Interactive Plot Manipulation using Natural Language

Yihan Wang, Yutong Shao and **Ndapa Nakashole**
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
yiw007@ucsd.edu,{yshao, nnakashole}@eng.ucsd.edu

Abstract

We present an interactive Plotting Agent, a system that enables users to directly manipulate plots using natural language instructions within an interactive programming environment. The Plotting Agent maps language to plot updates. We formulate this problem as a slot-based task-oriented dialog problem, which we tackle with a sequence-to-sequence model. This plotting model while accurate in most cases, still makes errors, therefore, the system allows a feedback mode, wherein the user is presented with a top-k list of plots, among which the user can pick the desired one. From this kind of feedback, we can then, in principle, continuously learn and improve the system. Given that plotting is widely used across data-driven fields, we believe our demonstration will be of interest to both practitioners such as data scientists broadly defined, and researchers interested in natural language interfaces.

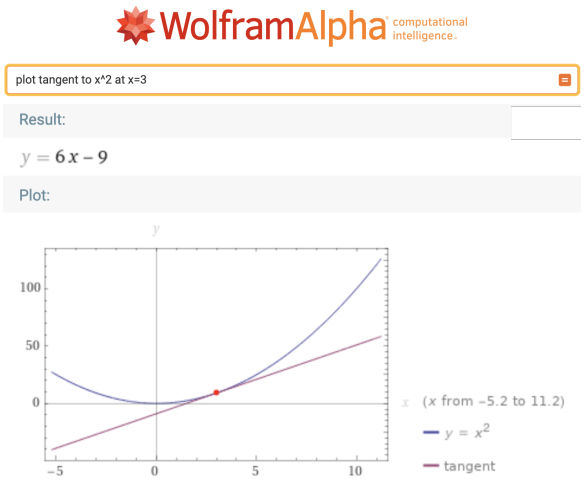

Figure 1: Related to our work is a commercial product, *wolframalpha.com*, which enables users to describe the *function* they would like to visualize, in this example, "plot the tangent to x^2 at $x = 3$". However, the user has no control over the plotting details. In contrast, we allow the user to use natural language to manipulate the plot.

1 Introduction

Motivation. Data can be utilized to improve outcomes in many sectors, for example healthcare, education, and business. However, when presented in its raw form, it is difficult to derive insights from data. Plotting is a simple yet powerful technique for making raw data readable, and exposing trends. Data plotting libraries, such as `matplotlib`, provide operations that enable users to visualize their data. Such libraries support functionalities at different levels, from high-level, "change the X-axis from *linear* to *log* scale"; to low-level "color this screen pixel red". However, novice users and expert programmers alike may still find it time-consuming to create plots of interest using these libraries. We therefore propose an interactive natural language interface (NLI) for plotting, that enables users to manipulate plots using natural language. The interactive aspect allows complex plotting needs to be specified and refined in multiple steps.

Prior Work. Previous work on NLIs for plotting focused on enabling users to describe the *data* or the *mathematical function* of interest. In contrast, our approach enables users to directly manipulate a *plot*. NLIs that focus on describing the *data* have emerged from Human Computer Interaction (HCI) and related areas (Gao et al., 2015; Setlur et al., 2016; Srinivasan and Stasko, 2017; Yu and Silva, 2019; Sun et al., 2010). Thus the user poses queries such as: "Show me medals for hockey and skating by country." or "Is there a seasonal trend for bike usage?". The system retrieves the relevant data, performs simple data analysis, and produces a visualization. Commercial products such as *wolframalpha.com* enable users to describe the *function*

Proceedings of NAACL-HLT 2021: Demonstrations, pages 92–98
June 6–11, 2021.

they would like to visualize. By leveraging knowledge of functions and mathematical procedures, the system produces meaningful results for queries such as: "plot the tangent to x^2 at $x = 3$", as shown in Figure 1.

Our work is also related to conversational image editing (Manuvinakurike et al., 2018b,a), which yields results for queries such as "Can you fix the glare on my dog's eyes". The key difference is that our images are plots, and thus the manipulations are different from those involving photo images.

