Data2Vis: Automatic Generation of Data Visualizations Using Sequence to Sequence Recurrent Neural Networks

Victor Dibia
IBM Research

Çağatay Demiralp
IBM Research

ABSTRACT

Rapidly creating effective visualizations using expressive grammars is challenging for users who have limited time and limited skills in statistics and data visualization. Even high-level, dedicated visualization tools often require users to manually select among data attributes, decide which transformations to apply, and specify mappings between visual encoding variables and raw or transformed attributes.

In this paper we introduce Data2Vis, a neural translation model for automatically generating visualizations from given datasets. We formulate visualization generation as a sequence to sequence translation problem where data specifications are mapped to visualization specifications in a declarative language (Vega-Lite). To this end, we train a multilayered attention-based recurrent neural network (RNN) with long short-term memory (LSTM) units on a corpus of visualization specifications.

Qualitative results show that our model learns the vocabulary and syntax for a valid visualization specification, appropriate transformations (count, bins, mean) and how to use common data selection patterns that occur within data visualizations. Data2Vis generates visualizations that are comparable to manually-created visualizations in a fraction of the time, with potential to learn more complex visualization strategies at scale.

Author Keywords

Automated visualization design; neural machine translation; sequence to sequence models; deep learning; LSTM; Vega-Lite.

INTRODUCTION

Users create data visualizations using a range of tools with a range of characteristics (Figure 1). Some of these tools are more expressive, giving expert users more control, while others are easier to learn and faster to create visualizations, appealing to general audiences. For instance, imperative APIs such as OpenGL and HTML Canvas provide greater expressivity and flexibility but require significant programming skills and effort. On the other hand, dedicated visual analysis tools and spreadsheet applications (e.g., Microsoft Excel, Google Spreadsheets) provide ease of use and speed in creating standard charts based on templates but offer limited expressivity and customization.

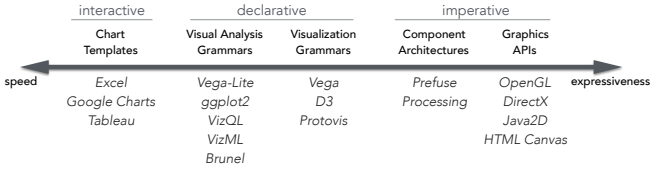

Figure 1. Axis of visualization specification. Data visualizations are created with a spectrum of tools with a spectrum of speed and expressivity. Some of these tools are faster to create visualizations, others are more expressive [28].

Declarative specification grammars such as ggplot2 [61], D3 [10], Vega [50], and Vega-Lite [49] provide a trade-off between speed and expressivity. However, these grammars also come with steep learning curves, can be tedious to specify depending on the syntax and abstraction level adopted, and can suffer from reusability issues. In fact, there is little known about the user experience with visualization grammars, beyond the degree with which they are used. For example, ggplot2 can be difficult for users who are not familiar with R. Vega, which is based on a JSON schema, can be tedious even for users who are familiar with JSON. Even tools with higher-level abstractions such as the ones based on chart templates often require the user to manually select among data attributes, decide which statistical computations to apply, and specify mappings between visual encoding variables and either the raw data or the computational summaries. This task can be daunting with complex datasets especially for typical users who have limited time and limited skills in statistics and data visualization.

To address these challenges, researchers have proposed techniques and tools to automate designing effective visualizations [14, 20, 36, 43, 37, 48] and guide users in visual data exploration [2, 19, 23, 44, 48, 52, 54, 59, 65, 66, 67].

Prior techniques and tools for automated visualization design and visualization recommendation are based on rules and heuristics. The need to explicitly enumerate rules or heuristics limits the application scalability of these approaches and does not take advantage of expertise codified within existing visualizations. Automated and guided visualization design and exploration can significantly benefit from implicitly learning these rules from examples (i.e., data), effectively incorporating both data and visualization design context.

In this work, we formulate visualization design as a sequence to sequence translation problem. To operationalize our for-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

mulation, we train an LSTM-based neural translation model (Data2Vis) on a corpus [46] of Vega-Lite visualization specifications, taking advantage of Vega-Lite's (and of similar grammars') design motivation to support programmatic generation. We demonstrate the model's use in automatically generating visualizations with applications in democratizing the visualization authoring process for novice users and helping more experienced users jump start visualization design. Our contributions include 1) formulating visualization design as a sequence to sequence translation problem, 2) demonstrating its viability by training a sequence to sequence model, Data2Vis, on a relatively small training dataset and then effectively generating visualizations of test data, and 3) integrating Data2Vis into a web-based application that has been made publicly available at `http://hci.stanford.edu/~cagatay/data2vis`. Our work is the first in applying deep neural translation to visualization generation and has important implications for future work, opening the way to implicitly learn visualization design and visual analysis rules from examples at scale.

In what follows, we first summarize related work followed by details of the Data2Vis model and its training process. We then present our results, providing several visualization examples automatically generated using the trained model. Next we discuss the potential impact of Data2Vis and its current limitations and provide an agenda for future work. We conclude by summarizing our contributions and insights.

RELATED WORK

Our work is related to earlier efforts in effective visualization specification, automated visualization design, and deep neural networks (DNNs) for synthesis and machine translation.

