Plots Made Quickly: An Efficient Approach for Generating Visualizations from Natural Language Queries

Henrik Voigt[1], Kai Lawonn[1], Sina Zarrieß[2]
University of Jena[1], Bielefeld University[2]
[1]first.last@uni-jena.de
[2]first.last@uni-bielefeld.de

Abstract

Generating visualizations from natural language queries is a useful extension to visualization libraries such as Vega-Lite. The goal of the NL2VIS task is to generate a valid Vega-Lite specification from a data frame and a natural language query as input, which can then be rendered as a visualization. To enable real-time interaction with the data, small model sizes and fast inferences are required. Previous work has introduced custom neural network solutions with custom visualization specifications and has not systematically tested pre-trained LMs to solve this problem. In this work, we opt for a more generic approach that (i) evaluates pre-trained LMs of different sizes and (ii) uses string encodings of data frames and visualization specifications instead of custom specifications. In our experiments, we show that these representations, in combination with pre-trained LMs, scale better than current state-of-the-art models. In addition, the small and base versions of the T5 architecture achieve real-time interaction, while LLMs far exceed latency thresholds suitable for visual exploration tasks. In summary, our models generate visualization specifications in real-time on a CPU and establish a new state of the art on the NL2VIS benchmark nvBench.

Keywords: NL2VIS, visualization generation, visualization-oriented dialog

1. Introduction

Generating plots from natural language queries (NL2VIS) allows users to create visualizations without knowing the syntax of the visualization library (e.g. matplotlib or Vega-Lite) (Hunter, 2007; Satyanarayan et al., 2016). This functionality is useful, for example, when a user is working on a Pandas data frame and wants to create a visualization for it in a Jupyter Notebook, as shown in Figure 1. To date, the NL2VIS task has received relatively little attention in NLP compared to closely related semantic parsing tasks. Previous work on NL2VIS has come mainly from the visualization community and has not fully incorporated state-of-the-art techniques such as pre-trained LMs. One goal of this work is to draw the attention of the NLP community to the NL2VIS task and to explore the potential of state-of-the-art NLP technology for NL2VIS.

Current methods for this task have introduced custom architectures that use specific encodings for the visualization specification to be generated. The problem with custom visualization representations, such as *Vega-Zero* (Luo et al., 2021b), is that they represent only a limited subset of the attributes of a visualization. This raises the barrier to entry for widespread use of NL2VIS models, since only a small subset of the possible options offered by the underlying visualization library can be queried by users. In addition to high accuracy in the mapping between queries and visualizations, the models' capabilities in terms of fast inference and smooth interaction are a key aspect of the NL2VIS task.

```
df.head(3)
```

	name	miles_per_gallon	cylinders
0	chevrolet chevelle malibu	18.0	8
1	buick skylark 320	15.0	8
2	plymouth satellite	18.0	8

```
nl2vis_model.generate_visualization(
    query="""show me the number of cars per
    year as a bar plot"""
)
```

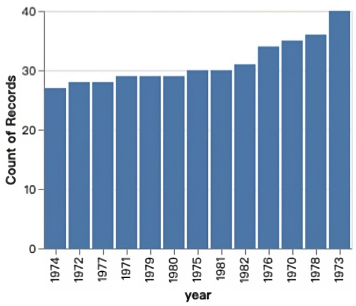

Figure 1: An application of a model trained on the nvBench dataset in a Jupyter notebook to translate natural language queries into visualization specifications.

The study by Liu and Heer (2014) shows that a latency of more than 500ms to the expected latency of an interaction significantly reduces user activity in visualization exploration tasks. However, interactive latencies are not systematically considered in existing LM-based methods for semantic parsing and could be a major limiting factor of LLMs in this

LREC-COLING 2024, pages 12787–12793

domain. Therefore, our goal in this work is to investigate not only the accuracy of a model, but also the relationship between the size of the LM and the inference time required, i.e., its usability in a visual exploration scenario.