Contributions and Demonstration Overview. We present a Plotting Agent for `matplotlib`, a popular Python plotting library. The Plotting Agent provides users with various ways to manipulate plots in an interactive programming environment, Jupyter Notebooks.

Our demonstration allows the user to explore the Plotting Agent in various ways: **(1)** Upload custom data and interactively manipulating plots on the uploaded data. **(2)** Work in a feedback mode, wherein the user is presented with a top-k list of plots, among which the user can pick the desired one. **(3)** Operate in batch mode where a series of instructions written in a file are loaded and executed sequentially by the system.

(4) Have a personalized experience, wherein the system learns user preferences, enabling them to perform certain tasks faster.

In addition to the novel functionality, we also build on ChartDialogs (Shao and Nakashole, 2020) to enable other functionality.

(5) Generate synthetic data using on pre-defined random data samplers from ChartDialogs , and interactively manipulate plots on the synthetic data.

(6) Load existing dialogs from the ChartDialogs dataset to observe the plot updates for each dialog turn. The user can make further changes to the plot.

In all the above cases, the user can use different lexical items and paraphrases to express the same intent. This demonstrates the advantage of our neural-based model compared to a system that might rely on rules and dictionaries. We have publicly released a live demo system[1] and a screencast showcasing the demo[2].

[1] https://github.com/Bawerlacher/Plotting_Agent

[2] https://youtu.be/a2D77JI7RVs

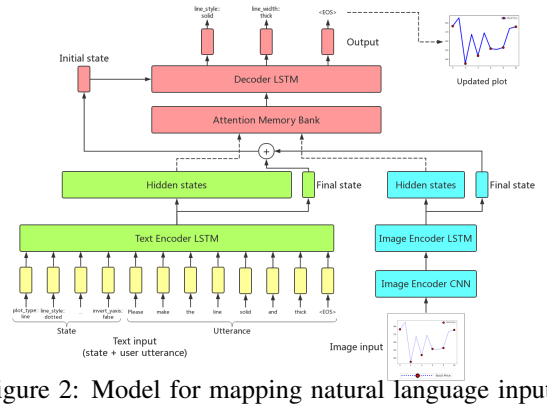

Figure 2: Model for mapping natural language inputs to plot updates.

2 The Plotting Agent System

We cast the Plotting Agent problem as a slot-based, task-oriented dialog problem. Each slot represents a plot property, such as *line color, marker size,* etc. Each plot type has different slots. However, some slots are shared and apply to multiple plot types. Consider the slot "X-axis scale", which takes the value "X-axis scale = log", from the request "change the x-axis scale from linear to log". Since the "X-axis scale" slot applies to the x-axis, it is applicable to any plot type with an x-axis, such as line chart, bar plot, or contour plot. Given the slots and their values, we can generate a plot image using `matplotlib`.

Model for Predicting Updates. Our model, depicted in Figure 2, for mapping natural language to plot slots and slot values builds on the sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014; Vinyals and Le, 2015). The input to the model is a natural language request, a text-representation of the current plot, the dialog history, formulated as a set of slot-value pairs, and the current plot image[3]. The dialog history consists of prior utterances, which are concatenated and treated as the dialog history.

The model outputs the update needed to go from the current set of slot-value pairs to the new slot-value pairs. For example, if the current slot-value pairs are {*('line_width': 'thin'), ('line_color': 'black')*} and the new slot-value pairs are {*('line_width': 'thin'), ('line_color': 'red')*} after the user utterance that says "change the line color to red", then the corresponding update

[3] Our experiments showed that incorporating the plot image did not improve model performance, thus plot images are omitted in the model we present in this demo.

Top-k	Exact Match
@1	0.61
@2	0.71
@3	0.74
@5	0.78
@10	0.78
@20	0.79

Table 1: Top-K exact match (EM) performance

(Δ) that the model must predict is {*('line_color': 'red')*}. We output decoder predicts Δ as a sequence.

Implementation and Training. The system is implemented in Python and makes use of the Pytorch library for neural network models. We use a 2-layer Bi-LSTM for the text encoder and another 2-layer Bi-LSTM for the decoder. We trained the model on the ChartDialogs (Shao and Nakashole, 2020) dataset which contains dialogs about plots. The dataset was generated by pairs of humans, where one human plays the role of the user, and the other plays the role of the agent. We use the train/dev/test split provided.

