <sup>6</sup>

Table 3: Slot change F1 plotting performance.

human performance as the upper bound of expected model performance.

## 7.1 Experimental Setup

**Train, Dev, and Test Splits.** We used 2,628 dialogs for training, 328 for validation and 329 for testing. In terms of datapoints, there are 11,903 for training, 1,562 for validation and 1,481 for testing.

**Token Granularity for Prediction.** We consider three different token granularity settings for mapping between TPSpecs and actual token sequences on both the input and output side: PAIR, SINGLE and SPLIT. In the PAIR strategy, the token for the slot name and slot value are concatenated to create one single token of the form: “slot\_name:slot\_value”. In SINGLE, each slot name and slot value is predicted independently. In SPLIT, slot and value names are split into actual words. For example, predicting that the slot “x\_axis\_scale” takes on the value “log” under the PAIR strategy involves one prediction, “x\_axis\_scale:log”. Under SINGLE, this involves two predictions, “x\_axis\_scale” and then “log”. Under SPLIT, the expected prediction becomes “x”, “axis”, “scale”, “:” and “log”.

<sup>6</sup>Due to the BPE encoding used in BERT, SINGLE and PAIR inputs are tokenized to be almost identical as SPLIT, therefore we do not report their performance.

## 7.2 Evaluation Metrics

We evaluate performance using two metrics: *Exact Match (EM)* and *Slot change F1*. Exact Match measures how accurate the models are at updating the plots exactly as expected. It is defined as the percentage of datapoints whose current TPSpec, when updated with the model-predicted  $\Delta$ TPSpec, can exactly match the gold target TPSpec. Slot change F1 measures accuracy on individual slots. Let  $S_p$  be the set of slot-value pairs in the predicted  $\Delta$ TPSpec and  $S_g$  be the set of slot-value pairs in the gold  $\Delta$ TPSpec, precision  $P = \frac{|S_p \cap S_g|}{|S_p|}$ , recall  $R = \frac{|S_p \cap S_g|}{|S_g|}$  and  $F1 = \frac{2PR}{P+R}$ .

## 7.3 Performance (P1)

We report Exact Match performance in Table 2, and Slot change F1 in Table 3. From the tables, it is clear that seq2seq-based models generally perform better than classification models. A possible reason is that, by modeling  $\Delta$ TPSpec as a whole in the decoder, the models implicitly learned dependencies between different slots and thus improved the overall performance. Also, neural classification methods including RNN+MLP and Transformer+MLP displayed poor performance, not even beating MaxEnt with bag-of-words. Further, as an ablation study, the S2S-NoState and S2S-NoUtterance performed significantly worse than S2S-TXT, confirming that both the current state and the user utterance are necessary to seq2seq methods in performing this task.

Both S2S-TXT and S2S-PLOT+TXT perform the best at the SINGLE token granularity. On this granularity, there is no significant difference between their performance on exact match. For slot F1, S2S-TXT even performs significantly better than S2S-PLOT+TXT, with  $p = 0.033$  in an unequal variance T-test, which implies that for seq2seq methods adding the image modality does not add much on top of the text modality in this task.

Table 4 shows performance of the best performing methods, S2S-PLOT+TXT and S2S-TXT, per plot type. We ran 5 experiments and reported the means and standard deviations in order to gain a better comparison between their performances. We can see that, as expected for our above results, for most plot types, performance of the two methods is similar.