Our contributions can be summarized as follows:

- We show that pre-trained T5 LMs using a string encoding of the visualization specification and data frame **outperform** custom state-of-the-art models on the *nvBench* benchmark.
- We analyze **error cases** and find that pre-trained T5 models achieve a **lower** word error rate than custom models, even though the latter use guided decoding strategies.
- We benchmark the performance of all models against the perception threshold of Liu and Heer (2014) and show that our models can be used in a **real-time** data exploration scenario on a **CPU**.

2. Related Work

Research on Natural language Interfaces (NLIs) enjoys increasing popularity in Visualization and NLP (Voigt et al., 2022). One line of work on NLIs has focused on usability, e.g., Srinivasan et al. (2019, 2021) found that user queries in natural language often focus on *attributes, diagram type, aggregation method* and visual features of the visualization. A second line of work addresses the mapping between natural language queries and visualization code, i.e. the NL2VIS task. Here, a number of rule-based parsers have been developed, e.g. *Articulate* (Sun et al., 2014), *NL4DV* (Narechania et al., 2020), or *FlowSense* (Yu and Silva, 2019). The shortcomings of these methods, especially their inflexibility with expressions unknown to the parser, have been addressed by recent neural architectures based on RNNs (Dibia and Demiralp, 2019), BERT (Liu et al., 2021), or sequence-to-sequence transformers (Tang et al., 2022). Yet, these approaches have not been compared on a standard NL2VIS benchmark. A third line of work implements NL2VIS as a component within dialog systems (Aurisano et al., 2016; Bacci et al., 2020), as in *Eviza* (Setlur et al., 2016) or *Chat2VIS* (Maddigan and Susnjak, 2023) where prompting methods for large language models such as *ChatGPT* (Ouyang et al., 2022) have been explored. The *ChartDialogs* (Shao and Nakashole, 2020) dataset provides dialogues where visualization specifications are incrementally modified by natural language commands. Our work addresses the second type of approach and focuses on benchmarking models for the NL2VIS task.

Recently, Luo et al. (2021a) introduced the first large-scale benchmark for NL2VIS, i.e. the *nvBench* corpus containing 25k query-visualization pairs. The state-of-the-art system on *nvBench*, *ncnet* (Luo et al., 2021b), uses a special encoding of the visualization called *Vega-Zero*, which maps a set of given attributes of a Vega-Lite specification into a string representation and uses a template-guided mechanism for decoding the visualization code. This ensures that the generated specification can also be rendered as a visualization. Other work uses generation approaches that condition on previously retrieved similar visualization specifications given a user query (Song et al., 2022; Bavishi et al., 2021) or generate proposals for visualizations from pure data frame representations (Dibia and Demiralp, 2019). Our work focuses on overcoming specialized coding for data frames and visualization specifications as well as template-based decoding or retrieval toward more generic and transferrable models.

3. Model and Training

Model. We experiment with three variants of the T5 architecture (Raffel et al., 2020) in sizes *small*, *base*, and *large*. We compare these to the instruction fine-tuned version called FLAN-T5 (Longpre et al., 2023), also in sizes *small*, *base*, and *large*, to see if instruction fine-tuning has an effect on the ability to map intentions from user queries. All model weights are taken from the huggingface hub (Wolf et al., 2020).

Dataset. The path from a user query to a finished visualization in the NL2VIS task is as follows: first, the user query in natural language is encoded and translated by the model into a visualization specification in JSON format. This specification is then passed to a visualization library such as Vega-Lite. This library has access to an underlying database and retrieves the relevant data defined in the specification. It then maps it to a visualization format also defined in the specification, such as a bar graph. Finally, it renders an image of the desired visualization and returns it to the user. In this pipeline, the NL2VIS models evaluated in this paper take over the first step: the mapping between query and specification.

To train and evaluate the models, we use the *nvBench* (Luo et al., 2021a) dataset. It contains 7,274 visualizations representing seven different types of graphs. Each of these visualizations is associated with one or more NL queries. All NL queries are provided in English. In total, the dataset consists of 25,750 (NL, VIS) pairs. The training set consists of 20,598 pairs, the validation set of 1,162 pairs, and the test set of 3,990 NL-VIS pairs. Furthermore, the data is categorized into four complexity levels, *easy, medium, hard* and *extra hard*, based on the difficulty of the natural

language query.

