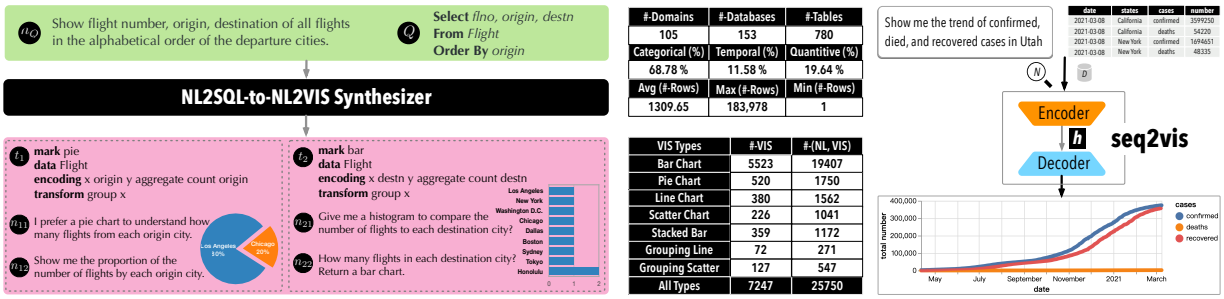
Empowering Natural Language to Visualization Neural Translation using Synthesized Benchmarks

Yuyu Luo
Tsinghua University, China

Jiawei Tang
American School of Doha, Qatar

Guoliang Li
Tsinghua University, China

Chengliang Chai
Tsinghua University, China



(a) Synthesizing **nvBench** from NL2SQL Benchmarks (b) Statistics of **nvBench** (c) **seq2vis**: NL2VIS by Neural Machine Translation

Figure 1: (a) An example of synthesizing multiple (NL, VIS) pairs from one (NL, SQL) pair. (b) The statistics of **nvBench**, the NL2VIS benchmark synthesized from NL2SQL benchmarks. (c) A sequence-to-sequence model, trained using **nvBench**, for NL2VIS translation.

Abstract

NL2VIS – which supports the translation from natural language (NL) to visualization (VIS) – is an important interaction paradigm for data visualization. Over the years, deep learning models have achieved near-human intelligence in many natural language processing (NLP) tasks, which clearly indicates that a learned approach is a viable option for NL2VIS as well. Nevertheless, a big obstacle of enabling neural machine translation for NL2VIS is the lack of benchmarks. We present **nvBench**, the first NL2VIS benchmark synthesized from NL2SQL benchmarks, with 25,750 (NL, VIS) pairs on top of 780 tables across 105 domains, which has been extensively validated by 23 experts and 300+ crowd workers. We will also demonstrate that **nvBench** can empower deep learning-based neural machine translation for NL2VIS, using COVID-19 use cases.

1 Introduction

Natural language interface is a promising interaction paradigm for data visualization, and is a crucial step towards democratizing data visualization. Not surprisingly, both commercial vendors (*e.g.,* Tableau's Ask Data, Microsoft Power BI, ThoughtSpot, and Amazon's QuickSight) and academic researchers [5, 8] have started to explore the techniques to support NL2VIS.

Recent advances, such as NL4DV [5] and FlowSense [8], employ NLP toolkits (*e.g.,* NLTK, Standford CoreNLP, and NER) to parse an NL query and produce a variety of linguistic annotations (*e.g.,* parts of speech, named entities, coreference, etc). These are also known as semantic parsers that use symbolic NLP techniques.

In the last decade, we have witnessed an unstoppable rise of deep learning for NLP. These deep neural networks, from recurrent neural networks (RNNs) [1], long short-term memory (LSTM) [2], to Transformer [7], have shown superior performance than traditional semantic parsers in almost all NLP tasks. Evidently, deep learning models should play a key role for pushing the field of NL2VIS.

NL2VIS is a *machine translation* task that translates a natural language query (*e.g.,* "show me the trend of confirmed, died and recovered cases in Utah") to a visualization query (*e.g.,* mark line encoding x date y aggregation ...") so as to be rendered (*e.g.,* by Vega-Lite). A key factor to empower neural machine translation for NL2VIS is a benchmark with lots of (NL, VIS) pairs, because deep learning models are known to be *data hungry*.

