Qualitative analysis

Michelle Mazurek

(some material from Vibha Sazawal, Bilge Mutlu, Susan Zickmund, Klaus Krippendorff)

QUALITATIVE ANALYSIS

Coding, in general

- Process data (e.g., transcribe)
- Break into units (a sentence, answer to one question, etc.)
  - Smaller units: More nuanced results, tougher to get intercoder agreement
- Assign a descriptive code to each unit
  - Potentially more than one

Open vs. fixed coding

- Open (inductive)
  - Create codes as you go
  - Generally reach a final set of codes "codebook" partway through (maybe minor changes after)
- Fixed (deductive)
  - Starts from THEORY / existing taxonomy
- Are codes mutually exclusive?

A good codebook

- Codes are sufficiently detailed / well defined
  - Clear when a code does/doesn't apply
- Clear link between RQs and codes
  - Don't code a bunch of irrelevant stuff
  - (revisit during grounded theory discussion)
- Granularity of coding
  - Too fine grained vs. too coarse grained
  - Capture subtle nuances
  - Don't overwhelm coder / lead to mistakes
  - Coding in levels to help with this?

Tools

- MaxQDA, Atlas.ti, Nvivo, Cassandra, QDAMiner, Dedoose
  - Others: https://en.wikipedia.org/wiki/Computer-assisted_qualitative_data_analysis_software
- Most of the good ones aren't free
- Features to consider:
  - Granularity, multiple coders, auto stats, etc.

Different approaches

- Grounded theory
- Content analysis

And relatives ….

GROUNDED THEORY

Key ideas

- Formulate THEORY from the data
- As close as possible to zero preconceived ideas
  - Open minded, multiple viewpoints, comparative
- Open coding (fine-grained)
- Followed by axial coding (to generate themes)
- Theory developed: specific -> general
- Theory must FIT

Formal/traditional

- Very rigid and rigorous
- Not usually applied in HCI context
  - Same general approach, a little more relaxed

- Straussian elements:
  - Code as you collect
  - Theoretical sampling: Choose samples with best chance of confirming/disconfirming/deepening your current working theory

Process

- Open coding
- Axial coding
- Selective coding
- Comparative analysis
- Theory building

Open coding

- Constant comparison: compare data pieces to each other
- Begin to induce concept names / codes
- As you work, continue comparing all pieces that have the same code
- Refine definition of codes as you go
- Then compare new data to existing codes
- Cycle, repeat

Building theory

- Axial: categorize codes into themes that go together
- Selective: place axial codes into "big picture" relational models. What causes/relates to what?
- Compare these big-picture findings across dimensions (e.g., people in different groups)
- These comparisons are the pieces of the theory

Theory fit

- Fitness: theory belongs to data, not forcing into pre-existing categories
  - Categories emerge from data, modify as go along
- Relevance: explain what happened, predict what will happen, interpret
- Adaptability: modify based on new data

Content analysis

- Any "text" (words, visual, etc.)
- Traditionally: conceptual analysis
  - Quantify presence of concepts in text
  - Existence vs. frequency
- Amenable to open or fixed coding
- Generally more concrete and finer grained
  - Set of codes can be iterative, but meaning per each is generally not
  - Can devolve to word counting (not great)

"TYPICAL" HCI PROCESS

Borrows from both

- Inductiveness of coding
  - When appropriate!
- Layered approach (open, axial)
- Coding concepts rather than counting words
- But less open-ended
  - Codes become fixed relatively early
  - No theoretical sampling
  - Reliability

One typical process

- Relaxed open coding to develop codebook
- Rest of coding according to codebook
  - Interrater agreement
- Axial coding process
  - Affinity diagrams? Etc.

Relaxed open coding approach

- Open coding/codebook construction
  - Typically 20% of cases? Depends on sample size
  - Can stop when approximate saturation in codes
- Next 20% to validate codebook
  - Which codes continue to be useful? Any redundant? Any to split into pieces if too general?
  - Are any new codes needed?
  - Hopefully final codebook now
- Formalize codebook
  - Definitions, inclusion/exclusion examples

Establishing validity

- Typical process:
  - One or two researchers develop codebook
  - Independently code samples
  - Measure agreement
  - Resolve disagreements to 100%
- Best: independently code all data
  - Get agreement, then resolve
- If needed: once high agreement reached, one coder does the rest.

Reliability

- Intra-rater: same coder is consistent
- Inter-rater/coder: scheme is sufficiently stable that multiple people get same answer

What do we want to measure?

- Agreement btwn independent observers
  - Not about how many observers, which ones
  - Not about how many categories/scale points
- Account for both applied and missing codes
  - Agreement on what does/doesn't apply!

Measures of agreement

- Percent agreement
  - In most cases, overstates agreement due to chance
- Cohen's Kappa
  - Accounts for chance
  - Statistical independence of coder data
  - Can be high if disagreement is **systematic**

Instead: Krippendorff's alpha

- Fundamentally measures *disagreement*
- Good: alpha > 0.8
  - Can live with > 0.667 for very fuzzy/tentative
- Works for > 2 coders
- Works for multiple variables, non-exclusive, nominal/ordinal/interval, missing data, etc.

Krippendorff's alpha

- alpha = 1 – D_o / D_e
- D_o = observed disagreement
- D_e = expected disagreement if totally chance
- Defined via coincidence matrices
- Simple example from the Kripp. Paper (2 coders, binary data, no missing data)
  - http://web.asc.upenn.edu/usr/krippendorff/mwebreliability5.pdf