Co-Designing for Transparency: Lessons from Building a Document Organization Tool in the Criminal Justice Domain

Hellina Hailu Nigatu
hellina_nigatu@berkeley.edu
UC Berkeley
USA

Lisa Pickoff-White
lpickoffwhite@kqed.org
KQED
USA

John Canny
canny@berkeley.edu
UC Berkeley
USA

Sarah E. Chasins
schasins@cs.berkeley.edu
UC Berkeley
USA

ABSTRACT

Investigative journalists and public defenders conduct the essential work of examining, reporting, and arguing critical cases around police use-of-force and misconduct. In an ideal world, they would have access to well-organized records they can easily navigate and search. In reality, records can come as large, disorganized data dumps, increasing the burden on the already resource-constrained teams. In a cross-disciplinary research team of stakeholders and computer scientists, we worked closely with public defenders and investigative journalists in the United States to co-design an AI-augmented tool that addresses challenges in working with such data dumps. Our Document Organization Tool (DOT) is a Python library that has data cleaning, extraction, and organization features. Our collaborative design process gave us insights into the needs of under-resourced teams who work with large data dumps, such as how some domain experts became self-taught programmers to automate their tasks. To understand what type of programming paradigm could support our target users, we conducted a user study (n=18) comparing visual, programming-by-example, and traditional text-based programming tools. From our user study, we found that once users passed the initial learning stage, they could comfortably use all three paradigms. Our work offers insights for designers working with under-resourced teams who want to consolidate cutting-edge algorithms and AI techniques into unified, expressive tools. We argue user-centered tool design can contribute to the broader fight for accountability and transparency by supporting existing practitioners in their work in domains like criminal justice.

CCS CONCEPTS

• **Human-centered computing** → **Interaction paradigms**; **User studies**; *User interface design*; *User centered design*; *Interface design prototyping*; • **Computing methodologies** → *Artificial intelligence*; *Machine learning*.

FAccT '23, June 12–15, 2023, Chicago, IL, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0192-4/23/06.
https://doi.org/10.1145/3593013.3594093

KEYWORDS

Co-Design, Document Organization, User-Centered Design, Collaborative Design

ACM Reference Format:
Hellina Hailu Nigatu, Lisa Pickoff-White, John Canny, and Sarah E. Chasins. 2023. Co-Designing for Transparency: Lessons from Building a Document Organization Tool in the Criminal Justice Domain. In *2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23), June 12–15, 2023, Chicago, IL, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3593013.3594093

1 INTRODUCTION

Access—or lack thereof—to Law Enforcement Agency [1] (LEA) procedures and records in the United States shapes the interaction between the public and the agencies in several ways. Firstly, the lack of transparency influences the public's view and trust of LEAs [24, 46]. Secondly, disclosure of data around LEAs allows investigative journalists to hold institutions accountable and public defenders to increase fairness in the criminal justice system [23, 42]. Historically, there have been several efforts to increase transparency through legislation [53], journalism work [23], and community advocacy [43]. While such efforts have led to data disclosures around police misconduct and *use-of-force*[36], the process of disclosing these data is not so straightforward. In addition to ongoing efforts to block data disclosures [32], when requests are successful, data are released in large and messy formats [56], requiring large amounts of cleaning and organizing before they can be useful for the public's understanding and trust of LEAs.

Entities like public defenders and investigative journalists play the middle organizational role between the public and LEAs in terms of data disclosures. Public defenders carry out the essential job of holding LEAs accountable in a court of law by using the disclosed data to support and form their cases [56]. Investigative journalists focusing on data or record-based police accountability projects [35, 42] conduct fieldwork and research to force agencies to release the records necessary for producing articles, and help the public understand the disclosed data. Usually, such entities are constrained in time, money, and resources [2, 56], limiting the efforts they can put into automation, while also constraining the person-hours they can afford to spend on data cleaning and organization.