Declarative Visualization Specification

Earlier data visualization work proposes grammars and algebraic operators over data as well as visual encoding and design variables to specify visualizations (Figure 1). Wilkinson's seminal work [62] introduces a grammar of graphics and its implementation (VizML), greatly shaping the subsequent research on visualization specification. Polaris [55] (now called Tableau) uses a table algebra drawn from Wilkinson's grammar of graphics. The table algebra of Polaris later evolved to VizQL [27], forming the underlying representation of Tableau visualizations. Wickham introduces ggplot2 [61], a widely-popular package in the R statistical language, based on Wilkinson's grammar. Similarly, Protovis [9], D3 [10], Vega [50], Brunel [64], and Vega-Lite [49] all provide grammars to declaratively specify visualizations. Some of them require more complete specifications than others. For instance, Protovis, D3 and Vega support finer control over visualization specification with incurred cost of verbosity. Satyanarayan et al. [49] introduce Vega-Lite (Figure 2) as a strict subset of Vega, a JSON schema based grammar, to facilitate clarity and conciseness with some loss in expressivity. We train our model on a Vega-Lite corpus [46], which contains datasets and corresponding visualizations specified in Vega-Lite.

Declarative grammars eschew chart templates typically used in dedicated visualization tools or spreadsheet applications such as Microsoft Excel and Google Spreadsheets, which

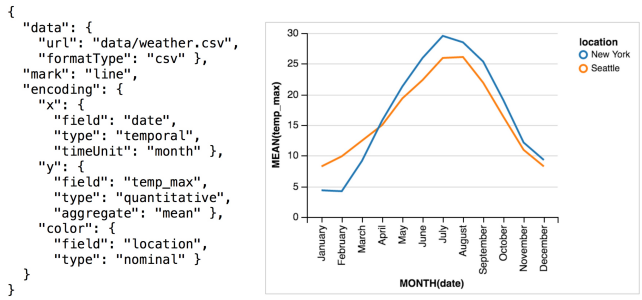

Figure 2. A Vega-Lite specification (left) and the generated visualization (right). Users can succinctly specify complex selections, transformations and interactions using the Vega-Lite grammar formatted in JSON [49].

have limited support for customization. Conversely, these grammars facilitate expressivity by enabling a combinatorial composition of low-level building blocks such as graphical marks, scales, visual encoding variables, and guides. However, increased expressivity often decreases the speed with which visualizations can be created and makes the learning more difficult, limiting the number of users who can effectively use the specification method. One of our aims with Data2Vis is to bridge this gap between the speed and expressivity in specifying visualizations.

Automated Visualization

Prior work proposes desiderata and tools (e.g., [14, 20, 37, 43, 48]) to automatically design effective visualizations, building on Bertin's study [7] of visual encoding variables and earlier graphical perception research, e.g., [1, 5, 18, 38, 43, 53]. Earlier research also develops interactive systems and recommendation schemes [11, 25, 44, 52, 54, 57, 58, 59, 63, 65, 66, 67] to guide users in exploratory data analysis and visualization design. PRIM-9 [23], GrandTour [2] SeeDB [59], Zenvisage [54], ShowMe [44], Voyager [66], Voyager 2 [67], SAGE [48] and VizDeck [36] prioritize charts according to one or more evaluation measures such as data saliency, data coverage, perceptual effectiveness, user task, and user preferences. Similarly, Rank-by-Feature [52], AutoVis [65], and Foresight [19] use statistical criteria over data attributes and instances in recommending and ranking visualizations. Data2Vis exhibits a departure from rule-based approaches of prior work both in conceptual formulation and technical approach taken. The Data2Vis learns how to create visualization specifications from examples without retorting to enumeration of rules or heuristics, complementing earlier work. The Data2Vis model can be integrated into existing higher-level recommendation systems of visual data exploration and used in tandem with rule-based techniques to drive these systems.

Deep Neural Networks for Synthesis

Prior deep neural network (DNN) research studies adopt generative approaches to learn human-like cognitive and creative capabilities. Examples include the use of models to synthesize music, drawings, images from textual descriptions, code from hand-drawn sketches or interface screenshots. Ha et al. [26] train a recurrent neural network (RNN) to predict and generate stroke-based drawings of common objects. Reed et al. [47]

present a DNN architecture and GAN formulation to "translate" textual visual concepts to pixels. Others learn how to generate code from user interface screenshots [6] and how to compose music using purely sequential models [22, 31] and cascading a sequential model with a restricted Boltzman machine [12]. All these approaches aim to simplify the creative process for both novices and experts. In this sense, our work here shares a motivation similar to prior work. We also use a variation of sequential neural network models, a sequence to sequence model, to generate visualization specifications from given data.

Deep Neural Networks for Machine Translation

Recent work introduces DNN models, e.g., [3, 16, 32, 41, 56] that significantly improves [30, 42, 51, 68] the performance of machine translation systems, surpassing the preceding phrase-based approaches. Deep neural translation models eschew hand engineering the features, in large part, by using large training data, enabling the end-to-end learning in practice. Sequence to sequence models (e.g., [3, 41]) are a particularly successful and popular class of deep learning models applied in machine translation (see [13] for an evaluation of alternative architectures). Akin to autoencoders, these models have also a symmetric, encoder-decoder architecture. Sequence to sequence models are composed of encoder-decoder layers which consist of recurrent neural networks (RNNs) and an attention mechanism that aligns target tokens with source tokens.

Earlier work, e.g., [4, 15, 21, 39, 45, 70] also uses DNNs to translate between domain specific languages (DSLs), natural language specifications and DSLs, and programming languages. Data2Vis directly uses the source and target text without taking syntax into consideration (e.g., using abstract syntax trees) explicitly. In this sense, it is similar to [39], which uses a sequence to sequence model to translate Trading Card Games (TCG) cards to their Python and Java specifications without explicitly representing the target syntax.