Plot type	S2S-TXT		S2S-PLOT+TXT	
	Exact Match	Slot F1	Exact Match	Slot F1
Line	0.602±0.026	0.889±0.005	0.605±0.011	0.888±0.006
Bar	0.572±0.022	0.873±0.004	0.565±0.020	0.866±0.004
Pie	0.685±0.009	0.896±0.005	0.691±0.005	0.894±0.008
Contour	0.618±0.006	0.916±0.004	0.624±0.015	0.913±0.004
Streamline	0.610±0.016	0.901±0.009	0.598±0.023	0.895±0.007
Histogram	0.476±0.048	0.886±0.007	0.505±0.026	0.890±0.019
Scatter	0.492±0.026	0.849±0.014	0.492±0.017	0.851±0.014
Matrix	0.717±0.022	0.944±0.006	0.683±0.033	0.939±0.004
3D Surface	0.733±0.047	0.910±0.023	0.768±0.041	0.928±0.023
<b>Total</b>	0.613±0.005	0.893±0.002	0.613±0.005	0.890±0.002

Table 4: Exact match and Slot F1 score by plot type, under the SINGLE granularity.

#### 7.4 Agreement Among Workers (P2)

In order to further inspect the quality and difficulty of our dataset, we sampled a subset of 444 partial dialogs. Each partial dialog consists of the first several turns of a dialog, and ends with a Describer utterance. The corresponding Operator response (plot update) is omitted. Thus, the human has to predict what the Operator (the plotting agent) will plot, given this partial dialog. We created a new MTurk task, where we presented each partial dialog to 3 workers and collected their responses. We calculated the agreements between the newly collected responses and the original Operator response, results shown in Table 5.

The cases in which the majority of the workers (3/3 or 2/3) exactly match the original Operator, corresponding to the first two rows, happen **72.6%** of the time. The cases when at least 3 out of all 4 humans (including the original Operator) agree, corresponding to row 1, 2 and 5, happen **80.6%** of the time. This setting is also worth considering because the original Operator is another MTurk worker, who can also make mistakes. Both of these numbers show that a large fraction of the utterances in our dataset are intelligible implying an overall good quality dataset.

Fleiss’ Kappa among all 4 humans is 0.849; Cohen’s Kappa between the original Operator and the majority among 3 new workers is 0.889. These numbers indicate a strong agreement as well.

#### 7.5 Models vs. Gold Human Performance (P3)

The gold human performance was obtained by having one of the authors perform the same task as described in the previous subsection, on a subset

Original	New	Proportion
√	√√√	55.1%
√	√√×	17.5%
√	√××	2.4%
√	×××	0.0%
×	√√√	8.0%
×	√√×	10.3%
×	√××	4.4%
×	×××	2.3%
<b>Total</b>		100.0%

Table 5: Agreement evaluation result. √ stands for “exact match with majority” and × for “no exact match with majority”. The majority is obtained slot-wise, i.e. the majority for each slot is obtained separately.

of 180 samples. The result is a **76.8%** exact match. That is, our best model is **15.5** percentage points behind gold human performance, showing there is room for models to improve on this dataset.

#### 7.6 Comparison to Performance on Image Editing

The best accuracy reported on the aforementioned conversational image editing dataset was 74% on intent classification, ignoring actual attribute values (Manuvinakurike et al., 2018a). This result is not directly comparable to the best accuracy 61.3% on our dataset due to the difference in accuracy definition. To our knowledge, no comparable results has been reported on the image editing dataset, and the dataset is not publicly available.

### 8 Error Analysis

We inspected the output of our best-performing models in order to identify the most common



causes of errors. Here we used S2S-TXT with SINGLE granularity as a representative; the error categories are similar for S2S-PLOT+TXT or other granularity.

### 8.1 Ambiguity

Sometimes the Describer utterance is ambiguous and makes different actions all reasonable. We spotted two kinds of ambiguities: the *unspecified new slot* and the *value*, exemplified in Table 6a and 6b respectively. **1) Unspecified new slot.** The Describer added a new component to the plot (the grid lines), which activated new slots (“grid\_line\_type”) whose values are unspecified. Therefore, any value for these slots should be correct. **2) Ambiguous value.** The Describer asked to change the size of a component (the font), but did not specify the value. As in the example, the font size was “large”; to make it “smaller”, both “medium” and “small” are correct.