In this poster, we present **nvBench** [4], the **first NL2VIS benchmark** that is designed for empowering neural machine translation models for NL2VIS. **nvBench** was synthesized from NL2SQL benchmarks, with synthesized results being manually validated by experts and crowd-workers. We will also describe how **nvBench** can be used to train a Sequence-to-Sequence model, namely **seq2vis**, to effectively support NL2VIS with several COVID-19 use cases.

2 nvBench: A Large NL2VIS Benchmark

The widely used practice of producing benchmarks is through time consuming manual labeling, *e.g.,* providing visualizations and ask experts to write corresponding NL queries.

The main issue of the above approach is that the required experts are simple not enough. Alternatively, we propose to synthesize NL2VIS benchmarks [4] from a plethora of NL2SQL benchmarks. Because it is known that verifying results (*i.e.,* whether an NL query is suitable for a given visualization) is much easier than writing the NL query manually, both experts and crowd-workers can help.

The rationality that NL2VIS benchmarks can be synthesized from NL2SQL benchmarks is because the semantic connection between VIS queries and SQL queries: SQL queries specify *what* data is needed (*e.g.,* columns, filtering, aggregation, sorting); and VIS queries specify both *what* data is needed and *how* to visualize (*e.g.,* bar or line charts) – the *what data* parts highly overlap. Intuitively, we can piggyback NL2SQL benchmarks on the *what data* part and focus on synthesizing *how to visualize* for NL2VIS.

2.1 Synthesizing nvBench from NL2SQL Benchmarks

Briefly speaking, given a (NL, SQL) pair, our method will synthesize a set of (NL, VIS) pairs. Consider Fig. 1(a), the input is a pair (n_Q, Q). It outputs four pairs (v_1, n_{11}), (v_1, n_{12}), (v_2, n_{21}), and (v_2, n_{22}), where v_1 (resp. v_2) is a pie (resp. bar) chart, and n_{11} and n_{12} (resp. n_{21} and n_{22}) are variants of NL queries for v_1 (resp. v_2).

The synthesis steps from one (NL, SQL) pair to multiple (NL, VIS) pairs are summarized below (please refer to [4] for technical details).

(S1) Synthesizing visualizations. It treats an SQL query Q as a tree

structure and does tree edits (*e.g.,* deleting some tree branches and inserting the type of visualizations), which may result in multiple trees, with each tree corresponding to one possible visualization.

(S2) Filtering "bad" visualizations. In order to ensure that each VIS query is "good" (for example, a bar chart with several hundred bars is not readable, and thus is considered to be bad), we need to filter "bad" charts. We use a pre-trained machine learning model, namely DeepEye [3], to prune synthesized bad VIS queries. DeepEye was trained on 2520/30892 labeled good/bad charts, using features such as the number of distinct values, the number of tuples, the ratio of unique values, max and min values, data type, attribute correlations, and VIS type. Given a VIS query, DeepEye will return either true (*i.e.,* a good VIS) or false (*i.e.,* a bad VIS).

(S3) Synthesizing NL queries. For the remaining "good" visualizations, we need to modify the input NL query for SQL (*e.g.,* n_Q in Fig. 1(a)) to reflect the changes *w.r.t.* tree edits, which might result in multiple output NL queries, *e.g.,* n_{11} (resp. n_{12}) is synthesized from n_Q based on the differences between t_1 (resp. t_2) and n_Q.

(S4) Manual verification. We asked 23 experts and 312 crowd-workers to verify the quality of synthesized (NL, VIS) pairs. Experts/crowd-workers consider 86.9%/88.7% of synthesized (NL, VIS) pairs are well-matched, *i.e.,* scored 4 or 5 in a range [1,5] with 1 for bad matches and 5 for perfect matches. As measured by [4], our synthesis method reduces the man-hour to 5.7% of developing a NL2VIS benchmark from scratch. In other words, building a NL2VIS benchmark only by human takes 17.5× man-hours of our method.

2.2 nvBench: Statistics

Figure 1(b) overviews the statistics of **nvBench**, synthesized from a cross-domain NL2SQL benchmark Spider [9].

nvBench has 153 databases along with 780 tables in total and covers 105 domains (*e.g.,* sports, customers). Among the columns, 68.78% of columns are categorical columns, 11.58% of columns are temporal columns, and 19.64% of columns are quantitative columns. The maximum number of rows in a table is 183,978, and the minimum number of rows is 1, with an average 1309.65 rows.

On top of 153 databases, **nvBench** contains 7,274 visualizations on seven types of charts. For each visualization, **nvBench** provides one to several NL queries. In total, **nvBench** consists of 25,750 (NL, VIS) pairs.