PROBLEM FORMULATION

Building on earlier work that applies deep learning for translation and synthesis, we formulate the data visualization problem as a sequence to sequence translation problem, which can be readily addressed using sequence to sequence models (seq2seq) [3, 16, 56]. Our input sequence is a dataset and our output sequence is a valid Vega-Lite specification. Seq2seq models have been applied with great success to learn mappings between sequential data [3, 13, 40, 41]. Examples areas include neural machine translation, neural conversation modeling, text summarization, image captioning etc.

Seq2seq models for machine translation are trained using embeddings of the source and target tokens which can be generated based on words or subword units or per character [3, 16, 56]. In this work, we select per character tokenization given our source and target sequences consist of symbols as opposed to learnable word groups seen in related problems like language translation.

MODEL

Our model (Figure 3) is based on an encoder-decoder architecture with attention mechanism that has been previously applied in machine translation [3, 40, 41]. The encoder is a bidirectional recurrent neural network (RNN) that takes in an input sequence of source tokens $x = (x_1, ..., x_m)$ and outputs a sequence of states $h = (h_1, ..., h_m)$. The decoder is also an RNN that computes the probability of a target sequence $y = (y_1, ..., y_k)$ based on the hidden state h. The probability of each token in the target sequence is generated based on the recurrent state of the decoder RNN, previous tokens in the target sequence and a context vector c_i. The context vector (also called the attention vector) is a weighted average of the source states and designed to capture the context of source sequence that help predict the current target token.

We use a 2-layer bidirectional RNN encoder and a 2-layer RNN decoder, each with 256 Long Short-Term Memory (LSTM) [24, 29] units (cells). To decide which RNN unit type to use, we experimented with both gated recurrent unit (GRU) [17] and LSTM, both of which are common RNN cell variants. We found LSTM cells providing better results than GRU cells, which concurs with earlier empirical results [13].

DATA AND PREPROCESSING

To generate plausible visualizations conditioned on a given source dataset, our model should achieve several learning objectives. First, the model must select a subset of fields to focus on when creating visualizations (most datasets have multiple fields which cannot all be simultaneously visualized). Next, the model must learn differences in data types across the data fields (numeric, string, temporal, ordinal, categorical etc.), which in turn guides how each field is specified in the generation of a visualization specification. Finally, the model must learn the appropriate transformations to apply to a field given its data type (e.g., aggregate transform does not apply to string fields). In our case, this includes view-level transforms (aggregate, bin, calculate, filter, timeUnit) and field level transforms (aggregate, bin, sort, timeUnit) supported by the Vega-Lite grammar.

Achieving these objectives using a character based sequence model can be resource intensive. While character based models results in smaller vocabulary size and are more accurate for specialized domains, they also present challenges — a character tokenization strategy requires more units to represent a sequence and requires a large amount of hidden layers as well as parameters to model long term dependencies [8]. To address this issue and scaffold the learning process, we perform a set of transformations. First, we replace string and numeric field names using a short notation — "str" and "num" in the source sequence (dataset). Next, a similar backward transformation (post processing) is eplicated in the target sequence to maintain consistency in field names (see Figure 4). These transformations help scaffold the learning process by reducing the vocabulary size, and prevents the LSTM from learning field names (as we observed in early experiments). In turn we are able to reduce the overall source and target sequence length, reduce training time and reduce the number of hidden layers which the model needs to converge.