### 8.2 Human Errors

We report some of the errors that are due to mistakes made by MTurk workers. Operators can overlook part of the Describer’s instruction. These erroneous actions are recorded and in turn be counted as errors of models in our automatic evaluation process.

### 8.3 Model Errors

In addition to human errors, many cases were also due to the model itself. We show examples of model errors in Table 7. **1) Multi-turn dialog history.** In most samples, the dialog history consists of only one utterance, the Describer’s instruction. As a result, when confronted with multiple utterances concatenated, the model may get confused. **2) Complex slot value.** Some slot values are relatively hard to describe in natural language, such as “colormap” in example 7b. They can cause the models to make mistakes. **3) Infrequent expressions.** When the user expresses their request in an unusual way (in example 7c, “log style” for log scale), the model may not understand since it is rarely seen in the training data.

## 9 Conclusions

In this paper, we defined the problem of conversational plotting agents, which is of great practical

<sup>7</sup>Interlocutor signs are shown only for clarity; they are not input for models.

<b>Previous State</b>	(no grid lines)
<b>Dialog History</b>	invert y axis , red dashed <b>gridlines</b> , markers should be down triangle
<b>Gold Output</b>	grid_line_type horizontal
<b>Model Output</b>	grid_line_type both

(a) Ambiguity: unspecified new slot

<b>Previous State</b>	font_size large
<b>Dialog History</b>	make font size <b>smaller</b> again , sorry
<b>Gold Output</b>	font_size medium
<b>Model Output</b>	font_size small

(b) Ambiguity: ambiguous value

Table 6: Examples of different kinds of ambiguities.

<b>Previous State</b>	line_style dotted
<b>Dialog History</b>	[Desc] <sup>7</sup> dot line dot line [Op] this is dot , do you mean dot-dash ? [Desc] that would be it ... sorry
<b>Gold Output</b>	line_style dashed_dots
<b>Model Output</b>	line_style dotted

(a) Error: multi-turn dialog history

<b>Previous State</b>	(empty plot)
<b>Dialog History</b>	matrix display , <b>yellow to red</b> , x axis inverted on top , y axis inverted on right
<b>Gold Output</b>	color_map transparent_yellow_to_solid_red
<b>Model Output</b>	color_map red_to_yellow

(b) Error: complex slot value

<b>Previous State</b>	(empty plot)
<b>Dialog History</b>	hello , we have a bar plot ... orange bars with a black outline , <b>log style</b> please
<b>Gold Output</b>	y_axis_scale log
<b>Model Output</b>	y_axis_scale linear

(c) Error: infrequent expression

Table 7: Examples of different kinds of model errors.

importance considering the large volume of questions online about plotting library usage. We also presented a dataset, CHARTDIALOGS, to facilitate the development of such agents. Our experiments have demonstrated the feasibility of seq2seq-based methods to produce working models for dataset; however, there is still a large gap between our best performing methods and human performance.

Future work includes methods that get closer to human performance on the dataset. A practical line of future work is embedding our plotting agent in interactive environments such as Jupyter Lab.

## Acknowledgments

This work was partially supported by a Hellman Fellowship. We also appreciate constructive feedback from our anonymous reviewers.