Input Encoding. The input to our model consists of the *user request* and a Pandas *data frame*. We encode both as strings using the SentencePiece tokenizer (Kudo and Richardson, 2018). The Pandas data frame is represented as the header, followed by the first three rows. An example of the encoding of a data frame looks like this: `table_name : products col : product_id (int64) | product_type_code (object) | ... row_0 : 1 | Hardware | ...` . The user query is appended to the encoded data frame, separated by double line breaks.

Output Encoding. Previous work by Luo et al. (2021b) has introduced a specific encoding for Vega-Lite specifications called *Vega-Zero*. The idea behind this encoding was to simplify the creation of visualization specifications, which are typically represented as large, nested JSON objects, by converting them to a simple sequential representation that captures the key attributes of the specification. A sample description of a visualization in *Vega-Zero* looks like this: `mark bar data products encoding x product_name y aggregate count product_name transform group x sort y desc`. To keep the performance of our approach comparable to the state-of-the-art architecture *ncnet* (Luo et al., 2021b), we train all model ablations using this *Vega-Zero* encoding as target output in the first experiment. In the second experiment, we train the same architectures to generate Vega-Lite specifications in JSON instead. The SentencePiece tokenizer that T5 was pre-trained with does not allow parentheses when encoding strings. Therefore, the Vega-Lite JSON object is flattened and converted to normalized JSON (Wes McKinney, 2010). An example of a normalized Vega-Lite JSON object looks like this: `"mark": "bar", "encoding_x_field": "product_name", "encoding_x_type": "nominal", "encoding_x_sort": "-y", "encoding_y_field": "product_name", ....`

Training. All models are trained for 10 epochs on an NVIDIA RTX A6000 GPU at a learning rate of 1e-4. We save the checkpoint with the best performance relative to the validation dataset and use it for evaluation.

4. Experiments and Results

The goal of our experiments is to investigate whether a string encoding of data frames and visualization specifications can be used to generate visualizations from natural language queries, and how the performance compares to customized state-of-the-art models (Section 4.1, 4.2). We also investigate whether these approaches allow real-time inference on CPUs (section 4.3) and analyze error cases (Section 4.4).

Model	Easy	Medium	Hard	EHard	Total
NN*	0.70	0.66	0.69	0.53	0.65
ncnet-3	0.76	0.74	0.77	0.59	0.72
ncnet-6	0.68	0.61	0.67	0.53	0.62
ncnet-12	0.63	0.55	0.62	0.49	0.57
t5-small	0.85	0.71	0.77	0.67	0.75
ft5-small	0.86	0.74	0.78	0.66	0.76
t5-base	0.90	0.74	0.87	0.68	0.80
ft5-base	0.86	0.75	0.86	0.69	0.79
t5-large	0.82	0.67	0.81	0.69	0.75
ft5-large	0.88	0.78	0.85	0.65	0.79

Table 1: **Vega-Zero Experiment.** Exact match accuracies on the nvBench dataset using *Vega-Zero* encoding. Starred (*) results are taken from Luo et al. (2021b). The numbers *3, 6, 12* represent the number of encoder/decoder layers of the *ncnet* model.

4.1. Scaling Experiments

In Table 1 we compare the results of different model variations. All accuracies are calculated as **exact match accuracies** to the gold standard outputs in the test set, to be comparable to previous work by Luo et al. (2021b). The results in Table 1 show that when the *ncnet* architecture is scaled to larger parameter sizes, its performance decreases as the model starts to overfit the training data. Without pre-training, the model reaches its performance limit already at 3 encoder/decoder layers, and further scaling of the parameters does not lead to a higher score. The pre-trained T5-*small* architecture, with a size of 6 encoder/decoder layers, performs significantly better than *ncnet* on the easy and extra hard splits, and equally well on the hard test split. The scores of the *base* model, increase significantly on the easy, medium, and hard splits in comparison to the *small* models, by an average of 0.04 for all splits. Interestingly, when the model size is further scaled to *large*, the performance decreases again in comparison to smaller T5 models, while still outperforming *ncnet*. This shows that even a pre-trained architecture tends to overfit the NL2VIS dataset if it has too many parameters. Comparing T5 with the FLAN models, we find that the *small* FLAN models perform slightly better, while the *base* models are relatively equal (except for the easy split), only the *large* model seems to overfit significantly less than the pure T5 version.[1]

[1]We also experimented with similarly sized GPT-2 architectures, but did not observe significant differences.

