Gestural Query Specification

Hi everyone, my name is Carmine, and I am presenting Gestural QUery Specification.

There is a strong desire for improving user friendliness with databases, and there has been a rising set of modalities and interactions such as touch manipulation, gesture tracking, gaze tracking, etc. They saw that they need to guide the users to perform queries with gestures, as well as providing feedback for their users. Needs to be closed and expressive.

Related work focused on things needed for users to perform queries. One- enumeration (showing all possible interactions with the data). Needs to be responsive (having no latency is the goal). Also, Touchviz, which is like an ipad pointer with features (like in Tableau) and using touched based gestures for exploration (its expressiveness is limited to a single table). Does not allow you to go into aggregate (thus seems weaker than Gesture Query). System that they compared Touchviz to was a windows control panel-based system. In 92, system by Schneidermman
that proposed the benefits of interaction on data for dynamic queries. Damaraju and Kerne tried to figure out if you could figure out 20 new gestures from learning 20 preset gestures (based on space offsets). Instead of forcing coders and designers to hand code and hand learn gestures, this allowed them to learn gestures from old gestures.

Challenges for gesture queries: There is not continuous feedback system (new paradigm - continuous interaction loop). If you're interacting with things in 2d or 3d, continuous interaction is an important consideration. Want to let users to interact with data themselves. Need to guide user and provide feedback, as even the limited number of interactions that the user can perform is quite large.

There are several contributions: define notion of gestural query specification. Novel querying framework, which tries to learn intent and content of a gesture/intended query. Implementation of GestureQuery, which explains their models and their design choices.

Example: Looking for all the titles of all the albums created by Black Sabbath. Difficulty: user may not know the schema of the tables, and does not know the query language. Difficulty in executing this is what they are tryign to solve with their systems. *switches to video* Question: I'm unclear about the part where they have to see all the feature and column values? Answer seen in the video. Unclear in the paper when there are too many things to show. Question: are they performing the real query on the data, as it seems it may be expensive? Answer:
yes.

Gestures: In their system, gestures are articulated body movements. IN their space, gestures are represented as a time-series of points. P_i = <t, l, m>, where t is time, l is location, and m is metadata. After the gesture, they try to get to the query itsself. Umbrella term: specification.

How to go from gesture to intended gesture. Intent - probability distribution over space of valid articulated queries. Context: set of query intent transitions, the intermediate results, and the feedback that has already been put into the system. Once the context is finished, context is put into classifier and then the classifier looks into the possible queries.

Slide showing some gesture to query mappings. This is for SQL tables. Question: is there a notion of grouping? There is, but it doesn't seem to appear in the images. Most important is the Interactive join. Scared by the possible MxN separate drag actions. Second consideration: what happens when people join tables together? Linear side-by-side is unclear, radial doesn't allow for secondary joins, but arcs make it unambiguous and readable.

Classification: Once the gesture is performed, you have your context, you get to the execution stage. They sed proximity and compativility of the tables on a mximum entropy classifier. Some features include the touches... fi(q) evaluates to from negative infinity to 0, or if it's boolean it's -inf to 0. Performed a user study, which went over multiple different cases: completion time, discoverability (how intuitive), and ... compared to VQB and Console SQL. GestureQuery outperformed in completion time, but was not as effective for usability or learnability of JOINS. GesturalQuery outperformed others in discoverability. For anticipation, JOIN suffers from not having the data (UNION does fine along all metrics). Response time was under 30ms.. it is shown to perform well across the board. Using a cache (computing compatibility features and not having to recompute for proximity) let to 4x speedup.

Question: does this support query joins? Professer Wu: I think they're equijoins. Do they actually research discoverability? Short answer: this type of measurement does not capture the nuances of discoverability and perhaps doesn't show discoverability. Despite this, it does sohw there are some things that are much harder to do in VQB than in GestureQuery. They have a compound tasks and perhaps doesnt show discoverability.

Professor Wu: THey make a lot of assumptions of what people think about JOINs and perhaps this can be researched further. Should understand why they're doing better because of how they express joins (which is unclear! is it the UI or the gestures?). Discarded MxN obvious interaction for an arc-design, should have piloted the baseline MxN interactions and compared to arc-based (these are the assumptions that can be challenged). Perhaps this would be better for JOINs. Would have to change a significant amount to the articulation for something like MongoDB and in graph-based DB's. Possible scaling issues, as testing data is likely to be super small in size.

SpeakQL

March 11, 2020

Overview

- For intermediate to advanced SQL users who want to write SQL on-the-go
    - Immediate feedback from the class questioning target demographic of proficient SQL users who need to query data in a phone/tablet format
    - Potential adverse privacy implications were also mentioned in that you speak your query out loud – begging the question, in what scenarios is this "on-the-go" speech-based querying useful for?
        - Potential use cases include DBA debugging system while on the go, nurses accessing patient records while in transit, analysts answering customer sales questions while in a meeting or on a call, etc.
- What are they existing approaches?
    - With Natural Language Interfaces (NLI), there are many intermediate steps where errors can occur so the SpeakQL argument is to go straight from user speech to SQL
    - Touch interfaces like vizdom and gesturequery are more so for beginners instead of proficient SQL users
    - Template-based interfaces allow us to use templates and then fill in variables
- SpeakQL Approach
    - First figure out the SQL template from the spoken SQL query using an automated speech recognition (ASR) engine
    - Second, match certain speech tokens to SQL template with struct termination
    - Finally, match the literals
- Structure Determination
    - Build all possible query templates offline
        - Using SQL grammar production rules
        - Build trie for queries of same length
        - Have <= 50 separate generated trees each for a certain length
    - Try to find a template that matches online
        - Match by edit distance
            - String edit distance used because fast and convenient
            - Machine learning inference would be much more expensive to perform online
        - Advanced bound and prune speed ups
- Literal Determination
    - Fill in variables to template by "sounds-like-ness"
        - First, translate all tokens and values into phonemes
        - Infer variable types from the grammar
        - Token to variable mappings
    - Hardest part is picking attribute name / values

        - Look at edit distance of each speech token against each attribute name
        - Could use CRFs or other techniques instead
- Experiment Setup
    - How accurate is it?
        - Where the tokens correctly identified?
        - Token edit distance
            - How many units of effort needed to transform SpeakQL output to correct ground truth query?
    - How easy is it to use?
        - Token edit distance again
        - Time taken
    - Empirically, dates and values are hardest for SpeakQL to get right
- User Study on SpeakQL vs. phone/tablet mobile typing
    - Statistically significant improvement for both basic and complex queries
- Discussion
    - Should people be speaking SQL directly as the best way to query data on-the-go?
        - Maybe easier to touch a graphical interface
        - Why not a mobile Tableau?
        - Mobile friendly SQL keyboard?
        - Perhaps there is an immersive situation where your hands are not free
    - Issues with SpeakQL
        - Are 50 tokens enough? / Is it expressive enough?
            - Class verdict -> probably not. In what use cases are 50 tokens enough?