Procedural Knowledge Libraries for Data Visualization Engineers

Abstract

This work explores the concept of procedural knowledge libraries tailored for data visualization engineers. It discusses the dynamic nature of such libraries, the importance of capturing procedural metadata, and the implications for knowledge preservation and retrieval. The paper also examines existing tools and methodologies, highlighting their limitations and proposing a framework for more effective procedural knowledge management.

Table of Contents

1 Introduction

Initially, this centered around procedural knowledge as a whole entity and now it's information about the process of building new knowledge itself. Explicating these processes makes it possible to audit [human/machine] reasoners. You can imagine having repos of procedural knowledge for those analyzing (building models of) the same dataset. This framework would then make it possible to assess the mechanistic differences between their approaches, clearly defining the branches of thought that took place.

The important distinction to make here is that libraries are systems; they are dynamic, as the contents within them are living. To date, traditional "libraries" have assumed otherwise. Furthermore, the objects inside them are defined as an artifact of knowledge. That is, artifacts that are the product of thought and developed with a goal in mind. The function of the system is then to be a home to these artifacts and preserve them for prosperity's sake, but more importantly for retrieval's sake. As such, while the integrity of the work should be preserved, there are ways to build the system and store the objects such that this can be done.

Provenance refers to the lineage of an object. Provenance, synonymous with metadata, has been developed with the static knowledge artifact in mind, so extending upon it for the sake of the dynamic artifact is key for a "new age of libraries."

Now, we are faced with the question of what it means to develop new knowledge. That is, what is something tractable than we can anchor to, to help build better libraries: libraries that work for knowledge that evolves over time. I use the instance of model-building. While there's no dedicated library to models, they exist within research papers, repositories, and notebooks. Here, I outline the ways that we can capture how they are developed (e.g. improved version control, collaboratively, integrating "layered context architecture"), what they mean (i.e. to help people find exactly what they need by explicating what models and their requests mean), what it looks like to retrieve them (the ideal interface to help people search in accordance with their preferences), etc. And then I consider how they may fit into a larger system: how they may be cataloged, organized (e.g., the metadata that can be assigned to them, scalable metadata attribution, etc.), and stored (e.g. internet search engine that works for "findings.").

Here, we focus on the model-builder who is building graphical models or visualizations. Visualization is often the last step in conveying insight and analysis of data that has been obtained through the research process. Therefore, within this

test-bed, we lay out a framework and suggest the limitations and trade-offs in real-world application (i.e. Log-based CRDTs being computationally expensive than traditional CRDTS), concluding with discussion on promising speculative and existing areas of work to address them.

2 Status quo

- File versioning where you decide what's a version. For example, naming versions of a project as "v1, v2, v3." Most document libraries have some version of crude file records. arXiv, the preprint repository will keep versions of a paper as they're re-uploaded with edits from the authors. Google Scholar indexes the versions of a paper and will link it to the same citation.
- Versioning with a Distributed Version Control System (DVCS) such as Git where you have logs of your past commits (that make your versions) and cloud DVCS where you're able to collaborate with others. Here, there's not only versions pertinent to the changes that you make, but the changes that other people make to your instance of the project, forked versions of your project, and the changes that their collaborators make to those instances.
- There's a number of domain-specific projects and tools, such as CodaLab where machine learning researchers can host and run their experiments (reproducible) and share their papers in executable form as worksheets.
- For visualization workflows–the primary application of this project–there's VisTrails, a provenance tracker for visualizations where data is stored in XML files. With it, users can query workflows, as the system tracks decisions and data products to help answer the actions that led to a result. It's core focus is reproducibility. It's mainly developed for Python and Python-based visualization libraries.

Perplexity trail (could VizTrail work for a UMAP model and viz? What are its capabilities?)

3 Overview

- Broadening the scope of the library
  - Procedural knowledge libraries are living and they're archives of "dynamic" reasoning.
  - There should not be a distinction because even static artifacts take time to make and evolve over time. Even if we have static artifacts our archives should be dynamic to account for development.
  - Preserving the right to "priviledged information."