Model	Easy	Medium	Hard	EHard	Total
t5-small	0.85	0.72	0.80	0.68	0.76
ft5-small	0.84	0.74	0.79	0.65	0.75
t5-base	0.86	0.73	0.86	0.68	0.78
ft5-base	0.89	0.78	0.81	0.68	0.79
t5-large	0.81	0.65	0.76	0.63	0.71
ft5-large	0.88	0.74	0.79	0.65	0.77

Table 2: **Vega-Lite Experiment.** Exact match accuracies on the nvBench dataset using *Vega-Lite* encoding.

Model	Time in s	RTF
ncnet-3	0.09 ± 0.10	0.18
t5-small	0.31 ± 0.13	0.62
t5-base	0.46 ± 0.23	0.92
t5-large	1.63 ± 0.72	3.26
t5-small-onnx	0.12 ± 0.06	0.24
t5-base-onnx	0.27 ± 0.14	0.54
t5-large-onnx	0.77 ± 0.30	1.54
mistral-7B-2bit	90.40 ± 19.45	180.80

Table 3: **Latency Benchmark.** Average generation times per model on the nvBench test set.

Model	Easy	Medium	Hard	EHard	Total
ncnet-3	0.132	0.131	0.217	0.241	0.180
t5-small	0.133	0.152	0.182	0.198	0.166
ft5-small	0.137	0.167	0.169	0.227	0.175
t5-base	0.135	0.177	0.185	0.216	0.178
ft5-base	0.166	0.178	0.176	0.100	0.155
t5-large	0.124	0.189	0.247	0.098	0.165
ft5-large	0.143	0.178	0.241	0.131	0.173

Table 4: **WER Evaluation.** Average WER on the **error cases** per split for the different models.

4.2. Encoding Experiments

To see if Vega-Zero encoding is really necessary for reliable visualization specification generation, we train all T5 model ablations using the normalized JSON representation as the target output. The results in Table 2 show that the overall performance of the *small*, *medium*, and *large* model ablations is comparable to that of the Vega-Zero encoding. For the T5-*large* model, the same drop in performance is observed as for the Vega-Zero experiments. We see slightly higher values for the T5-*base* models trained with Vega-Zero, but only for the easy and medium splits. The averages across all splits are almost identical. Overall, these results show that both template-guided generation and Vega-Zero encoding, as used in the *ncnet* architecture, can be outperformed by string encoding of data frames and visualization combined with pre-trained T5-*small* or *base* language models.

4.3. Latency Experiments

Liu and Heer (2014) studied the effects of latency on user behavior in visual exploration tasks. According to Liu and Heer (2014), an interaction in this scenario is considered real-time if the system responds within 500ms of the user's known delay of an interaction. This means that if a mouse click with a known physical delay of 20ms occurs within a window of 520ms, it will still be perceived by the user as real-time. Liu and Heer (2014)'s Real-Time Factor (RTF) refers to the delay with which a process runs in real-time. An RTF of 1 means no delay, a factor greater than 1 means slower than real-time, and a factor less than 1 means faster than real-time.