## References

- Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219.
- Christina L. Bennett and Alexander I. Rudnicky. 2002. The carnegie mellon communicator corpus. In *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*.
- Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.
- Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.
- A Church. 1957. Applications of recursive arithmetic to the problem of circuit synthesis—summaries of talks. *Institute for Symbolic Logic, Cornell University*.
- 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Vasant Dhar. 2013. Data science and prediction. *Commun. ACM*, 56(12):64–73.
- Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 37–49.
- Mihail Eric and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *SIGDIAL Conference*.
- Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. 2015. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 489–500. ACM.
- Milica Gašić, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. In *Fifteenth Annual Conference of the International Speech Communication Association*.
- Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.
- Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 263–272.
- Anthony JG Hey, Stewart Tansley, Kristin M Tolle, et al. 2009. *The fourth paradigm: data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA.
- Julia Hirschberg and Christopher D Manning. 2015. Advances in natural language processing. *Science*, 349(6245):261–266.
- Min-Yen Kan, Xiangnan He, Wenqiang Lei, Xisen Jin, Zhaochun Ren, and Dawei Yin. 2018. Seqicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *ACL*.
- J. F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. Inf. Syst.*, 2(1):26–41.
- Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. [OpenNMT: Open-source toolkit for neural machine translation](#). In *Proc. ACL*.
- Fei Li and Hosagrahar Visvesvaraya Jagadish. 2014. Nalir: an interactive natural language interface for querying relational databases. In *International Conference on Management of Data, SIGMOD*, pages 709–712.
- Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 285–294.
- Ramesh Manuvinarake, Trung Bui, Walter Chang, and Kallirroi Georgila. 2018a. Conversational image editing: Incremental intent identification in a new dialogue task. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 284–295.

- Ramesh R. Manuvinakurike, Jacqueline Brixey, Trung Bui, Walter Chang, Doo Soon Kim, Ron Artstein, and Kallirroi Georgila. 2018b. Edit me: A corpus and a framework for understanding natural language image editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.
- A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Borde, D. Parikh, and J. Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).
- Antoine Raux, Brian Langner, Dan Bohus, Alan W. Black, and Maxine Eskénazi. 2005. Let’s go public! taking a spoken dialog system to the real world. In *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, September 4-8, 2005*, pages 885–888.
- Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 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, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 438–449.
- Stephanie Seneff and Joseph Polifroni. 2000. Dialogue management in the mercury flight reservation system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 11–16. Association for Computational Linguistics.
- Vidya Setlur, Sarah E Battersby, Melanie Tory, Rich Gossweiler, and Angel X Chang. 2016. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 365–377. ACM.
- Pararth Shah, Dilek Z. Hakkani-Tür, Bing Liu, and Gökhan Tür. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *NAACL-HLT*.
- Armando Solar-Lezama and Rastislav Bodik. 2008. *Program synthesis by sketching*. Citeseer.
- Arjun Srinivasan and John Stasko. 2017. Natural language interfaces for data analysis with visualization: Considering what has and could be asked. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pages 55–59. Eurographics Association.
- Yiwen Sun, Jason Leigh, Andrew E. Johnson, and Sangyoon Lee. 2010. Articulate: A semi-automated model for translating natural language queries into meaningful visualizations. In *Smart Graphics*.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.
- Wei Wei, Quoc V. Le, Andrew M. Dai, and Jia Li. 2018. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854.
- Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.
- Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference, The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 22-24 August 2013, SUPELEC, Metz, France*, pages 404–413.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.
- Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *PACMPL*, 1(OOPSLA):63:1–63:26.
- Bowen Yu and Cláudio T Silva. 2019. Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE transactions on visualization and computer graphics*.
- John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

## Appendix

### A Plot Types and Slots

We show all plot types and slots related to each type in Table 8. All the plot types and slots are illustrated in Figure 6.

### B Model Implementation Details

Model implementations are based on OpenNMT (Klein et al., 2017) and HuggingFace Transformers (Wolf et al., 2019).

#### B.1 S2S-TXT

LSTM hidden size is 128, batch size is 16. Model is trained for 100,000 steps. Learning rate is initialized to 1.0; starting from step 50,000, the learning rate is halved every 10,000 steps.

#### B.2 S2S-PLOT+TXT

Text Encoder and Decoder have the same configuration as in S2S-TXT. The Plot (image) Encoder is a CNN with following layers: conv1 (64x3x3) - pooling1 (2x2) - conv2 (128x3x3) - pooling2 (2x2) - conv3 (128x3x3) - batch normalization3 - conv4 (256x3x3) - pooling4 (2x1) - conv5 (256x3x3) - batch normalization5 - pooling5 (1x2) - conv6 (256x3x3) - pooling6 (5x5). The output size from CNN is original image size reduced 40 times on both height and width. After CNN, a row-wise RNN is applied and the output for each row are concatenated to form the plot image encoding.<sup>8</sup> The learning rate scheme is the same as in S2S-TXT.