These teams, who have fought to access essential data for their work, now find themselves in a "transparency paradox," in possession of large amounts of information but unable to distill and utilize it [8]. While previous work at FAccT [5, 39] has argued for structuring disclosed data to ensure those receiving it can easily use it; current data management practices rarely produce easy-to-use data, particularly in the case of LEAs. Data releases are sometimes intentionally obfuscated, as in the case of "data overloads" [13]. In this paper, we focus on resource-constrained teams who process and utilize this kind of hard-to-use data releases.

We conducted 14 months of cross-disciplinary collaboration between (i) stakeholders who work on processing large "data dumps" of police use-of-force and misconduct documents (journalists, public defenders, and workers at public defender support organizations) and (ii) computer scientists. Over the course of the collaboration, we built Document Organization Tool (DOT), an open-source Python library with data cleaning, extracting, and organizing features. Previous work [19, 22, 30, 58] has shown the importance of active engagements between designers and their users. In line with this insight, our design process emphasized frequent and open discussions within our cross-disciplinary team alongside direct engagement with the datasets our users process manually.

Through our engagements, we found a number of relevant observations for tool builders, such as how some of our users became self-taught programmers to automate some of their data organization tasks. Based on our user-centered co-design values, we wanted to support users in how they were already thinking about the problem. Hence, we built a Python library and two programming tools on top of our Python library, presenting DOT in three programming paradigms—Visual, Programming-By-Example, and standard Text-Based—to identify which programming environment made the most sense to our self-taught, domain-expert practitioners.

Throughout this paper, we illustrate how user-centered co-design can allow researchers to support domain experts in the broader fight for transparency and accountability. Our contributions include:

- **Lessons learned from a co-design process, building a document organization tool with domain experts**: We discuss our co-design approach and detail the resultant design goals in Section 4.2; Section 7.1 details the lessons we learned.
- **Implementation of DOT, a library for processing and organizing disclosures from Law Enforcement**: We outline the features of DOT and the design decisions that went into its implementation in Section 5.
- **A user study exploring the programming needs of resource-constrained domain experts**: We present the study procedure and results in Section 6 and discuss the implications in Section 7.2.

In this work, we show how co-designing with stakeholders through an engaged process allows designers to augment domain experts' skills, particularly in high-stakes situations like the legal domain. Our work contributes to the fairness and transparency cause by (1) presenting our reflections from co-designing computational tools with domain experts, and (2) presenting an open source tool for domain experts working with large amounts of disclosed data. We hope that other designers working with such practitioners will learn from our reflections and transfer the lessons to other domains where large data disclosures are abundant.

2 RELATED WORK AND BACKGROUND

2.1 User-Centered Design of Data Tools

Outside of the computer science community, domain experts have picked up computer programming for their computational needs [27] across many domains [26, 28, 34, 45]. Prior work contributes tools for domain experts in non-technical fields who program with data, addressing data needs such as data collection [4, 57], data labeling [62], and data visualization and analysis [7, 21]. Domain experts' needs differ greatly from professional software developers'—thus designing for these audiences is substantially different [27]. In the case of public defenders and investigative journalists, they care not just about efficiency and speed but about the broader social and political goals that motivate their work. Hence, our design process must center community goals, shaping our design constraints (Section 4.2).

In the case of "data disclosures," previous literature in FAccT argues for the re-conceptualization of information disclosures as "'interfaces'—designed for the needs, expectations and requirements of the recipients they serve to inform" [39]. However, in some settings, e.g., LEA disclosures, asking for usable formats may not always produce releases in usable formats, both because LEAs may wish to obfuscate data and because their own data practices may make it difficult to release well-structured records. Previous work at FAccT [41] argues a primary characteristic of 'disclosure datasets' is how they are produced and reported by the same institutions they are meant to hold accountable. This puts both the power and the responsibility in the hands of the disclosing agencies. Building on [39], we lay out an alternative, complementary approach that revolves around changing data recipient practices rather than data releaser practices. Looking at disclosures from the perspective of receiving agencies rather than disclosing agencies, we build on previous HCI works to support end-user programmers working with large, disclosed data. Our work offers a demonstration that user-centered design techniques can help researchers tackle high-leverage needs in Fairness, Accountability, and Transparency. Although our work primarily centers transparency, we anticipate user-centered tool design could enhance work in a variety of spaces—for example, community-driven dataset and algorithmic audits.