  - Trails are now included in the corpus of knowledge and librarians should be able to navigate them and they should be included in the set of things that we want to query to answer the questions that we pose. In other words, including the drafted states means you can have "end-to-end" libraries: libraries of reasoning and libraries of artifacts.
  - This won't come without intervention, but then how do you translate the "what" and "how" of development into the "why?" Next, why does it matter if you can do that?
  - Barriers to tracking history
    * Problems with CRDTs
      · Eg-walker algorithm as alternative: https://arxiv.org/pdf/2409.14252 (allows for fine-grained editing history, mentioned the promise in being extended to rich-text, graphics, and other applications; HN thread suggests that this would need an additoinal CRDT implemented for rich-test)
      · (From thread) might be fruitful to look at what exists for photoshop and similar software: where layers can cam be inserted/deleted and as a result the index of following items update. And then there's a question of how layer re-orders would work. (a set of operations, even graph where each event corresponds to an operation, replay, apply/retreat/advance methods for eddicient replay)
      · If you have an advanced CRDT say XCRDT which represents the internal structure in your work then there are way ways to combine the original operations with the CRDT operations.
    * The non-linearity of history (Log-based CRDTs: https://sites.cs.ucsb.edu/ ckrintz/papers/ic2e22.pdf)
    * Format / interoperabilityof historical data
    * Key parameters of queries/history/metadata
    * This is useful because you can distinguish goals and deviate from these goals (set out by others or your past self) in pointed, clear ways.
- The implications of broadening the scope:
  - It takes more to train to become a librarian. The scope of the librarian also evolves. The librarian is always the human but machines can be tools to help us become better ones (work faster, distributively, and more effectively)
  - Engineering and research questions that stem from this, especially on the implementation side.

- The different layers of context: action-layer, procedural-layer, narrative-layer, object/output-layer (reconstructing by laying these on top of each other and have them span the space of time in the horizontal direction) –> assuming this takes the form of a database
    - From CGPT: "You have 4-D GithHub (model = code (action) + reason/method/strategy (procedural intent, step in readable language, what someone else could theoretically carry out) + interpretation (why the action was done, what outcome was inferred from the reason for the code, reason for the strategy)compared to 2-D GitHub (action (what was done) + object (the output))"
        - 4-D (new): Model = Action + Rationale + Interpretation + Object
        - 2-D (old): Model = Action + Object
        - **Reconstructing narratives from the procedural information: turning procedural information into knowledge**

4 How to read this work. Listing the organization of the paper.

5 Structure and points

- Redefine what is a library in terms of its function and its form
    - This assumes that there's some core, instrinsic properties that make it a library and the rest of the variables we can play with according to what's nee
- There's a gap in the way that we currently store knowledge
    - What provenance do we already capture?
- How we manage information might matter more than just the sheer amount of knowledge that we have
- Procedural knowledge is the one we try the least to preserve but it holds a wealth of insight
- How do you interact with this kind of procedural knowledge as a librarian?
    - That is, how do you query and what does retrieval look like? (This seems unexplored).

  Trying to quantify the current (tacit) knowledge loss when you don't log

6 Motivations & Real-World applications

- More than just capturing procedural knowledge how do you make it operational.
- Being able to have "structured procedural metadata'' for search in internet research libraries. For example, you could have methodology-based queries. For a tool like Elicit, it would be possible to search for papers that use specific methods, had margins or metric results within these ranges, and so on: this would make it easier to conduct meta-analyses.
- Being able to abstract away the implementation completely and pick between different workflows on the output-layer.
- Procedural knowledge libraries are effectively tutorials–they allow people to do the things you've done beyond authentication.
- We can get better understandings of systems and do summary statistics and make opinions and judgements about the world.
- Help interpreters understand how the "formalism" and/or "representation" disagrees with reality. And being able to assert this at a more granular, generalizable angle. Models are not objectively worse than each other (for the most part) but they do carry different goals and assumptions and it's useful to make that clear.
- Increased use of "unsupervised methods." This paper mentions "unsupervised explanatory steps; others off-load the analysis to an unsupervised model (e.g. Unsupervised Dimensionality Component Analysis).
- The Colab Data Science Agent isn't very good. There isn't training data for this kind of work.
- There's Julius AI, Tableau, Causal (Taimur), etc.
- There's no data analysis reasoning examples that would help them express the different choices that they went down.
- What would it look like to have branching paths interface for automated reasoners and knowledge work by machines. "Choose your own adventure but for prompting."
- "Collective intelligence" in the spirit of the "Paradigms of Intelligence" team at Google. Problems are "solved" by small groups or single researchers but they're ability to do so is the product of collective human output. What would it look like to work with people who are not in our immediate vicinity or take advantage of past knowledge in a way that is just as beneficial.
  - Solving "open problems" can be reframed as a cooperative search game.