3 System Evaluation

Exact Match. Table 1 shows performance in terms of *Exact Match (EM)*, a measure that reflects how accurate the model is at updating the plots exactly as requested by the natural language utterance. We show performance at top-k ranked predictions, which are obtained from Beam Search. Beam Search keeps track of the k most probable partial predictions (hypotheses). A hypothesis has a score which is its log probability. As can be seen in Table 1, At $k = 1$, EM accuracy is only 61%. However, for $k = 5$, EM accuracy is much higher at 78%. We leveraged this fact, in the feedback mode of our demo, where instead of just showing the highest ranked plot, the top-k plots are shown, and the user selects the one that best corresponds to the intent of their utterance.

Runtime. Response times to utterances are on average 0.3s on modest hardware. This is much faster than using a Web search engine which might involve time consuming tasks such as refining the query multiple times and visiting community tutorial websites such as StackOverflow to search for similar questions that might have been answered.

4 Plotting Agent Demonstration

Interactive functionality is showcased within Jupyter Notebooks.

Data points and Instructions. To generate a plot, the following information must specified: the data points to plot and the slot-value assignments of the plot. The system therefore consists of two parts: data loading and instruction delivery. For data loading, the system supports uploading data files or randomly sampling synthetic data using our pre-defined data samplers. For instruction delivery, the system allows users to instruct the agent interactively or to load instructions from existing dialogs from our ChartDialogs dataset. In both cases, the natural language input and the current plot slot-value assignments (states) are fed into the back-end model to predict a set of slot-value pairs for plot updates.

4.1 Data Specification

Custom Data Upload. The user can upload a csv file containing the data to visualize. For "clean" csv files in which all the columns are to be plotted, e.g. using the first column for X axis, the second column for Y axis, etc., the data can be automatically loaded without further specifications. For more complex csv files, we also provide a more detailed and customized data specification process. If data loading is successful, the system will output "Data loaded from the csv file!" Figure 3 shows an example of uploading a csv file containing the cumulative COVID-19 daily confirmed cases in the United States until mid June 2020.

Synthetic Data. The user can generate data using our pre-defined data samplers for each plot type. After the user specifies the plot type, the system will invoke the corresponding data sampler to generate a group of data suitable for the given plot type. An example of plotting with generated data is shown in the Figure 4.

4.2 Plotting Intent Specification

Interactive Mode. The user can send instructions directly to the system using the interactive interface. There are two kinds of instructions: special commands and plot descriptions. Special commands are instructions that refer to specific system functionalities, such as "undo", "redo", "load csv", "plot", etc. For example, "plot" will show the plot