2.2 Large Data Dumps and Public Records

Dealing with messy data takes significant, valuable time from resource-constrained teams such as public defenders and investigative journalists. As previous work shows, public defenders engage in a "scavenger hunt" type of data searching and believe that their lack of time, technical resources, and technical skills negatively affect their ability to do their jobs [2, 56]. Through our co-design process, we found that investigative journalists also have similar data needs. In this paper, we offer one way of addressing this systemic issue by supporting the technical skills they already possess through user-centered tool design.

2.2.1 Nature of Data Dumps. Modern "data dumps" of public records present profound challenges for organization, analysis, and dissemination. Data includes a variety of PDF files, generated from many diverse templates and formats. Data can also include several levels of duplication—standard pages may be shared across multiple files, and partial or full PDF files may be copies of each other. Data can also differ based on scanning quality and redaction levels. The same physical paper may have been scanned multiple times, resulting in different images. It may appear in multiple different PDFs, even redacted in different ways. Different pages within a single PDF may include forms, narratives, interview transcripts, images and drawings, or handwritten notes. Additionally, individual cases may be spread across several files in different layers of folders, or several cases might all be consolidated in one large PDF. Adding to the complexity, records related to the same case may come from more than one agency; for instance, a police department, district attorney, medical examiner, and the state Department of Justice could all have related files, all in separate documents.

3 CO-DESIGN METHODOLOGY

Our work centers on the collaboration of a cross-disciplinary team of domain experts and tool-building experts that took place over the course of 14 months. The two central members of our team were: the first author, a computer scientist at the University of California, Berkeley; and the second author, a journalist at KQED, a public radio station in San Francisco. The team included: (i) additional computer scientists from UC Berkeley, (ii) additional journalists from KQED, (iii) public defenders from across the United Sates, and (iv) programmers and engineers from the National Association of Criminal Defense Lawyers (NACDL) Full Disclosure Project, an organization aimed at supporting defense attorneys across the United States. All team members from groups (ii)–(iv) regularly worked with police misconduct data at the time of the co-design work.

Research in HCI [20, 48, 49] points to the benefits of collaboration when designing for domain experts. Designers have used methodologies and frameworks such as *action research* [29] and *participatory design* [33] to provide solutions that solve real-world problems [19, 47]. Domain experts' insights in such processes help ground the solutions in practical ways; these frameworks also highlight how subjective decisions by designers are desirable and necessary to the process as opposed to threatening validity [30]. A key emphasis in collaborative design processes is that such work should center *"transferability not reproducibility"* [47]. Although we initially started working with investigative journalists, our design process allowed us to transfer our tool capabilities to public defenders with similar—but not identical—data needs. We discuss transferability in more depth in Section 7.1.

Our design process followed what previous research in visualization has termed *design by immersion* [17]: a methodology by which experts from one domain engage with and participate in the work of another domain. In particular, we practiced *apprenticeship*, whereby the first author (*designer*) spent considerable time building first-hand experience in processing the domain experts' large datasets. The initial stages of our design process emphasized understanding how users manually conducted data cleaning and organization. The first author spent time directly analyzing the datasets these practitioners were working with and replicating the manual organization processes herself. Getting hands-on experience with the data gave us an intuition that helped structure our collaborative need-finding and brainstorming phases. Our team shared early prototypes in the form of computational notebooks and iterated on features and methods we employed in our design. The second author (*domain expert*) served as the navigator for the vast data dumps and focused designer attention on edge-cases encountered by their team during prototype usage. The entire design process of DOT took 14 months of collaboration.

