Technical Perspective: From Sketching to Natural Language: Expressive Visual Querying for Accelerating Insight

Bill Howe
University of Washington
billhowe@uw.edu

Visualization enables effective data exploration by harnessing the higher bandwidth interactivity of the human visual cortex. But the space of possible visualizations is immense, such that general abstractions for creating (i.e., finding) the right visualization are elusive, despite recent progressin systems like vega [2] and Draco [1].

This paper provides a general abstraction, along with advanced interfaces, focusing on visualization search. If you have ever created a long sequence of visualizations looking for interesting patterns, you have manually performed a visualization search task. The visualization search problem is to find subsets of the data that, when suitably rendered, generate a visualization similar to a provided pattern specification. This task is intuitively difficult, requiring at least a model of visualization similarity, a representation of a massive search space, a strategy for navigating the search space, and appropriate interfaces through which users can express specifications. The authors approach these challenges by designing a shape query algebra consisting of primitives and operators that can describe complex functions; i.e. "trendlines." The algebra assembles complex patterns from simple segments, akin to the way complex continuous functions are approximated by assemblies of piecewise linear functions in engineering and science (e.g., finite element models).

The key insight of the ShapeSearch algebra is that reasoning about regions of a function that are rising, falling, or flat is simple enough to capture a user's notion of pattern, but expressiv enough to model complex patterns. For example, a peak pattern rises then falls, a plateau rises and is then flat. By including expressions that can reference preceding or subsequent segments, the language can capture patterns such as "rising, then rising more sharply." In addition to sequencing via concatenation, conjunctions and disjunctions are supported; e.g., the value should rise AND not fall in a given region. Other convenience extensions include quantifiers (the value should rise at least once and at most twice), nesting (a peak within a given range), and iteration (look for a peak anywhere within a given range); these extensions can be rewritten using core primitives.

Shapesearch expressions are designed to be ambiguous — matching and scoring ambiguous expressions against dataset is another key contribution. Each segment is scored using an inverse tangent expression to produce a value between -1 and 1; these segment scores are then aggregated to produce expression scores. Fuzzy matches involve underspecified queries, where the start and/or end point of a region is left to the system to find — a potentially expensive search. A dynamic programming algorithm provides an accurate but expensive baseline, while a provably optimal approximate algorithm significantly reduces the search space by observing that start and end points will typically fall at critical points of the underlying function (i.e., where the slope changes direction), then matching and merging the query segments, greedily, according to their score.

Shapesearch supports a natural language interface for novice users, speech-to-text scenarios, and integration with other NL services, a regular expression interface for more complex expressions, and a sketching interface for drawing patterns by hand. The sketching interface promotes visual patterns as the primary model for reasoning about subsets of data — not equations, not complicated queries or code, not unreliable labels or metadata. It is this idea that separates ShapeSearch from other proposals for visualization search, including Draco [1] and Voyager [3].

Shapesearch represents a significant departure from conventional data analysis approaches by lifting up approximate visual patterns as a first class component of interaction and computation. This work has the potential to help develop a class of software and methods subscribing to this view, and, as the authors point out, immediately suggests some intriguing directions for future work: better interfaces to support mixing sketches and code, auto-complete features to support incremental composition of complex queries, and search over different attributes simultaneously.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 2008 ACM 0001-0782/08/0X00 ...$5.00.

1. REFERENCES

[1] D. Moritz, C. Wang, G. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2019.

[2] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2017.

[3] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *ACM Human Factors in Computing Systems (CHI)*, 2017.