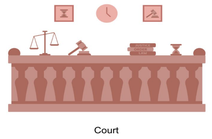
PRESENTER
Hellina Hailu Nigatu

Background

Domain experts like journalists and public defenders have to do a lot of manual data cleaning and processing when working with large document dumps. We collaborated with domain experts and built a document organization tool. We tested three programming paradigms to identify what works for our set of users.

Public Defender



Court

Methods

Data Needs

Data Cleaning
1. Duplicates.
2. Quality of pages.

Data Extraction
1. Page Type
2. Agency Format

Data Organization
1. One case; several files.
2. Multiple cases might be found in a single PDF.

Programming Paradigm Study Results

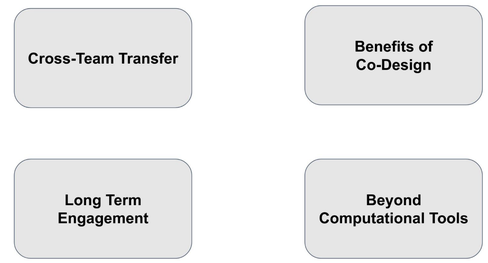
- Text-based gives users low level control and exploration outside of designer provided abstraction.
- Visual gives designers the opportunity to provide information that users could not uncover on their own.
- PBE puts the focus on the data rather than the program structure.

Through long-term co-design process, we created a communal, cross-disciplinary team, lowered barriers to communication, and created a shared vision.

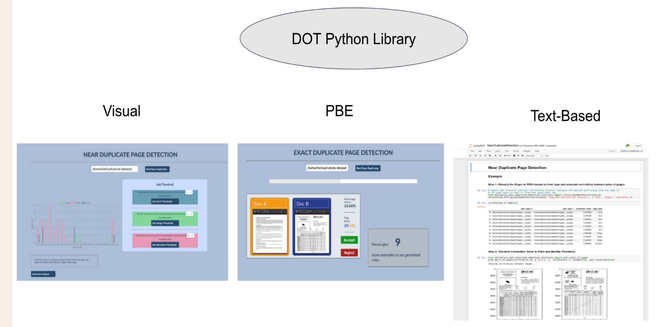
*** Work to appear in FAccT23***

Design Goal	Constraints	Design Implications
Human Control and Intervention	• Risks of mistakes in document classification are too high for this domain. • We found corrective actions to be more time and energy consuming than active, incremental decisions.	Design should prioritize supporting users exclusively in organizing the data rather than automating the whole process.
Non-Interference with Existing Practices	• Cross-team differences in data management and handling practices. • Conflicting needs: Privacy and security concerns with using online platforms for one team conflicting with lack of local storage space for large data size in another team. • Pre-existing workflows for post-data organization tasks and file sharing.	Design should account for and be adoptable to pre-existing workflows and practices.
Robustness to Data Variants	• Different teams with similar but not identical data. • Changes in data structure due to differences between LEAs themselves.	Potential solution would need to be resilient to changing formats and representations and inter-operable with similar but not identical datasets from others.
High-level Abstractions	• Plain programming languages like Python or R require too much detailed technical knowledge to execute the required tasks. • Pre-built software solutions give limited flexibility to our users.	Tools should prioritize meeting users where they are with technical skills; requiring minimum training while allowing flexibility.
Cost-Sensitive Solutions	• Resource-constrained teams lack the monetary resource to employ commercial software for their tasks. • Open-source software products do not produce same level of quality results.	When relying on open source software, tools should identify trade-offs with quality and ensure quality control with other schemes.

Lessons Learned: Co-designing



Programming Paradigms



EPIC DATA lab
UC Berkeley

Hellina Hailu Nigatu, Lisa Pickoff-White, John Canny, Sarah Chasins