Averaged over the course of our collaboration, we met across disciplinary boundaries more than once per week, with meeting frequency varying according to need at various stages of the work. Within-discipline meetings were also more than weekly. In addition to meetings, we also communicated regularly both within and across disciplinary boundaries over a shared Slack space; the Slack space was limited to team members from KQED, the NACDL, and UC Berkeley. Finally, we communicated regularly both within and across disciplinary boundaries over email.

4 FINDINGS FROM CO-DESIGN

Over the course of our co-design process, we identified three main data needs and five design goals we outline below.

4.1 Data Needs

In this subsection, we taxonomize the data needs we identified through our collaborations and direct engagement with the data.

4.1.1 Data Cleaning. We identified the central role of data cleaning, especially de-duplication: identifying copies of the same data spread across multiple parts of a data dump. We found two types of duplication: (1) *exact duplicates* in which images of pages are pixel-for-pixel copies of each other and (2) *near duplicates* in which two images are not pixel-for-pixel identical, but both are images of the same physical (paper) document in the real world. Exact duplicates happen in cases where (i) there are standard pages shared across several files (e.g., header pages), (ii) PDF files are entirely or partially included within larger PDF files, or (iii) exact copies of the same file are shared repeatedly by the same party or from different parties and appear across the document dump. Near duplicates occur due to (i) multiple scans of the same document, which may vary in orientation, scan quality, scan size, or physical damage to the document (e.g., stains) and (ii) differences in redaction applied to a single scan. Both types of duplication are a problem because they (1) waste human time either by causing them to re-process the same data or requiring them to go through the painstaking process of manually identifying duplicates to avoid re-processing and (2) take up storage space. Near duplicates come with the additional issue that redacted copies may miss data contained in other copies.

4.1.2 Data Extraction. Practitioners also struggled to extract relevant information such as names, dates, locations, and case numbers from case files. Due to the different formats used by different agencies as well as the different layouts of individual pages, extracting data from such data dumps requires time and manual effort.

4.1.3 Data Organization. Finally, practitioners needed to organize PDF files into individual cases. This requires both splitting and grouping files, as (i) a single case might be spread across several files in several folders, and (ii) multiple cases might appear in a single, large PDF. In the first case, users have to look through many files to collect the ones that belong together. In the second case, users have to look through a PDF, identify the boundaries between cases, and split it.

4.2 Design Goals

Via a combination of observations of stakeholder practice, direct conversations with stakeholders, and conversations across the cross-disciplinary team, we identified five key design goals: *human control and intervention*, *non-interference with existing practices*, *robustness to data variants*, *high-level abstractions*, and *cost-sensitive solutions*. The collaborative nature of our team allowed us to identify the constraints of our users' needs and practices and iteratively adapt our design while refining our design goals. For instance, initially, our design for the data organization (Section 5.3) task utilized clustering algorithms, following trends in previous works [3, 31, 37, 50, 54]. However, we identified that our design had to allow for *human control and intervention* (see Table 1). This is in line with previous research on human-machine collaboration and hybrid decision-making [5, 6, 9, 12, 39, 55], indicating design processes should prioritize supporting, not replacing, users. Our design also had to be sensitive to trade-offs that impact quality of output. For instance, keeping with the design goal for *cost-sensitive solutions*, we had to use a free, open-source OCR engine, which necessitated post-processing for data extraction to make up for the lower-quality results. Table 1 presents a summary of the core design goals that we constructed via our cross-disciplinary co-design process.

5 IMPLEMENTATION OF DOT

This section covers the technical details of the implementation of DOT. DOT is an open-source Python library available at: https://github.com/hhnigatu/DOT. First, we will discuss DOT's features for data cleaning: identifying exact and near duplicates in Section 5.1. Then we will present the features for data extraction in Section 5.2. Finally, we will describe DOT's data organization features in Section 5.3. We present details of available functions in Appendix C.