    - "Cooperative search games are collective tasks where all agents share the same goal of reaching a target in the shortest time while limiting energy expenditure and avoiding collisions."
        - "The substance to which searchers respond acts as a memory over which agents share information about the environment. **The actions of writing, erasing, and forgetting are equivalent to production, consumption, and degradation of chemoattractant.**

- You can have "super-intelligence" by tapping into "collective intelligence," just by virtue of having the power of the sum. It then becomes a question of having the various human inputs be as synergistic as possible, enabling coordination, encouraging participation, etc.

7 Data Visualization and Modelling as a test-bed

Scoping this project was important because there are many workflows to document. I wanted to choose to document the work of someone whose procedural knowledge is quite abstruse, scarce, and unexplicated, yet important and ubiquitous. Most knowledge workers are to some degree faced with the have to make sense of high-dimensional, complex data. Thus, here, I focus on procedural knowledge libraries for the data visualizer.

Data visualization is a way of simplifying the data, drawing patterns, and then visualizing them in a way that can be understood by others. Still, there are many decisions that a data visualization engineer will make and it is unclear why they make them.

For a given dataset, a visualization engineer may follow a number of paths, all of which are reasonable in their own ways. However, a consumer of a visualization often lacks the context to make sense of the decisions that were made during the development process. This leads to the interpreter taking conclusions for granted. Arguably, this is the root of lacking media literacy–an important problem in today's society.

Here, we will try to explain why procedural knowledge in this case is valuable, the kinds that are valuable to capture, and how it may be retrieved, organized, and cataloged. I call this a library of procedural knowledge for the visualization engineer, and thus use this as an opportunity to derive a functional definition of the contemporary library.

I try to describe the features of the data visualization engineer's workflow as a basis for coming up with a more extensible framework for procedural knowledge libraries that could work for a number of them. I frame the answer to these questions in terms of rough workflow sketches. Thus, this work also serves as

the basis of conversation for what the ideal library for this kind of knowledge work, recognizing that there are many details that have been omitted out of brevity and ignorance.

Data analysis is hard to "learn." There's a lot of tacit knowledge and "intuitions" that are built over time which makes transparency difficult. There's tacit judgements, exploratory detours, and aesthetic and communication trade-offs.

There's unique features of graphical model-building: **we will explore how to diff non-text artifacts**; we will learn the limits of formalizing and abstraction.

Extrapolating to anytime you want to make intermediate states explicit.

For visualization methods, there's no "absolute consistency" for what is a saliency map compared to a heatmap or neural activation. Procedural knowledge can help us intuit what are better explanations for a given model. Theory-guided data science.

Visualizations of learned representations or models. Visualization are post-hoc interpretations and we render visualization to qualitatively capture what models have learned.

Is it possible to come up with a rigorous standard of correctness?

8 Relevant

Margo Seltzer

New York Times R&D

Case-studies on Datawrapper, Tableau, etc.

8.1 Annotated Bib

- Learning to See by Looking at Noise
- A Framework for Considering Comprehensibility in Modelling
- In this All of Us failed post (From UMAP blog posts). A grife of the author is that there were no descriptions: in the Rye paper there was little justification provided for the decisions that were made such as picking 16 principal components or *What the difference would be between 20 principal components and 16*. And there is no "general analysis describing the robustness of results [of the] parameter." The author questions why the entirety of the "All of Us" consortium chose to use UMAP.
- https://arxiv.org/pdf/1802.03426v3
  - The paper mentions that "fuzzy topological representation" is a way to "merge the incompatible local views of the data."