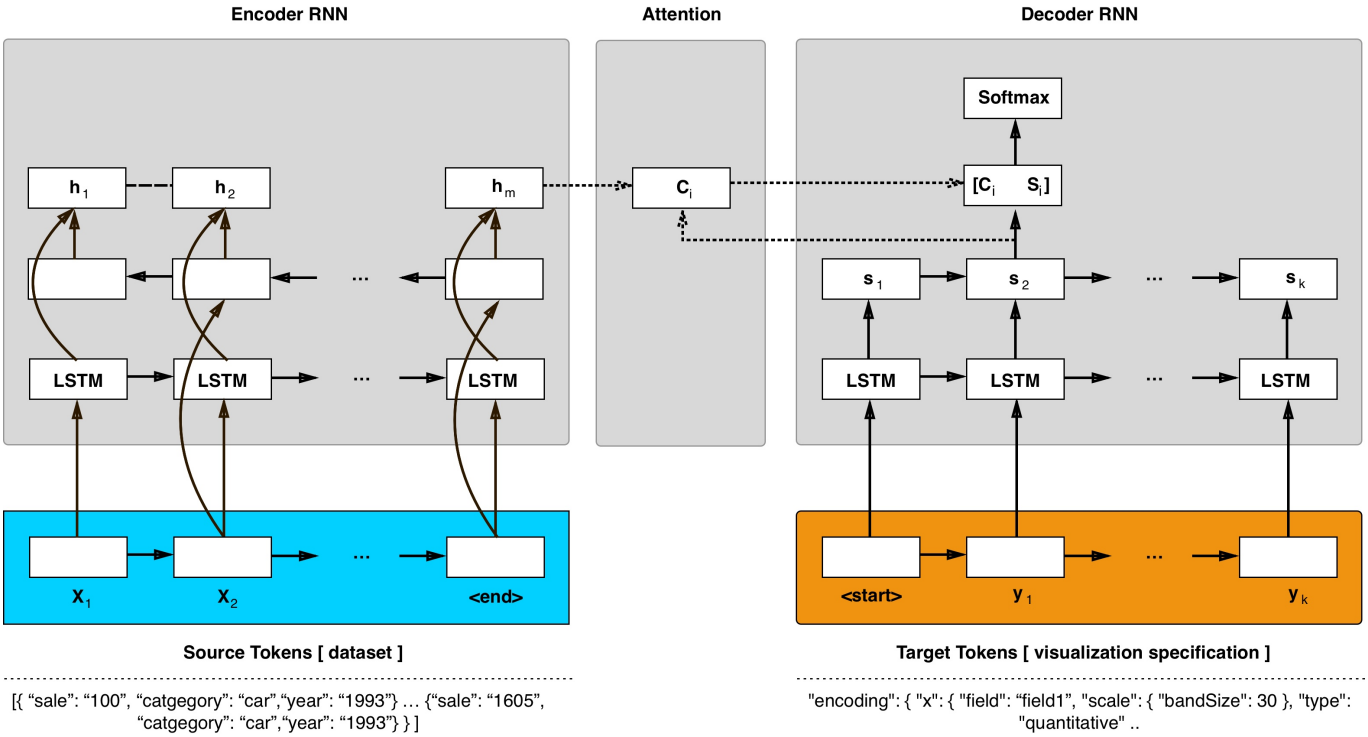

Figure 3. Data2Vis is a sequence to sequence model with encoder-decoder architecture and attention module. Source and target tokens are character token representations of the source and target data respectively.

Our training dataset is constructed from 4300 automatically generated Vega-Lite examples, based on 11 distinct datasets. The dataset was originally compiled by [46] where the authors use Voyager [66] to generate charts with 1-3 variables, filtered to remove problematic instances. The dataset contains charts with 6 different types of visualizations (area, bar, circle, line, point, tick) and three different transforms (aggregate, bin, timeUnit)(see Figure 5). We iteratively generate a source (a single row from the dataset) and target pair (see Figure 4) from each example file. Each example is sampled 50 times (50 different data rows with the same Vega-Lite spec) resulting in a total of 215000 pairs which are then used to train our model.

Training

We begin by generating a character vocabulary for our source and target sequences (84 and 45 symbols, respectively). A dropout rate of 0.5 is applied at the input of each cell and a maximum source and target sequence length of 500 is used. The entire model is then trained end-to-end using a fixed learning rate of 0.0001 with Adam optimizer, minimizing the negative log likelihood of the target characters using stochastic gradient descent. Our implementation is adapted from an open source neural machine translation framework by Britz *et al.* [13]. We train our model for a total of 20,000 steps and final loss of 0.02.

RESULTS

Qualitative Evaluation

Quantifying the performance of a generative model can be challenging. Following existing literature [31, 34, 35], we explore a qualitative evaluation of the model's output. To evaluate the model, we use the Rdataset repository [1] (cleaned and converted to a valid JSON format) which was not included in our training. Figure 9 shows visualizations generated from 24 randomly selected datasets in the Rdataset collection. The range of valid univariate and multivariate visualizations produced suggests the model captures aspects of the visualization generation process. As training progresses, the model incrementally learns the vocabulary and syntax for valid Vega-Lite specifications, learning to use quotes, brackets, symbols and keywords. The model also appears to have learned to use the right type of variable specifications in the Vega-Lite grammar (e.g. it correctly assigns a string type for text fields and a quantitative for numeric fields). Qualitative results also suggest the use of appropriate transformations (bins, aggregate) on appropriate fields (e.g. means are performed on numeric fields). The model also learns about common data selection patterns that occur within visualizations and their combination with other variables to create bivariate plots. As experts create visualizations, it is common to group data by geography (country, state, sex), characteristics of individuals (citizenship status, marital status, sex) etc. Early results suggests that our model begins to learn these patterns and apply them in its generation of visualizations. For example, it learns to subset data using common ordinal fields such as responses (yes/no), sex (male/female) etc and plots these values against other fields (Figure 7). Finally, in all cases, the model generates a perfectly

[1] Rdatasets is a collection of 1147 datasets originally distributed alongside the statistical software environment R and some of its add-on packages.

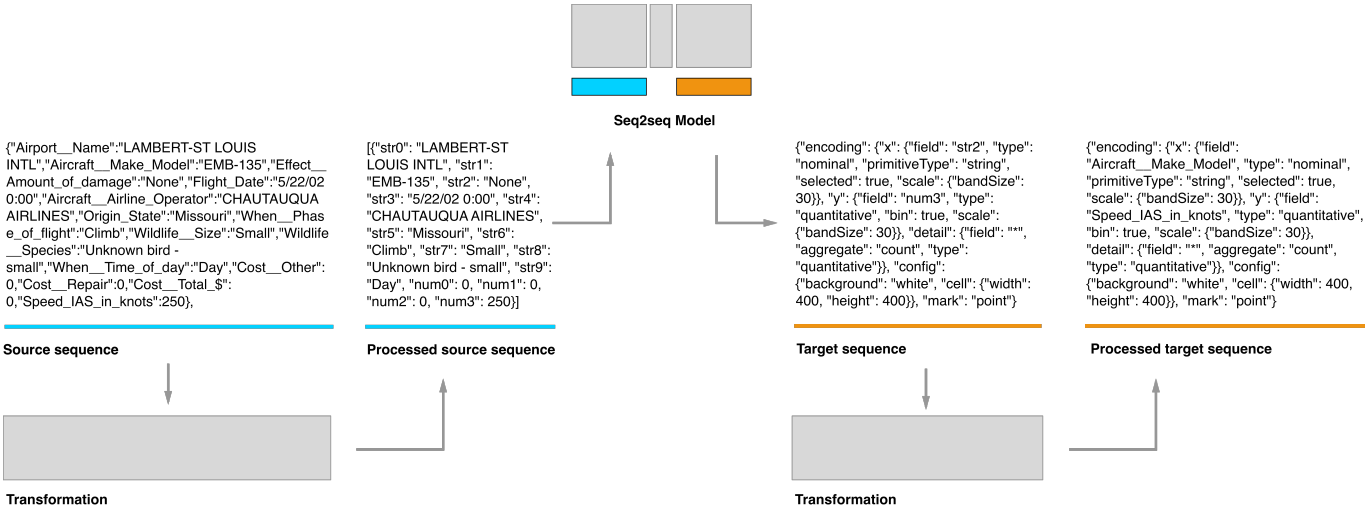

Figure 4. Transformations over source and target sequences. Both sequences are in JSON format.

