Data2Vis

Automatic Generation of Data Visualizations Using Sequence-to-Sequence Recurrent Neural Networks.

Victor Dibia
Research Engineer, Cloudera Fast Forward Labs

Çağatay Demiralp
Megagon Labs

March 21, 2019
Why Automate Visualizations?
Hint: We want to augment users (new capabilities, improved quality/speed).
Charts can make data more accessible

- Reduced cognitive load
- Effective and expressive

Compared to tabular representations of data ..
Creating visualizations is **EFFORTFUL**.
Visualization authoring is effortful

1. Hypothesis
2. Visual Encoding
3. Implementation (code)

- Generating **hypothesis** and questions regarding data.
- Identifying appropriate **visual** encoding strategies (chart type, data transformations etc.) that support hypothesis.
- Writing **code** to implement visualizations
Visualization authoring is effortful

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Visual Encoding</th>
<th>Implementation (code)</th>
</tr>
</thead>
</table>

Effective visual encoding and implementation can be challenging for novice users.

- Many (novice) **users lack the skills** to select appropriate visual encodings and to write code that implement visualizations.
- **Automated approaches** can help (augment) with tasks 2 and 3.
More so ..

Visualization Recommendation
(CompassQL, Voyager 2, VizML)

Visualization Ranking
(VizDeck, Draco, Deep Eye)

• Existing approaches to automated viz are **limited**.
  • Depend on **heuristics** and **hand engineered features** which need to be manually updated.
  • Does not leverage knowledge codified within **existing** visualization examples.
A Scalable, learning based approach?

Data2Vis

a (deep) learning based approach to automated visualization

Approach

• Formulate visualization authoring as a machine learning problem.
• Identify data sampling strategies that enable training with (limited) data
• Design metrics that enable evaluation of models
• Present a model that learns to map raw data to generated visualizations.
• Declare that we have solved AGI.
Related Work
Automated Visualization Tools
Automated Visualization

<table>
<thead>
<tr>
<th>Voyager2</th>
<th>Draco</th>
<th>Deep Eye</th>
<th>VizML</th>
</tr>
</thead>
</table>

**Voyager2**
Recommend visualizations based on partial specifications provided by users.

**Draco**
Modeling visualization design knowledge as a collection of constraints, learn weights for soft constraints from experimental data.

**Deep Eye**
Uses learned binary classifier + learning to rank algorithms to rank visualizations as “Good or Bad” based on examples.

**VizML**
Train a model to predict parts of visualization specifications - visual encoding choices (x,y axis) - using hand engineered features.
Neural Synthesis Models
DNNs for Neural Synthesis

Sketch RNN (Ha et al, 2017)
DNNs for Neural Synthesis

Models that learn human-like creative processes.

• SketchRNN: Generate Stroke based drawings for common objects (Ha et al 2017)

• Text to image synthesis. (Reed et al 2016)

• Google Smart Compose and Smart Reply (Kannan et al 2016)
Code Generation Models
DNNs for Code Generation

Models that learn to generate code.

- **Domain Specific Language Translation** (Yin et al 2017, Zhong et al, 2017)
- **Natural Language to SQL** (Dong & Lapata 2016, Zhong et al, 2017)
- **TCG (trading card games) to Python and Java Language specification.** (Ling et al 2016)

<table>
<thead>
<tr>
<th>Pick #</th>
<th>CFL Team</th>
<th>Player</th>
<th>Position</th>
<th>College</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Hamilton Tiger-Cats</td>
<td>Connor Healy</td>
<td>DB</td>
<td>Wilfrid Laurier</td>
</tr>
<tr>
<td>28</td>
<td>Calgary Stampeders</td>
<td>Anthony Forgone</td>
<td>OL</td>
<td>York</td>
</tr>
<tr>
<td>29</td>
<td>Ottawa Renegades</td>
<td>L.P. Ladouceur</td>
<td>DT</td>
<td>California</td>
</tr>
<tr>
<td>30</td>
<td>Toronto Argonauts</td>
<td>Frank Hoffman</td>
<td>DL</td>
<td>York</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Table: CFLDraft**

**Question:**

How many CFL teams are from York College?