#### B.3 RNN+MLP

LSTM hidden size is 64, batch size is 32. Each MLP head has 2 layers, mapping from 128 (LSTM cell and output concatenated) to 32 and from 32 to number of classes. Model is trained for 100,000 steps. Learning rate is initialized to 1.0; starting from step 50,000, the learning rate is halved every 10,000 steps.

#### B.4 Transformer+MLP

The version of pretrained BERT we used is bert-base-uncased. It is fine-tuned with our classification heads. Batch size is 8 and gradient is accumulated over every 4 steps. Each MLP head has only 1 layer, mapping from BERT hidden size (768) to

number of classes. Learning rate is  $2e-5$ . Model is trained for 30 epochs.

### C Amazon Mechanical Turk HIT Screenshots

We show several screenshots of our HIT in Figure 7 and 8.

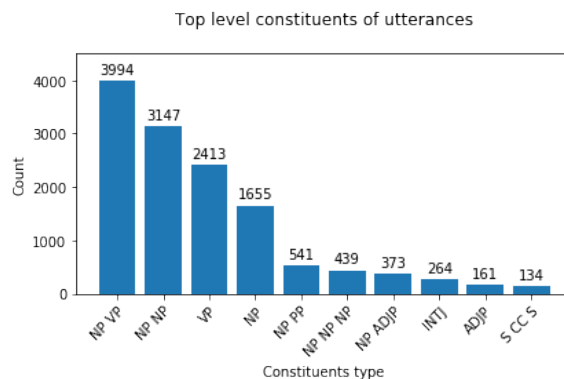


Figure 5: Most common top-level constituent combinations and their proportions (punctuations ignored).

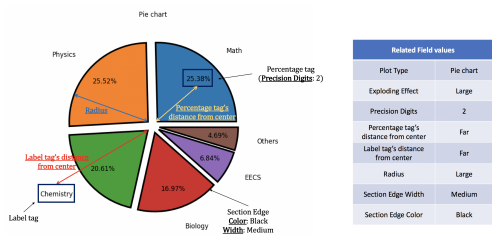
<sup>8</sup>This model structure is adapted from OpenNMT.



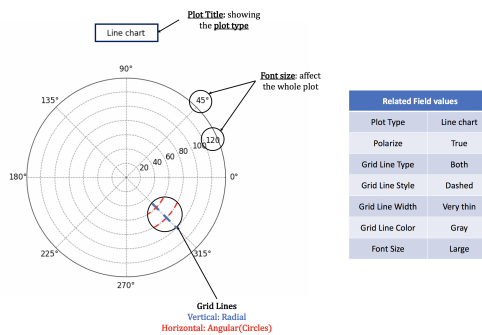
	Plot Types	Slots
1.	Axes	Polarize, X-axis Scale, Y-axis Scale, X-axis Position, Y-axis Position, Invert X-axis, Invert Y-axis, Grid Line Type, Grid Line Style, Grid Line Width, Grid Line Color, Font Size
2.	3D Surface	Color map, Invert X-axis, Invert Y-Axis, Invert Z-Axis
3.	Bar Chart	Bar Orientation, Bar Height, Bar Face Color, Bar Edge Width, Bar Edge Color, Show Error Bar, Error Bar Color, Error Bar Cap Size, Error Bar, Cap Thickness, Data Series Name
4.	Contour/Filled	Contour Plot Type, Number of levels, Color Map, Color Bar Orientation, Color Bar Length, Color Bar Thickness
5.	Contour/Lined	Contour Plot Type, Lined Style, Line Width
6.	Histogram	Number of Bins, Bar Relative Width, Bar Face Color, Bar Edge Width, Bar Edge Color, Data Series Name
7.	Matrix	Color Map, Invert X-axis, Invert Y-axis
8.	Line Chart	Line Style, Line Width, Line Color, Marker Type, Marker Size, Marker Face Color, Marker Edge Color, Marker Interval, Data Series Name, Show Error Bar, Error Bar Color, Error Bar Cap Size, Error Bar Cap Thickness
9.	Pie Chart	Exploding Effect, Precision Digits, Percentage tags' distance from center, Label tag's distance from center, Radius, Section Edge Width, Section Edge Color
10.	Polar	Polarize, Grid Line Type, Grid Line Style, Grid Line Width, Grid Line Color, Font Size
11.	Scatter	Polarize, Marker Type, Marker Size, Marker Face Color, Marker Edge Width, Marker Edge Color, Color Map, Color Bar Orientation, Color Bar Length Color Bar Thickness
12.	Streamline	Density, Line Width, Line Color, Color Map, Arrow Size, Arrow Style