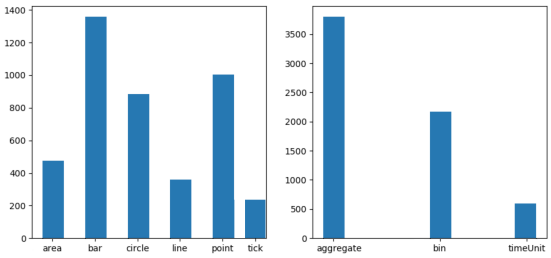

Figure 5. Frequency of the Vega-Lite mark types and transforms used in our training examples.

valid JSON file and valid Vega-Lite specification with some minor failure cases (Figure 9 ■).

Web Application Integrating Data2Vis

To further evaluate the utility of our model, we developed a web application prototype interface (Figure 6) that supports the use case of an analyst exploring data similar to [66, 67]. The interface supports 3 primary interactions - data import, visualization generation and visualization update. First, the analyst is able to import a dataset into the application. They can do this by using the "load dataset" button which loads a randomly selected dataset from the Rdataset repository or they can directly paste a JSON data array in the provided text field. Next, the analyst can select the "generate" button which submits the dataset to our model, receives a Vega-Lite specification (placed in a text field) and renders the plot. Finally, the analyst can update to the generated specification by modifying the Vega-Lite specification and selecting the "update visualization" button to render a updated specification. We showed this early prototype to two visualization researchers and they were able to quickly build on the specifications generated by the model, making changes to field selections and transformations.

DISCUSSION

We presented the very first attempt to transform data to visualizations using a deep neural network, applying a neural machine translation (seq2seq) model in automating visualization generation. Below, we discuss the potential impact of our approach and limitations of the current Data2Vis model along with future research directions.

Impact and Use Case

Making Visualization Authoring Easier Providing users with little or no programming experience with the ability to rapidly create expressive data visualizations empowers users and brings data visualization into their personal workflow. Data2Vis holds potential to make data visualization more accessible, speed-up the visualization authoring process and augment the visualization capabilities of all users.

Accelerating Data Exploration For visualization experts, it is likely that visualizations created by Data2Vis may be insufficient for their needs. This is especially true when the structure of the data being analyzed is unknown or unusual and effective exploration of the data requires complex transforms as well as deep domain expertise. However, Data2Vis can contribute to this process by "jumpstarting" the visualization—first by generating a valid visualization specification and *seeding* the creation process with an initial visualization. Analysts can initialize their visualization tasks with Data2Vis and iteratively *correct* its content while generating intermediate visualizations.

Limitations

Field Selection and Transformation The current version of our model has limitations which occur in about 15-20% of tests. First, the model occasionally selects what we refer to as a phantom field (a field that does not exist in the input dataset) as part of the visualization spec it generates (Figure 9). While plots are still realized in some cases despite this error (Vega-Lite incorporates good defaults), the affected axis is not interpretable. Another limitation of the model is observed in selecting fields (attributes) of the input data to visualize

Figure 6. Web application interface for Data2Vis qualitative evaluation. (a) A user can either load a random dataset from the RDdataset collection or paste a dataset (JSON format) and select "Generate." (b) Data2Vis generates a Vega-Lite specification based on the dataset. (c) The user can modify the Vega-Lite specification and update the visualization.

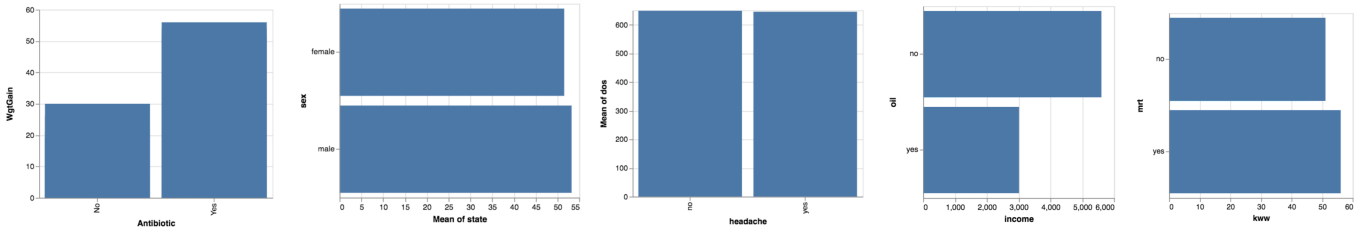

Figure 7. Examples of visualizations where the model has learned common selection patterns and leverages concepts such as responses (yes, no) and sex (male, female).

— the model sometime selects fields that are unintuitive or have little information value. For example, a frequency plot of grouped longitude and latitude values does not provide much information. Finally, the model generates relatively simple visualizations — univariate plots (which can serve as data field summaries) and bivariate plots. It is unable to apply complex transforms, use multiple variables.