3 seq2vis: Empowering NL2VIS Neural Translation

Sequence-to-Sequence Models for NL2VIS. A sequence-to-sequence (seq2seq) model [6] consists of two parts, an encoder and a decoder (see Fig. 1(c)), where each part can be implemented by different neural networks. The task of an encoder is to understand the input sequence, and generate a smaller representation h (*i.e.,* a high-dimensional vector) to represent the input. The task of a decoder is to generate a sequence of output by taking h as input. The network needs to be trained with a lot of training data, in the form of (Input sequence, Output sequence) pairs.

For NL2VIS, we train a seq2seq network, namely **seq2vis**, with a lot of (NL, VIS) pairs from **nvBench**, such that it learns to translate from an NL query to a visualization.

COVID-19 Use Cases. We use the COVID-19 table, with the column names (`date, states, cases, number`), to demonstrate how to create good visualizations with NL queries with Jupyter Notebook. We invited a data visualization enthusiast Kevin who has experience in building a COVID-19 dashboard. As shown in Fig. 2, Kevin first imports the **seq2vis** from the python package and then initializes the **seq2vis** by passing the dataset parameters. Next, he can overview the dataset by calling the function `show_dataset( )` or explore the dataset using other packages like `Pandas-profiling`. Kevin specifies an NL query via the function `nl2vis(nl_question)`,

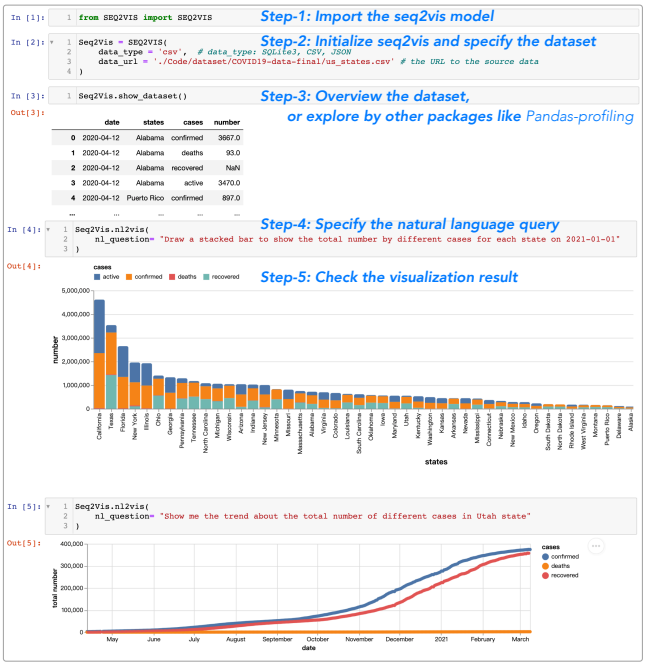


Figure 2: Sample of using **seq2vis**, trained using **nvBench**, for supporting NL2VIS in Jupyter Notebook.

and then he can check the visualization given by **seq2vis**. If he does not satisfy with the result, he can rephrase the NL query and try again. Before, he spends hours transforming the data and writing Vega-Lite code to visualize; now, Kevin blinks and it's done.

4 Conclusion and Next Steps

In this poster, we have presented **nvBench**, the first NL2VIS benchmark that was designed to enable deep learning-based neural machine translation for NL2VIS. The quality of **nvBench** has been validated by both experts and crowd-workers. We have also discussed about how to train a seq2seq model for learning the NL2VIS translation. Our case studies show that **seq2vis**, trained using **nvBench**, can satisfactorily perform NL2VIS. There are two interesting future works. First, we plan to expand **nvBench** to support more types of visualizations. Second, we plan to further optimize seq2seq for NL2VIS, so as to be more robust and achieve better accuracy.

References

[1] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[3] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. In *ICDE*, pp. 101–112, 2018.

[4] Y. Luo, N. Tang, G. Li, C. Chai, W. Li, and X. Qin. Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks. In *SIGMOD*, 2021. Available at https://sites.google.com/view/nvbench/.

[5] A. Narechania, A. Srinivasan, and J. T. Stasko. NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. In *VIS*, 2020.

[6] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS 2014*, p. 3104–3112. MIT Press, 2014.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[8] B. Yu and C. T. Silva. Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE Trans. Vis. Comput. Graph.*, 26(1):1–11, 2020.

[9] T. Yu and et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.