Table 8: Plot types and slots in our dataset

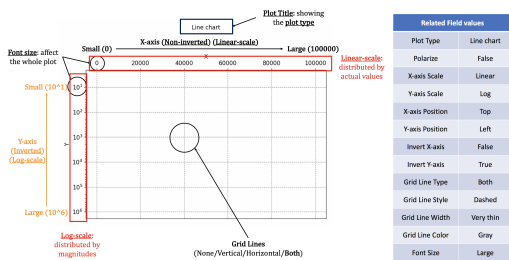
**Pie Chart Illustration**



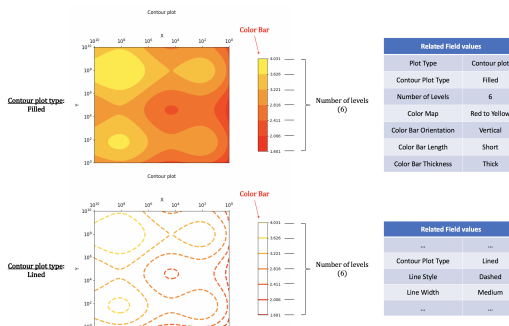
**Basic Polar Illustration**



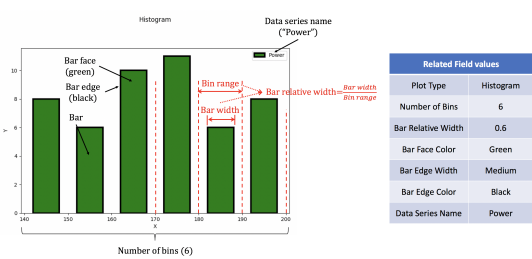
**Basic Axis Illustration**



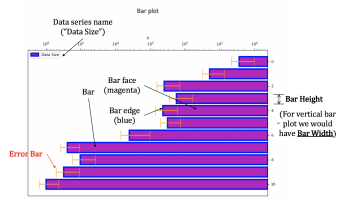
**Contour Plot Illustration**



**Histogram Illustration**

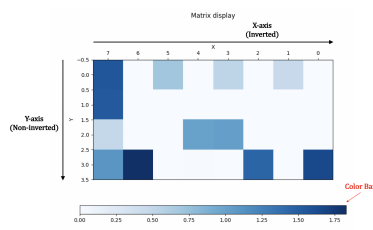


### Bar Plot Illustration



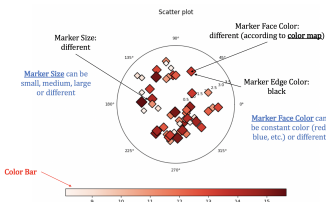
Related Field values	
Plot Type	Bar plot
Bar Orientation	Horizontal
Bar Height	0.8
Bar Face Color	Magenta
Bar Edge Width	Medium
Bar Edge Color	Blue
Show Error Bar	True
Error Bar Color	Orange
Error Bar Cap Size	Large
Error Bar Cap Thickness	Thin
Data Series Name	Data Size