Training Data While further experimentation is required, our intuition is that the limitations mentioned above reflect limitations in both the size and diversity of our training data. Our goal with Data2Vis was to evaluate the viability of machine translation in generating valid visualization specifications, we have conducted our experiments with a relatively small dataset (4300 examples upsampled to 215,000 training pairs). While our current results provide insights, we believe a larger and more diversified training dataset will improve learning and model generalization. Another limitation with our training data is related to our training pair generation strategy. Currently, we construct our source tokens from a single row from a dataset which is then preprocessed before training. While this approach shortens the input sequence length and enables us train our model, the model can only learn properties of each field (e.g. length, content, numeric type, string type) as opposed to properties of the distribution of the field (e.g mean, range, categories etc.) which encode useful signals for data visualization.

Future Work

Eliciting More Training Data Naturally, addressing limitations with our training data constitutes the next step for future work. We plan to conduct a structured data collection aimed at generating visualization examples across a large number of datasets, visualization types (bar, point, area, chart etc),

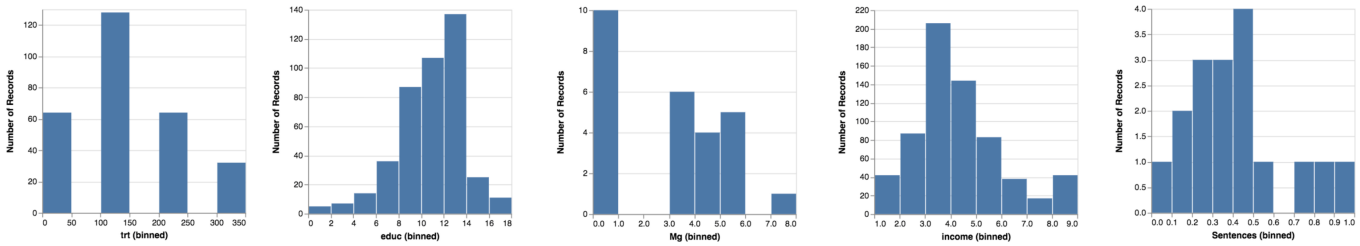

Figure 8. Examples where the model has learned to generate univariate plots that summarize fields selected from the dataset.

transforms, complexity (number of variables), interactions and visualization languages. We will also explore strategies to improve the training process that guide the model towards learning properties of the distribution for a given field.

Extending Data2Vis to Generate Multiple Plausible Visualizations Data2Vis is currently implemented as a sequence to sequence translation model. Sequence models work very well for domains where it is desirable to have fixed mappings of input sequences to output sequences (text summarization, image captioning, language translation, etc). It is generally expected that a sentence in one language always maps to the same sentence in a different language, and acceptable if a passage always maps to the same summary or an image to the same caption. However, when applied to the task of data visualization, it is desirable that input data maps to *multiple* valid visualizations. Thus, another avenue for future work is to explore the design of conditioned sequence models that enable such *one to many sequence* mappings between data and visualization specifications.

Targeting Additional Grammars Building on results from Data2Vis, important next steps also include efforts to train models that can map input data to multiple different visualization specification languages, including ggplot2, given a dataset. This line of research may also explore training models that learn direct mappings between different visualization specification languages, enabling visualization specification reuse across languages and platforms.

Natural Language and Visualization Specification We propose the exploration of models that generate visualizations conditioned on natural language text in addition to datasets. A potential approach is to first explore how users might describe or express visualizations for a given dataset and use this knowledge in generation of triplets—natural language description, data sequence, and visualization specification. These data points can then be leveraged in training a model that learns to generate visualizations based on natural language descriptions. These models can extend the expressive capabilities of existing systems that integrate multimodal interactions and visualizations for exploring data. Conversely, we can use textual descriptions of visualizations to automatically generate captions for them, akin to image caption generation (e.g., [33, 60, 69]).

CONCLUSION

The history of data visualization is rich with work that treats visualization from a linguistic perspective. Bertin systematized data visualization as "a language for the eye" [7]. Adopting Bertin's analogy, Mackinlay [43] viewed visualizations as sentences of a graphical language and formalized a model based on "expressiveness" and "effectiveness" criteria, borrowing concepts from formal languages. Subsequent research also introduced various "grammars" of visualization specification.

We significantly extend this earlier perspective and formulate data visualization as a sequence to sequence translation problem where we translate data specifications to visualization specifications. We train a deep sequence to sequence model and demonstrate its efficacy generating univariate and bivariate plots. We also identify initial failure conditions, offer ideas for their remediation and an agenda for future work.

It is our belief that the problem formulation and model presented in this work represents an appropriate baseline for future work in automated generation of visualizations using deep learning approaches. Our approach sets the stage for systems that learn to generate visualizations at scale with implications for the development of guided visual data exploration systems.

ACKNOWLEDGMENTS

We thank the authors of [46] for making the Vega-Lite corpus available.

REFERENCES

1. MacEachren M. Alan. 1995. *How Maps Work: Representation, Visualization, and Design*. Guilford Press.

2. Daniel Asimov. 1985. The Grand Tour: A Tool for Viewing Multidimensional Data. *SIAM J. Sci. Stat. Comput.* 6, 1 (1985).

3. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. (sep 2014). `http://arxiv.org/abs/1409.0473`

4. Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2016. DeepCoder: Learning to Write Programs. *CoRR* abs/1611.01989 (2016).

5. Shortridge B. Barbara. 1982. Stimulus processing models from psychology: can we use them in cartography? *The American Cartographer* 9 (1982), 155–167.