**SQL:**

```
SELECT COUNT CFL Team FROM CFLDraft WHERE College = "York"
```

**Result:**

2
Neural Machine Translation Models
DNNs for Machine Translation

• Family of Encoder-Decoder Models that learn mappings from an input sequence to an output sequence. (Britz et al 2017)

• Frequently referred to as Seq2Seq models, but have applications for non-sequential problems e.g. Image Captioning, Text Summarization, Code Generation.

• Non-sequential applications are enabled by Bi-Directional RNNs and Attention Mechanisms
BiDirectional RNNs

• Consists of both a **forward RNN** (reads input sequence and calculates forward hidden states) and a **backward RNN** (reads input sequence in reverse order and calculates backward hidden states). (Shuster et al 1997).

• Generates an hidden state that is a concatenation of both forward and backward RNNs.
DNNs for Machine Translation

Attention Mechanism

Allows a model to focus on aspects of an input sequence while generating output tokens

○ Makes translation models robust to performance degradation while generating lengthy sequences.
○ Enables the learning of mappings between source and target sequences of different lengths.
○ Allows for interpretability explorations.
Model
## Problem Formulation

<table>
<thead>
<tr>
<th>Data</th>
<th>Visualization Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data in JSON</td>
<td>{ &quot;country&quot;: &quot;AUS&quot;, &quot;il&quot;: &quot;0.058&quot; ..}</td>
</tr>
</tbody>
</table>

- Formulate as a **neural translation problem** (sequence to sequence models).

- Learn mappings from raw data to visualization specification in an End-to-End trainable task.
Model Input

Data

Input data in JSON | { "country": "AUS", "il": "0.058" .. }

Visualization Specification

Vega Lite Spec | { "y": {"field":"gender", "type": .. } }

JSON Data (Non-nested)

[{"Time":"152","size":"4.51","treat":"ozone","tree":"1"},{"Time":"174","size":"4.98","treat":"ozone","tree":"1"},{"Time":"201","size":"5.41","treat":"ozone","tree":"1"},{"Time":"227","size":"5.9","treat":"ozone","tree":"1"},{"Time":"258","size":"6.15","treat":"ozone","tree":"1"},{"Time":"152","size":"4.24","treat":"ozone","tree":"2"},{"Time":"174","size":"4.2","treat":"ozone","tree":"2"},{"Time":"201","size":"4.68","treat":"ozone","tree":"2"},{"Time":"227","size":"4.92","treat":"ozone","tree":"2"},{"Time":"258","size":"4.96","treat":"ozone","tree":"2"},{"Time":"152","size":"3.98","treat":"ozone","tree":"3"},{"Time":"174","size":"4.36","treat":"ozone","tree":"3"},{"Time":"201","size":"4.79","treat":"ozone","tree":"3"},{"Time":"227","size":"4.99","treat":"ozone","tree":"3"},{"Time":"258","size":"5.03","treat":"ozone","tree":"3"},{"Time":"152","size":"4.36","treat":"ozone","tree":"4"}]}
Model Output

Data
Input data in JSON | `{ "country": "AUS", "il": "0.058" ..}`

Visualization Specification
Vega Lite Spec | `{ "y": {"field":"gender", "type": ..}}`
Mapping

Data
Input data in JSON

```json
{ "country": "AUS", "il": "0.058" }
```

Visualization Specification

Vega Lite Spec

```json
{ "y": {"field":"gender", "type": ..} }```

```
[{
  "Time": "1992",
  "size": "4.51",
  "treat": "ozone",
  "tree": "1"
},
...
...
...