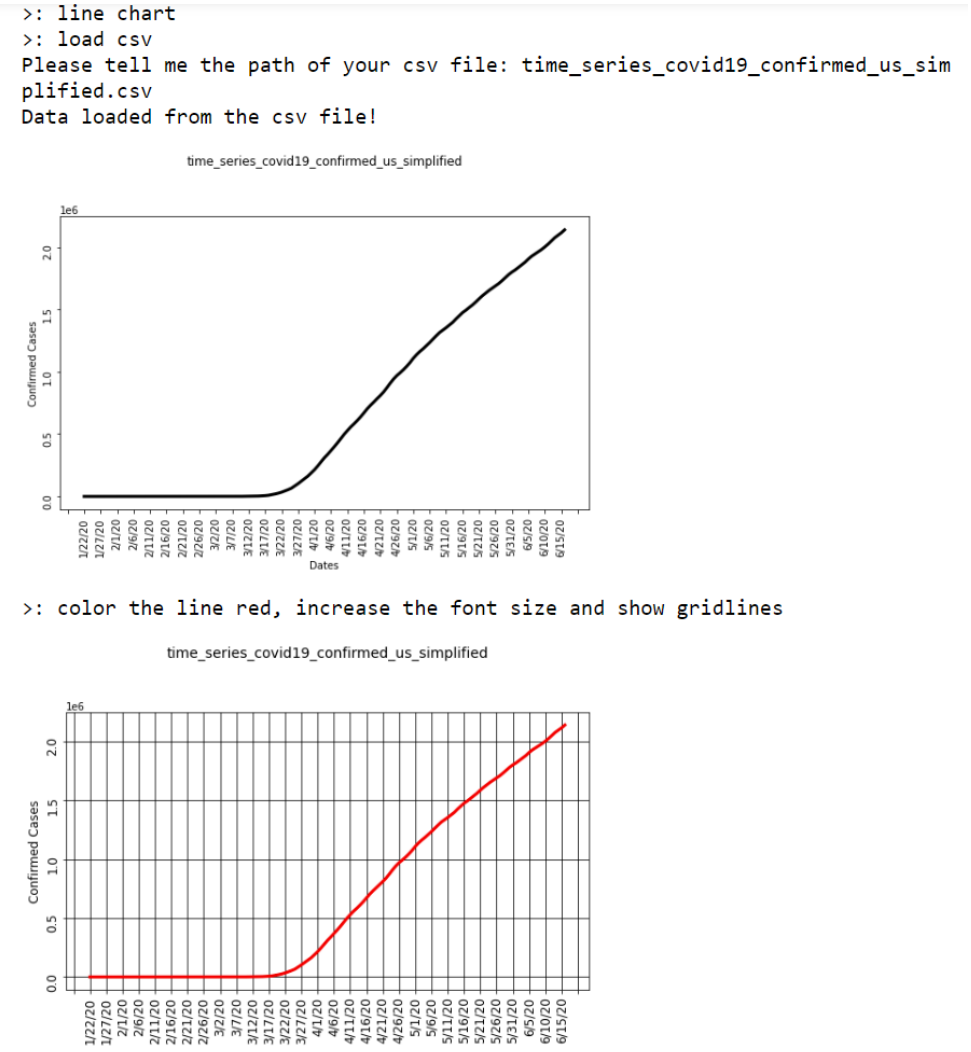

Figure 3: Custom Data with Interactive Instruction Delivery: upload of COVID-19 USA confirmed cases, and one instance of an interactive plot update.

image and "undo" will undo the last change to the plot.

Any other natural language instructions that are not in the special commands set are treated as plot descriptions. A plot description will be fed to the back-end model to predict the plot update, as described above. An example of interactive instruction delivery is shown on the COVID-19 US daily confirmed cases data, in Figure 3.

Prior Dialog Mode. In order to get a quick idea of how the system works, the user can choose to load a random dialog from the ChartDialogs dataset. The system treats the utterances in each dialog turn as a natural language instruction and predicts the plot update. The user chooses if they wish to view the updated plot step-by-step or only to obtain the final resulting plot. After the system processes all the dialog turns and shows the result, the user can continue to make further changes to the plot properties through natural language instructions. An example of this use case is shown in Figure 5.

Batch Mode. The user can also write their own instructions in a file and send the file path to the system. The system will read the instructions one by one, update the plot correspondingly, and show the final result.

4.3 User Feedback

Our plotting model while accurate inmost cases, still makes errors, therefore, the system allows a feedback mode, wherein the user is presented with a top-k list of plots, among which the user can pick the desired one. The top-k results are obtained from Beam search used in our decoder in the architecture shown in 2. At each step of the decoder, Beam