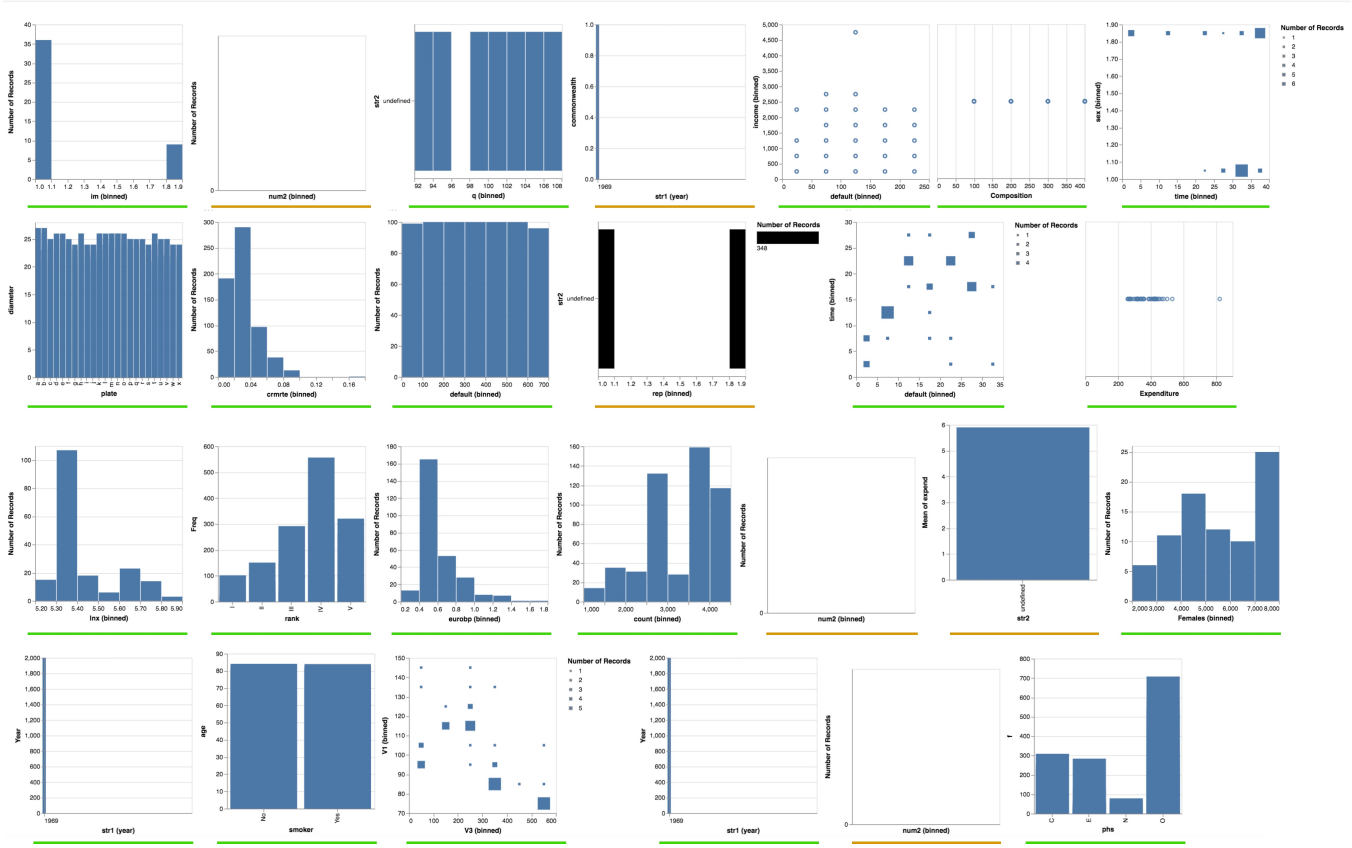

Figure 9. A collection of random examples generated by the model from the test dataset. These examples demonstrate initial results on the model's capabilities in generating plausible visualizations ■ and its current limitations (phantom variables ■).

6. Tony Beltramelli. 2017. pix2code: Generating Code from a Graphical User Interface Screenshot. *arXiv preprint arXiv:1705.07962* (2017).
7. Jacques Bertin. 1983. *Semiology of Graphics*. University of Wisconsin Press.
8. Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. 2015. Alternative structures for character-level RNNs. *CoRR* abs/1511.06303 (2015). `http://arxiv.org/abs/1511.06303`
9. Michael Bostock and Jeffrey Heer. 2009. Protovis: A Graphical Toolkit for Visualization. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2009).
10. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011).
11. Fatma Bouali, Abdelheq Guettala, and Gilles Venturini. 2016. VizAssist: an interactive user assistant for visual data mining. *Vis. Comput.* 32, 11 (2016), 1447–1463.
12. Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392* (2012).
13. Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints* (March 2017).
14. Stephen M. Casner. 1991. Task-analytic approach to the automated design of graphic presentations. *ACM Trans. Graphics* 10, 2 (1991), 111–151.
15. Xinyun Chen, Chang Liu, and Dawn Song. 2018. Tree-to-tree Neural Networks for Program Translation. *CoRR* abs/1802.03691 (2018). `http://arxiv.org/abs/1802.03691`
16. Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* abs/1406.1078 (2014). `http://arxiv.org/abs/1406.1078`
17. Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* abs/1406.1078 (2014).
18. William S. Cleveland and Robert McGill. 1984. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *J. Amer. Statist. Assoc.* 79, 387 (1984), 531–554.