- https://arxiv.org/pdf/2111.15506 (Towards a comprehensive visualization of structure in data
  - Data transformations as described in the paper: 1) Take non-linear manifold in lower-dim where a visualization would be largely uninformative 2) then you take linear projections of the high-dim data and make it human-readable (2 or 3-d)
  - The problem is with **non-linear methods** which are computationally complex and less deterministic (?). Examples of such methods include t-SNE and UMAP.
  - They address this with (standardized?) parametric configs? They claim this would be generalizable to other non-linear methods.
  - There's commonly trade-offs with capturing global structure compared to local structure. They propose a "retrieval information approach" where's the neighbour retriever visualizer (NeRV) that looks at the cost of precision relative to recall. They do this in terms of retrieving/missing neighbors in the high-dim representation and the low-dim representation.
  - You're also trading off speed compared to accuracy.
    * I assume the premise is that **chunking, discreteness, and cleaner parameters** translate into better queries compared to more continuous data.
- https://www.pnas.org/doi/epdf/10.1073/pnas.95.25.14863)
- https://alarmingdevelopment.org/?p=1570
- https://www.tableau.com/sites/default/files/2023-01/2008-GraphicalHistories-InfoVis.pdf Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation (2008)
- https://link.springer.com/article/10.1186/1471-2288-10-14 (Understanding human functioning using graphical models)

9 Why a library of (tacit) knowledge is valuable for the model developer

It seems odd that you can't get a snapshot of the entire state of your model as time goes on.

For typed models, you have the equational representation and the diagrammatic representation.

We want to assess the relationships between different variables and then manipulate our graphs to help us best understand what those relationships are.

With Datawrapper you can link "live" datasets and it would be useful to see how the graphs change in relaton to changes in the dataset.

This NYT project looked at changes in language use in different articles over time. There are semantic tags for the words that appear the most frequently.

10 Kinds of Procedural Knowledge to Capture

- Making sense of the dataset (i.e. whether it is tabular, etc.) Modifying the dataset based on the features that wll be abstracted (coming up with columns, etc.).
- Establishing a relation to try and synthesize a graphical design. Making sense of structural properties (with the domain sets) "and their functional relationships."
- Explaining why a data instance is anomalous. Explaining why an instance was an anomalous and defining the anomaly itself.https://dl.acm.org/doi/10.1145/3609333 (this work mentions visual models for anamoly detection)
- the assumptions
- Choosing the features to focus on for the lower-dim representation of the dataset.
- Assuming some pattern
- Trying to validate the pattern
- Iterating, debugging the visual representation

11 Benefits

- Explainability (understanding how the fnal visual came-to-be)

12 Analogies and Relevant Concepts

- Cognitive trails (temporal qualities, defining "memories" in the space of your memory)
  - There's causal links, conditional branches and attentional focus. **There's temporal sequences of thought. Showing the different *considerations* and when they arise, the different , and try to explicate many implicit processes.**
  - you reason about how "understanding of a domain manifests in an agent's behavior" (RL-related, behavioral psychology, markovian game theory, etc.)

  - "having understandable **stories** of reasoning"
  - How do you convert understanding Newton's laws of motion to calculating a projectile (Claude-generated example)
  - Tool selection (i.e. picking the right statistical theory based on understanding of probability theory (Claude-generated).
  - condition-action
- What do CRDT logs look like?
  - How do states relate over time?
  - ***Can you reconstruct a reasoning process from CRDT logs?***
    * CRDTs don't keep a full operational log. They only store the current state. Logs (for auditing, provenance, replay) then you can have operational logs manually made or through a layered system.
      · having a directed versus undirected graph
      · JSON patches
      · Track the intent and target of every change
      · you can replay them in any order (but how does work when decisions only make sense in certain/a given sequence(s))
      · here is where branching timelines could fit in
      · *ChatGPT says: "This preserves the integrity of insight while still getting the power of distributed sync."
      · **CRDTs are built to ensure convergence; but you can add-on a "procedural layer" for the sake of interpretablity.**
  - Synthesis (of visual environments)
- Coresets
  - Partitioning datasets that upon reconstruction preserve all the properties of the full dataset.
  - It's partitioned to speed up computations, making approximations faster (e.g. clustering, regression), and "compressing" procedural and data-intensive pipelines.
  - The structure is a weighted subset for a given objective function.
  - With each small subset of data you have the key properties of the transformed dataset . The coreset s "stored" with the step where it was computed (**linked list and indexing**)

  - The goal is to have a **''miniature, audit-ready trace**
  - Ordered, annotated log as a form of "narrative layer"
    * You can have human + machine annotations versus. algorithmically derived narrative layers

- Record linkage

13 Case-studies

14 Version Control

Meaningful diffs when you're working in different languages.

JS is both more imperative and un-typed. How do you retrieve the goals of different states when you're using these languages? This is compared to when I use SQL and I have clear explications of my goals and then if I migrate and there's an error then I can infer that the next steps are bug-fixing.