We benchmark the latency of all model architectures on an Intel(R) Xeon(R) Gold 6226R 2.90GHz CPU. Since NL2VIS models are used offline and within the visualization library, CPU benchmarking allows for a realistic assessment. We measure the average time it takes a model to generate a Vega-Zero visualization specification across 100 random queries in the test set. The results are shown in Table 3. Note that *ncnet-3* is three times faster than the *small* T5 model. However, due to the generic architecture of the T5 model, it can be very easily converted to an ONNX format and executed in ONNXRuntime (Developers, 2023). This allows the T5 model *small* to achieve almost the same performance as *ncnet-3*, with significantly higher accuracy in the outputs. With a generation time of 0.27 seconds, the *base* model also offers real-time capabilities. To compare generation times with current SOTA-LLM architectures, we prompt the CPU-capable 2-bit quantized version of the Mistral-7B-Instruct model (Jiang et al., 2023) to repeat given Vega-Zero strings from the test set. This simulates the model generating specifications. The results show that the generation takes about 750 times longer than for the *t5-small-onnx* model and with an RTF of 180.8 does not allow real-time data exploration on a CPU for our use case. This shows that without hardware acceleration, the real-time requirements of the NL2VIS task are a limiting factor for LLMs in this domain.

4.4. Error Analysis

For both the earlier state-of-the-art *ncnet* model and our T5 ablations, we compute the word error rate (WER) (Wang et al., 2003) on all **failure cases** per split, as an indication of how close the model was to the correct output. Table 4 shows that T5-*small* and *base* have a slightly lower WER on average than *ncnet*. In particular, in the hard and extra-hard divisions of the benchmark, the *small* and *base* approaches show lower WER. This is notable because *ncnet* uses a template-driven decoding mechanism, where it only needs to fill in blanks in a Vega-Zero template, rather than gener-

ating from scratch as T5 does. This suggests that template-guided decoding is helpful for controlling *ncnet's* output in the easy and medium cases. However, with more difficult examples, it becomes more difficult for the model to understand the user's intent based on the query correctly. This is where pre-trained models have an advantage, as they have gained more knowledge through pre-training that could help them infer these intentions better.

5. Conclusion

In this work, we investigated the use of string encodings for data frames and visualization specifications with small, pre-trained LMs such as T5 to generate visualizations in real-time. Our experiments have shown that their application outperforms custom neural network solutions on the NL2VIS benchmark. This eliminates the need for custom encodings such as Vega-Zero. Moreover, by analyzing error cases, we found that pre-trained T5 ablations achieve lower word error rates than custom models, especially on difficult splits. This indicates that knowledge acquired through pre-training is important for understanding user queries, which custom models have to a lesser extent. Finally, we show that small- and medium-sized LMs meet the requirements of real-time interaction on a CPU whereas LLMs exceed the expected thresholds dramatically. We hope that our results, showing that custom LMs are not necessary for high performance, will inspire people in the future to extend the NL2VIS task to interactive, visualization-oriented dialogs by leveraging existing visualization datasets.

6. Limitations

We deliberately chose very simple representations of data frames and Vega-Lite visualization specifications to explore the impact of pre-training when interpreting user queries through models. This does not mean that there are not other efficient and similar or even more powerful representation methods for data frames and Vega-Lite visualizations. We chose the T5 model and its variants because it is easy to train, tune, and deploy, which is important for our application scenario. However, the T5 architecture has weaknesses in terms of limited tokenization capabilities of arbitrary strings, as certain characters simply cannot be represented. It also has limitations due to its pre-training dataset and procedure. We experimented with comparable architectures in this model size category, such as GPT-2 or LaMini, but could not find any significant changes in the results. Therefore, we decided to stay with the T5 models and use them for the experiments. In order to compare our work with the state-of-the-art work of Luo et al. (2021b), we decided to use exact match accuracy to measure the performance of our models. In our experiments, we found that this measure has clear limitations when it comes to measuring how well a model really understands the user query. This makes it necessary to look at error cases in more detail. For the future, we also consider human evaluation as a valid form of estimating the quality of a model in this domain. Users sometimes have specific plots in mind, and the plot generated by the system differs greatly in notation from the gold standard, but the user may still visually perceive the plot as satisfactory. This could be an interesting direction for evaluating models for generating visualizations from natural language queries in the future. In conclusion, we believe that our approach has shown that the combination of pre-trained LMs and the use of straightforward encodings is promising for generating visualizations based on natural language queries. We hope that these results will motivate the community to collect more datasets and extend the idea of NL2VIS to interactive plot manipulation for different plotting languages.