{"Time":"1993","size":"4.98","treat":"ozone","tree":"1"}]
```

```
{"encoding": {
  "detail": {
    "type": "temporal",
    "timeUnit": "week",
    "field": "Time"},
  "x": {
    "type": "quantitative",
    "field": "size",
    "bin": true},
  "y": {
    "aggregate": "count",
    "field": "**",
    "type": "quantitative"},
  "mark": "area"}
```
Training Data

• 4300 Vega Lite specifications based on 11 datasets generated using CompassQL (Poco et al 2017).

• CompassQL is based on
  • Heuristics and rules which enumerate cluster and rank visualizations according to known data properties and perceptual considerations.
  • Filtered manually to remove problematic instances
  • 1-3 variables per chart, multiple chart types.
  • “MNIST” for automated visualization experiments
Sampling strategy

- Repetitive sampling of datum to visualization.

- Training data is generated by sampling examples
  - Training pair consists of single row from dataset (JSON) and visualization specification (JSON)
  - 50 random pairs selected from each example
  - Data normalized (replace field names with normalized values e.g. str0, str1, num0, num1)
  - 215k pairs after sampling
Training Data Pair

Data
Input data in JSON | 
| { "country": "AUS", "il": "0.058" ..} |

Visualization Specification
Vega Lite Spec | 
| { "y": {"field":"gender", "type": ..} |

{"Time":"1993","size":"4.51","treat":"ozone","tree":"1"}

{"encoding": {"detail": {"type": "temporal", "timeUnit": "week", "field": "Time"}, "x": {"type": "quantitative", "field": "size", "bin": true}, "y": {"aggregate": "count", "field": "," type": "quantitative"}}, "mark": "area"}
Training Data Transformation

**Data**
Input data in JSON: `{ "country": "AUS", "il": "0.058" .. }`

**Visualization Specification**
Vega Lite Spec: `{ "y": {"field":"gender", "type": .. }`

• We apply transformations to the data, replace numeric, string and date fields with short forms num0, str0, dt0.