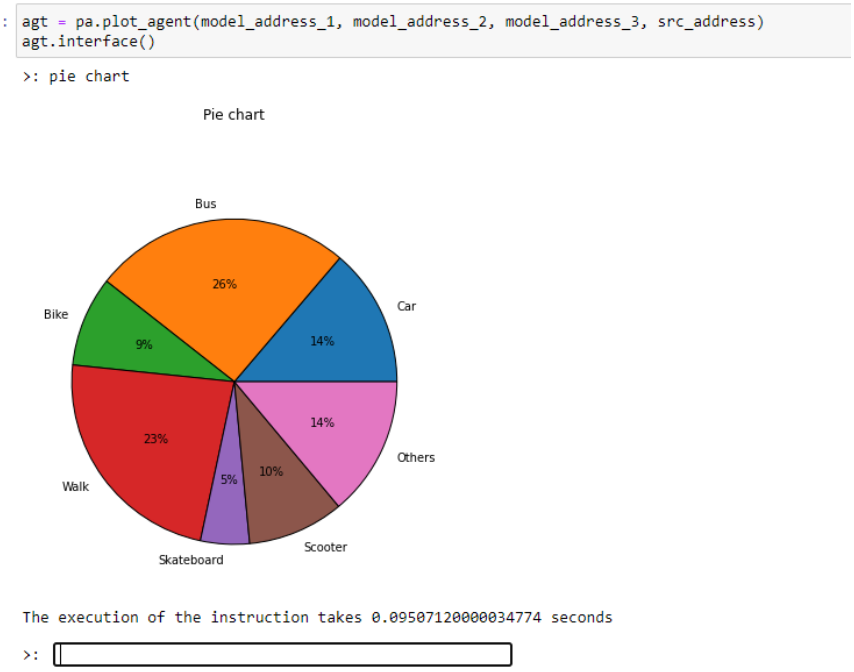

Figure 4: Synthetic Data: example plot of type pie chart generated from our pre-defined data samplers.

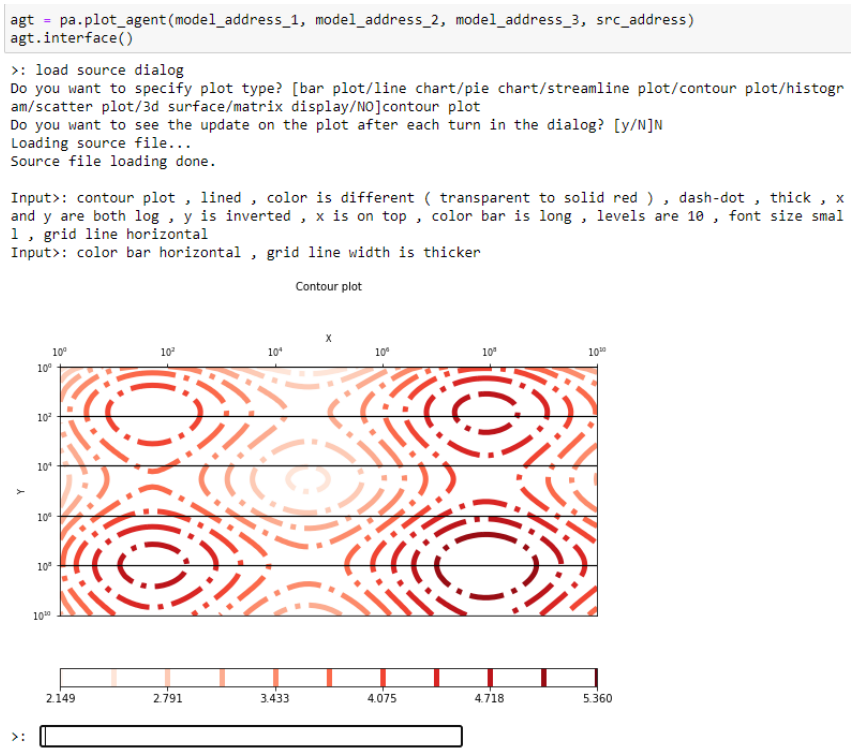

Figure 5: Prior Dialog: an example of executing a dialog taken from the ChartDialogs dataset.

search keeps track of the k most probable partial results, where k is the beam size. As shown in Table 1, the higher the value of k, the more likely it is to have the correct plot presented to the user. From this kind of feedback, we can then, in principle, continuously learn and improve the system

4.4 Personalization

A useful system should be personalized to an individual user. It should adapt to their unique goals, context, or nuances of the types of visualizations they like to produce. For example, the user can request "change the font size to 16, and the line