### Matrix Display Illustration



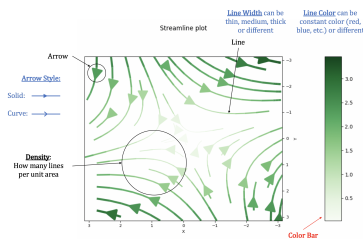
Related Field values	
Plot Type	3D surface
Color Map	Transparent to Solid Blue
Invert X-axis	True
Invert Y-axis	False

### Scatter Plot Illustration



Related Field values	
Plot Type	Scatter plot
Polarize	True
Marker Type	Diamond
Marker Size	Different
Marker Face Color	Different
Marker Edge Color	Black
Marker Face Width	Thin
Marker Edge Color	Black
Color Map	Transparent to Solid Red
Color Bar Orientation	Horizontal
Color Bar Length	Short
Color Bar Thickness	Thin

### Streamline Plot Illustration



Related Field values	
Plot Type	Streamline plot
Density	Medium
Line Width	Different
Line Color	Different
Color Map	Transparent to Solid Green
Arrow Size	Large
Arrow Style	Solid

### Color Bar Illustration



### Error Bar Illustration

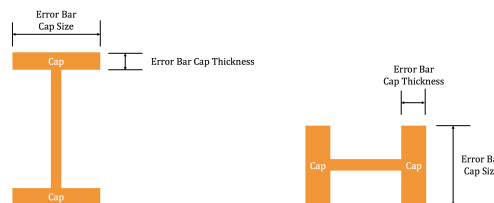


Figure 6: Plot types and slots.

## Plotting Agent - Describer

Concepts Illustrations | Colors

### Describer

You will see the target plot and the Operator's current plot. You need to describe the target plot in **clear and fluent natural language** so that the Operator can draw it exactly. We suggest you first describe the **plot type** and then any other features, since Operator needs it to activate other fields.

For example, you might say "It's a line chart with a dashed red line and blue circle markers" or "It's a histogram with 10 bins"; if the Operator's plot does not match what you want, you can try to adjust it, such as saying "Please make the marker larger", etc.

**When you encounter any difficulties, please check out the [Concepts Illustrations](#) link above and the [Tips](#) section below!**

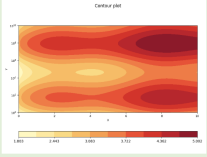
**Notice:**

1. Do not use the forward, backward or refresh button of your browser.


Reminder - please do not write anything that involves any level of discrimination, racism, sexism and offensive religious/politics comments, otherwise the submission will be rejected.

**SYSTEM:** Welcome! Please start describing the plot to your partner.

**Target Plot**



**Current Plot**



(a) Describer: starting the HIT

## Plotting Agent - Operator

Concepts Illustrations | Colors

### Operator

You have an operation panel to control how to draw the plot. You need to understand Describer's message and draw the plot according to it. At your turn, you can modify fields in the operation panel and click "Plot". After that, the new plot will be shown at bottom of your window and also in Describer's window (so that he/she can see it). If you really can't understand Describer's instructions, you can ask questions like "What do you mean by XXX?" or "Do yo mean XXX?"

**When you encounter any difficulties, please check out the [Concepts Illustrations](#) link above and the [Tips](#) section below!**

**Notice:**

1. Do not use the forward, backward or refresh button of your browser.
2. Do not close the window during the HIT if you...

Reminder - please do not write anything that involves any level of discrimination, racism, sexism and offensive religious/politics comments, otherwise the submission will be rejected.