19. Çağatay Demiralp, Peter J. Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. 2017. Foresight: Recommending Visual Insights. *Proc. VLDB Endow.* 10, 12 (2017), 1937–1940.
20. Çağatay Demiralp, Carlos Scheidegger, Gordon Kindlmann, David Laidlaw, and Jeffrey Heer. 2014. Visual Embedding: A Model for Visualization. *IEEE CG&A* (2014).
21. Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. 2017. RobustFill: Neural Program Learning under Noisy I/O. *CoRR* abs/1703.07469 (2017). http://arxiv.org/abs/1703.07469
22. Douglas Eck and Juergen Schmidhuber. 2002. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* 103 (2002).
23. Mary Anne Fisherkeller, Jerome H. Friedman, and John W. Tukey. 1974. PRIM-9: An interactive multidimensional data display and analysis system. In *Proc. Fourth International Congress for Stereology*.
24. Felix A. Gers, JÃijrgen Schmidhuber, and Fred Cummins. 1999. Learning to Forget: Continual Prediction with LSTM. *Neural Computation* 12 (1999), 2451–2471.
25. David Gotz and Zhen Wen. 2009. Behavior-driven visualization recommendation. In *ACM IUI*. 315–324.
26. David Ha and Douglas Eck. 2017. A Neural Representation of Sketch Drawings. *CoRR* abs/1704.03477 (2017). http://arxiv.org/abs/1704.03477
27. Pat Hanrahan. 2006. Vizql: a language for query, analysis and visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, 721–721.
28. Jeffrey Heer. 2013. CS448B: Data Visualization, Stanford University. Course Slides. (2013).
29. Sepp Hochreiter and JÃijrgen Schmidhuber. 1997. Long Short-term Memory. 9 (12 1997), 1735–80.
30. Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).
31. Daniel D Johnson. 2017. Generating polyphonic music using tied parallel networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 128–143.
32. Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1700–1709.
33. Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3128–3137.
34. Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and Understanding Recurrent Networks. *CoRR* abs/1506.02078 (2015). http://arxiv.org/abs/1506.02078
35. Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *CoRR* abs/1710.10196 (2017). http://arxiv.org/abs/1710.10196
36. Alicia Key, Bill Howe, Daniel Perry, and Cecilia Aragon. 2012. VizDeck. In *ACM SIGMOD*, Vol. 681–684.
37. Gordon Kindlmann and Carlos Scheidegger. 2014. An Algebraic Process for Visualization Design. *IEEE TVCG* 20 (2014), 2181–2190.
38. Stephan Lewandowsky and Ian Spence. 1989. Discriminating strata in scatterplots. *Journal of American Statistical Association* 84, 407 (1989), 682–688.
39. Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent Predictor Networks for Code Generation. *CoRR* abs/1603.06744 (2016).
40. Minh-Thang Luong and Christopher D. Manning. 2016. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. (apr 2016). http://arxiv.org/abs/1604.00788
41. Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. (aug 2015). http://arxiv.org/abs/1508.04025
42. Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206* (2014).
43. Jock Mackinlay. 1986. Automating the design of graphical presentations of relational information. *ACM Trans. Graphics* 5, 2 (1986), 110–141.
44. Jock Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show Me: Automatic Presentation for Visual Analysis. *IEEE TVCG* 13, 6 (2007), 1137–1144.
45. Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. 2016. Neuro-Symbolic Program Synthesis. *CoRR* abs/1611.01855 (2016). http://arxiv.org/abs/1611.01855
46. Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum (Proc. EuroVis)* (2017).
47. Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396* (2016).

48. Steven F. Roth, John Kolojechick, Joe Mattis, and Jade Goldstein. 1994. Interactive graphic design using automatic presentation knowledge. In *ACM Human Factors in Computing Systems (CHI)*.

49. Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2017).

50. Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. 2016. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2016).

51. Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891* (2016).

52. Jinwook Seo and Ben Shneiderman. 2004. A Rank-by-Feature Framework for Unsupervised Multidimensional Data Exploration Using Low Dimensional Projections. In *Procs. InfoVis*. 65–72.

53. Roger N. Shepard. 1987. Toward a Universal Law of Generalization for Psychological Science. *Science* 237 (1987), 1317–1323.

54. Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless data exploration with Zenvisage. *PVLDB* 10, 4 (2016), 457–468.

55. Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics* 8, 1 (2002), 52–65. DOI:http://dx.doi.org/10.1109/2945.981851

56. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *CoRR* abs/1409.3215 (2014). http://arxiv.org/abs/1409.3215

57. Bo Tang, Shi Han, Man Lung Yiu, Rui Ding, and Dongmei Zhang. 2017. Extracting Top-K Insights from Multi-dimensional Data. In *ACM SIGMOD*. 1509–1524.

58. Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. 2015a. Towards Visualization Recommendation Systems. In *DSIA Workshop*.

59. Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015b. SeeDB: Efficient Data-driven Visualization Recommendations to Support Visual Analytics. *PVLDB* 8, 13 (2015), 2182–2193.

60. Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 3156–3164.

61. Hadley Wickham. 2010. A layered grammar of graphics. *Journal of Computational and Graphical Statistics* 19, 1 (2010), 3–28.

62. Leland Wilkinson. 1999. *The Grammar of Graphics* (1st ed.). Springer.

63. Leland Wilkinson, Anushka Anand, and Robert Grossman. 2005. Graph-theoretic scagnostics. In *Proc. InfoVis*. 157–164.

64. Graham Wills. 2017. Brunel v2.5. https://github.com/Brunel-Visualization/Brunel. (2017). Accessed: 2018-04-04.

65. Graham Wills and Leland Wilkinson. 2008. AutoVis: Automatic visualization. *Info. Visual.* 9, 1 (2008), 47–69.

66. Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE TVCG* 22, 1 (2016), 649–658.

67. Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. In *ACM CHI*.

68. Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

69. Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. 2048–2057.

70. Pengcheng Yin and Graham Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. *CoRR* abs/1704.01696 (2017).