```
{ "dt0": "152", "num0": "4.51", "str0": "ozone", "num1": "1" }
```

```
{ "encoding": { "detail": { "type": "temporal", "timeUnit": "week", "field": "dt0" }, "x": { "type": "quantitative", "field": "size", "bin": true }, "y": { "aggregate": "count", "field": "*", "type": "quantitative" }, "mark": "area" }
```
Training Data Transformation

Data
Input data in JSON | { "country": "AUS", "il": "0.058" ..}  

Visualization Specification
Vega Lite Spec | { "y": {"field":"gender", "type": ..} }

• We apply transformations to the data, replace numeric, string and date fields with short forms num0, str0, dt0.

• Transformation provides following benefits
  • Reduce overall vocabulary size
  • Prevent LSTMs from learning specific field names
  • Reduce overall sequence length (faster training, less memory)
Model

Encoder/Decoder Model

Model based on architecture by Britz et al 2017
Model Training

- Character tokenization strategy
- Dropout Rate of 0.5
- Fixed learning rate (0.0001)
- Adam optimizer
- 20,000 steps
- Final log perplexity of 0.032
- Maximum seq length of 500

The training code is based on the Google Seq2Seq implementation (Britz et al 2017)
CLOUDERA DATA SCIENCE WORKBENCH
Accelerate Machine Learning from Research to Production

For data scientists

- **Experiment faster**
  Use R, Python, or Scala with on-demand compute and secure CDH data access

- **Work together**
  Share reproducible research with your whole team

- **Deploy with confidence**
  Get to production repeatably and without recoding

For IT professionals

- **Bring data science to the data**
  Give your data science team more freedom while reducing the risk and cost of silos

- **Secure by default**
  Leverage common security and governance across workloads

- **Run anywhere**
  On-premises or in the cloud
- Docker/Kubernetes based
- Analyze your data
- Train models (run, track, compare)
- Deploy APIs
- Multi tenant, collaborative, secure
Evaluation
Model Evaluation

• Diagnostic Metrics

• **Language Syntax Validity**
  A measure of how well the model learns the rules of the visualization language (JSON).
  % of all generated examples that are valid JSON

• **Grammar Syntax Validity**
  A measure of how well the model learns the visualization grammar (Vega Lite).
  % of all generated examples that compile in Vega Lite.
Beam Search Decoding

• Expands all possible next steps and keeps the k most likely, where k is a user-specified parameter.

• We leverage beam search in generating diverse specifications based on same data.
Qualitative Results

Model learns to generate multivariate and bivariate plots

Model is shown random data from the Rdataset collection not used in training.
Qualitative Results

Model learns to perform selections using categorical fields (yes/no, male/female, state, country etc.)

Model is shown random data from the Rdataset collection not used in training.
Qualitative Results

Beam search decoding (k=15) generates diverse chart types (bar, area, line)

Model is shown random data from the Rdataset collection not used in training.
Quantitative Evaluation

• Evaluation Metrics
  • We train 3 models -
    ■ No attention (Bidirectional),
    ■ Attention (Unidirectional),
    ■ Attention (Bidirectional)

• Test each model with various values for beam width $k$ (5, 10, 15).

• Compute mean metric for 100 randomly selected datasets from the Rdataset collection.
Quantitative Evaluation

- All models learn to generate valid JSON syntax.
- Bidirectional models perform better than unidirectional models on both metrics on the average.
- Attention based models do significantly better on Grammar metric.
- Attention based BiDirectional Models (with beam width $k=15$) have the best performance for generating valid “plotable” Vega Lite specifications.
Attention Plots

Attention plots show the model learns to pay attention to input data in generating aspects of visualization specification.

Example attention plots for a visualization generation case (a) Model learns to pay attention to field name "str" in generating the "string" field type applied to the field. (b) Model learns to pay attention to the field name "num0" and its value in specifying the "quantitative" field type applied to the field.
Summary

• **Limited Training Data**
   Our current dataset, while sufficient for demonstrating the viability of our approach, has limited coverage of real world use cases.

• **Phantom Fields**
   In 10~20% of cases, the model generates specifications with fields not in the dataset. In practice we can detect this at runtime and keep exploring beam search generation until valid specs are generated.
Evaluation
Limitations

• **Training Data**
  Our current dataset, while sufficient for demonstrating the viability of our approach, has limited coverage of real world use cases.

• **Conditioned Generation**
  Current model does not support conditioned visualization generation.

• **Phantom Fields**
  In 10~20% of cases, the model generates specifications with fields not in the dataset. In practice we can detect this at runtime and keep exploring beam search generation until valid specs are generated.
- Ofcourse ... there are failure cases!
Future Work

• **Additional Data Collection.**
  Curating a more diverse dataset that enables training a more robust model.

• **Extending Data2Vis to Generate Multiple Plausible Visualizations.**
  Explore approaches (e.g. conditioned GANs or VAEs) to train a model that generates multiple valid visualizations with specified conditions.

• **Targeting Additional Grammars**
  Training models that map input data to multiple different visualization specification languages (e.g. Vega, ggplot2, D3 etc.).

• **Natural Language and Visualization Specification**
  Training models that generate visualizations based on natural language text and input data.

• **Browser deployment**
  Javascript library that provides fast generation in the browser.
Summary

Formulating data visualization as a sequence to sequence problem works well. The following insights were useful.

- Transformations which scaffold the learning process.
- Bidirectional RNNs which significantly enable learning complex non-sequential mappings.
- Repetitive sampling and beam search decoding for multiple visualization generation.
Contributions

• Formulate automated visualization as a neural translation problem (map data to visualization specifications)
• End-to-End trainable model for visualization generation
• Training strategy (data generation, transformations etc.)
• Metrics for evaluating End-to-End visualization generation systems
• Sample code and demo

https://github.com/victordibia/data2vis
Code and Trained Model

Github: https://github.com/victordordibia/data2vis

Thanks.

Victor Dibia
victor.dibia@gmail.com