**SYSTEM:** Welcome! Please wait for your partner's description of the plot.

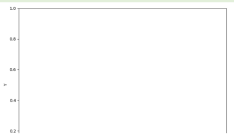
**Describer:** Hi! It's a filled contour plot with 10 levels, colored from light yellow to dark red

**Operation Panel**

**Plot Type**

(Please choose one)

**Current Plot**



(b) Operator: starting the HIT

Figure 7: HIT screenshots.

## Plotting Agent - Describer

Concepts Illustrations || Colors

### Describer

You will see the target plot and the Operator's current plot. You need to describe the target plot in **clear and fluent natural language** so that the Operator can draw it exactly. We suggest you first describe the **plot type** and then any other features, since Operator needs it to activate other fields.

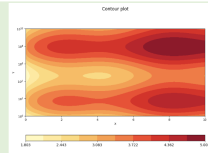
For example, you might say "It's a line chart with a dashed red line and blue circle markers" or "It's a histogram with 10 bins"; if the Operator's plot does not match what you want, you can try to adjust it, such as saying "Please make the marker larger", etc.

When you encounter any difficulties, please check out the [Concepts Illustrations](#) link above and the [Tips](#) section below!

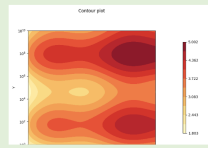
#### Notice:

- 1 Do not use the forward, backward or refresh

*Reminder - please do not write anything that involves any level of discrimination, racism, sexism and offensive religious/politics comments, otherwise the submission will be rejected.*



### Current Plot



Operator: [Plot Updated]  
 plot\_type: contour plot  
 contour\_plot\_type: filled  
 number\_of\_levels: 10  
 color\_map: transparent yellow to solid red  
 color\_bar\_orientation: vertical  
 color\_bar\_length: short  
 color\_bar\_thickness: thin  
 polarize: False  
 x\_axis\_scale: linear  
 y\_axis\_scale: linear  
 x\_axis\_position: bottom  
 y\_axis\_position: left  
 font\_size: medium  
 invert\_x\_axis: False  
 invert\_y\_axis: False

Describer: Y axis should be log-scale

Operator: [Plot Updated]  
 y\_axis\_scale: linear → log

Please enter here...

Send

(a) Describer: in progress

## Plotting Agent - Operator

Concepts Illustrations || Colors

### Operator

You have an operation panel to control how to draw the plot. You need to understand Describer's message and draw the plot according to it. At your turn, you can modify fields in the operation panel and click "Plot". After that, the new plot will be shown at bottom of your window and also in Describer's window (so that he/she can see it). If you really can't understand Describer's instructions, you can ask questions like "What do you mean by XXX?" or "Do yo mean XXX?"

When you encounter any difficulties, please check out the [Concepts Illustrations](#) link above and the [Tips](#) section below!

#### Notice:

- 1 Do not use the forward, backward or refresh button of your browser.
- 2 Do not close the window during the HIT if you...

*Reminder - please do not write anything that involves any level of discrimination, racism, sexism and offensive religious/politics comments, otherwise the submission will be rejected.*

### Operation Panel

#### Color Bar Orientation

Horizontal

#### Color Bar Length

Long

#### Color Bar Thickness

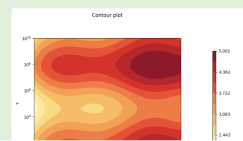
Medium

Polarize:

#### X-axis Scale

Plot

### Current Plot



color\_bar\_thickness: thin  
 polarize: False  
 x\_axis\_scale: linear  
 y\_axis\_scale: linear  
 x\_axis\_position: bottom  
 y\_axis\_position: left  
 font\_size: medium  
 invert\_x\_axis: False  
 invert\_y\_axis: False

Describer: Y axis should be log-scale

Operator: [Plot Updated]

Operator:  
 y\_axis\_scale: linear → log

Describer: The color bar is horizontal, long, medium width

Please enter here...

Send

(b) Operator: in progress

Figure 8: HIT screenshots.