7. Ethics Statement

The nvbench dataset and the T5 and FLAN-T5 models are publicly available for research and non-commercial use. The ncnet model is also publicly available on github. We clearly state the intended use of our models, which is to facilitate data exploration by generating visualizations using natural language. The factual correctness of generating visualizations based on natural language queries must always be taken with a grain of salt. The model generates a visualization based on a given dataset and the query based on its current level of performance. Sometimes this can lead to incorrect results that can be misinterpreted if not corrected. Therefore, it is strongly discouraged to use the results of the model per se and to take them as given or to draw unwarranted conclusions from them. Rather, the generation process should be reviewed by looking at the generated visualization specifications and verifying that they make sense and are correct in the context of the given dataset and query.

8. Bibliographical References

Jillian Aurisano, Abhinav Kumar, Alberto Gonzalez, Jason Leigh, Barbara DiEugenio, and Andrew Johnson. 2016. Articulate2: Toward a conversational interface for visual data exploration. In *IEEE VIS*, volume 16, page 1.

Franscesca Bacci, Federico Maria Cau, and Lucio Davide Spano. 2020. Inspecting data using natural language queries. In *Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part VI 20*, pages 771–782. Springer.

Rohan Bavishi, Shadaj Laddad, Hiroaki Yoshida, Mukul R Prasad, and Koushik Sen. 2021. Vizsmith: automated visualization synthesis by mining data-science notebooks. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 129–141. IEEE.

ONNX Runtime Developers. 2023. Onnx runtime. https://www.onnxruntime.ai. Version: 1.14.0.

Victor Dibia and Çağatay Demiralp. 2019. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications*, 39(5):33–46.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20. IEEE.

Zhicheng Liu and Jeffrey Heer. 2014. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20(12):2122–2131.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Yuyu Luo, Jiawei Tang, and Guoliang Li. 2021a. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task. *arXiv preprint arXiv:2112.12926*.

Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2021b. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226.

Paula Maddigan and Teo Susnjak. 2023. Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models. *IEEE Access*.

Arpit Narechania, Arjun Srinivasan, and John Stasko. 2020. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350.

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.

Yutong Shao and Ndapandula Nakashole. 2020. Chartdialogs: Plotting from natural language instructions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3559–3574.

Yuanfeng Song, Xuefang Zhao, Raymond Chi-Wing Wong, and Di Jiang. 2022. Rgvisnet: A hybrid retrieval-generation neural framework towards automatic data visualization generation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1646–1655.

Arjun Srinivasan, Mira Dontcheva, Eytan Adar, and Seth Walker. 2019. Discovering natural language commands in multimodal interfaces. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 661–672.

Arjun Srinivasan, Nikhila Nyapathy, Bongshin Lee, Steven M Drucker, and John Stasko. 2021. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–10.

Yiwen Sun, Jason Leigh, Andrew Johnson, and Barbara Di Eugenio. 2014. Articulate: Creating meaningful visualizations from natural language. In *Innovative Approaches of Data Visualization and Visual Analytics*, pages 218–235. IGI Global.

Jiawei Tang, Yuyu Luo, Mourad Ouzzani, Guoliang Li, and Hongyang Chen. 2022. Sevi: Speech-to-visualization through neural machine translation. *Proceedings of the 2022 International Conference on Management of Data*.

Henrik Voigt, Ozge Alacam, Monique Meuschke, Kai Lawonn, and Sina Zarrieß. 2022. The why and the how: A survey on natural language interaction in visualization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 348–374, Seattle, United States. Association for Computational Linguistics.

Ye-Yi Wang, Alex Acero, and Ciprian Chelba. 2003. Is word error rate a good indicator for spoken language understanding accuracy. In *2003 IEEE workshop on automatic speech recognition and understanding (IEEE Cat. No. 03EX721)*, pages 577–582. IEEE.

Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

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*, 26(1):1–11.