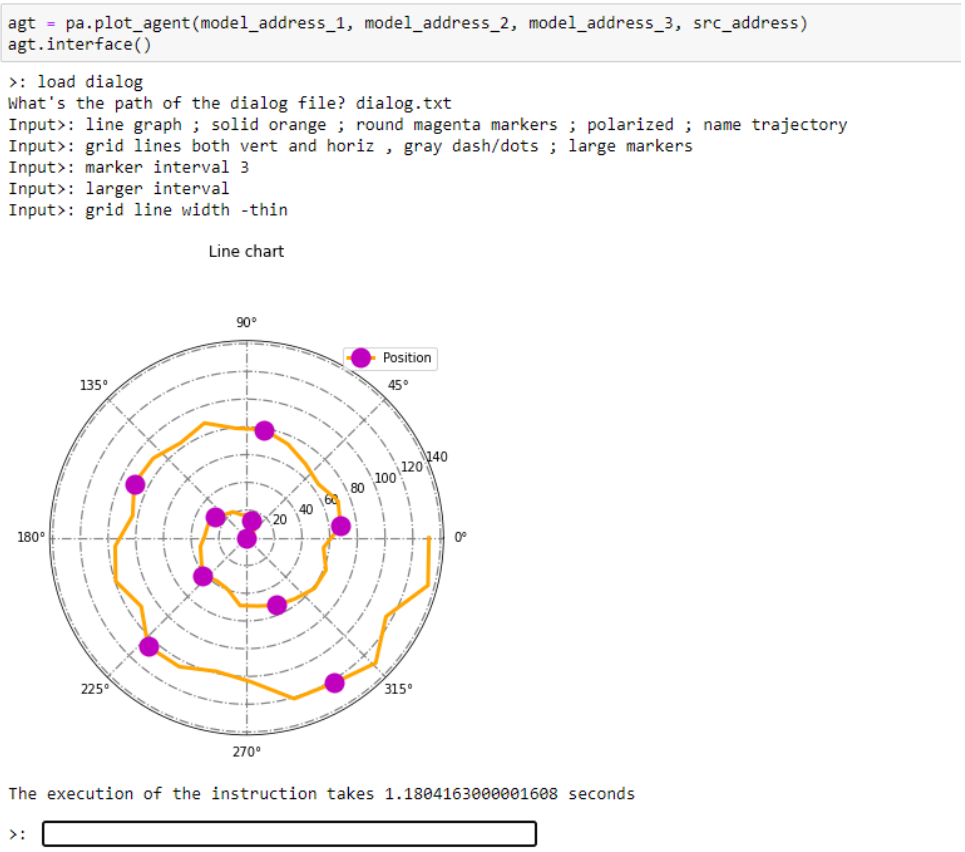

Figure 6: Batch mode: an example of loading instructions from a file.

color to cyan". But in a personalized form, the user may simply state "fix the fonts and colors", in which case the agent relies on the user prior preferences. Our demo includes a basic notion of personalization.

5 Discussion

Primitive to Complex Plotting Intents. Interactions with our current plotting agent are limited to manipulating slots preprogrammed by the API developers of the plotting library, such as changing the font size or the color of a particular item. Our goal is to expand commands understood by the plotting agent to include complex slots beyond the API slots (e.g., by teaching the system to "shift the legend so that it does not obscure important parts of the plot" or "make the text labels of a scatter plot to be aesthatically optimized", or "change colors to be colorblind friendly".).

Limitations of Slot-based Representation. The current demo system also has limitations due to design choices guided by the goal of task simplification. For example, some plot components that are not easily formulated as slot-value pairs are not supported. This is a research question of representation, while the slot based representation facilitates quick learning, more expressive representations can resolve these limitations.

6 Conclusion

In this paper, we introduced an interactive Plotting Agent for mapping natural language instructions to plot updates. The system supports various modes for specifying data and instructing the agent to update plots. Our interactive Plotting Agent is under further research and development to improve its language understanding capabilities, and to expand its functionality to other plot components, and plotting libraries. We hope the demo will be of interest to both practitioners such as data scientists, and researchers interested in natural language interfaces.

References

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.

Ramesh Manuvinakurike, 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.*

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.

Yutong Shao and Ndapa Nakashole. 2020. ChartDialogs: Plotting from Natural Language Instructions. In *ACL*. Association for Computational Linguistics.

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*.

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*.