5.1 Data Cleaning

Exact Duplicate Detection (EDD). We accomplish EDD by running each page of the PDF through an *MD5 hashing algorithm* [44] and collecting pages with the same hash value as duplicates. To avoid reprocessing the same pages, we select a representative page from a group of duplicates by choosing the first page in the list of duplicates. At the final stage, all the data extracted from this page is copied over to the other pages in the group of duplicates.

Near Duplicate Detection (NDD). We used vector embedding clustering and image correlation for NDD. Through our iterative design process, we found that calculating image correlation and setting a threshold allowed us to capture the *near duplicate* pages in the datasets. However, this approach had two issues: (1) calculating pairwise correlation for large datasets takes too much time, and (2) threshold value for correlation depended on the page layout. We circumvented these problems by fine-tuning LayOutLMv2 [59] on a subset of our data for sequence classification and using its visual backbone to produce the vector embeddings for the pages. We detail our fine-tuning efforts in Appendix A. We then used FAISS [1] k-means clustering to group similar pages and calculated pairwise correlation values on subgroups of similar pages, reducing processing time and resource consumption. NDD also catches pages that picture the same physical document but have different hash values (e.g, due to a slightly skewed scan). With these optimizations in place, our clustering and correlation approach addressed users' NDD needs.

5.2 Data Extraction

Page Type Classification. We used our fine-tuned LayOutLMv2 [59] model to classify each page in the data as a form, narrative, image, handwritten note, or interview transcript. This step allowed us to strategize how we do data extraction depending on the type of the page. For instance, we observed that case numbers are usually found in a specific place on forms and that Named Entity Recognition (NER) did not perform well for text extracted from forms.

Named Entity Recognition. We used BERT [10] to perform NER. For narratives, NER was the best tool for extracting information relevant to document grouping. In particular, we extracted names, dates, and locations.

Regular Expressions. For case numbers, we found that fine-tuning NER would take too much time and too many resources. We found that case numbers usually had a pattern that can be matched with regular expressions. We added the option to provide a bounding box for our users to narrow down the range of text on which our tool performs regular expression matching.

5.3 Data Organization

Splitting Large Files. We found that our users relied on, among other things, patterns in the page type to decide where to split a large PDF. For instance, if a large PDF had a pattern of forms followed by narratives followed by interviews, our users would break up the PDF at the first instance of a form while inspecting the content. DOT gives users the option to use this type of domain knowledge to split large PDFs into smaller chunks by specifying a page type.

Grouping Files Associated with the Same Case. To decide which files belong together, we observed that our domain experts rely, in order, on: (i) case number matches, then (ii) the combination of a name and incident date match. Following the steps described above, DOT produces names, dates, and case numbers detected from each file in a tabular format and highlights rows where those entities match, suggesting to users which files likely relate to the same case. The user has full control over whether to accept or reject the suggestions.

[1]https://github.com/facebookresearch/faiss

Table 1: Design goals identified through our iterative co-design process. In this table, we outline the constraints we identified and the design implications of the constraints.

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.

6 FORMATIVE STUDY OF PROGRAMMING PARADIGMS

Via the co-design process, we found that some domain experts had become self-taught programmers, and we wanted to leverage those skills. However, conversations within the cross-disciplinary team were not sufficient to teach us what programming paradigm would best equip our users to accomplish their tasks given their level of technical expertise. As a final stage in our co-design process, we conducted a formative user study focused on the data cleaning features of our tool and built interfaces in three programming paradigms. We aimed to use this formative study to complement the lessons drawn from co-design and shape the final choices in our design process. With the appropriate abstractions already implemented, how should we present the abstractions to the target audience?

6.1 Programming Paradigms

We presented our tool in: (1) visual, (2) programming-by-example (PBE) and (3) traditional text-based paradigms.

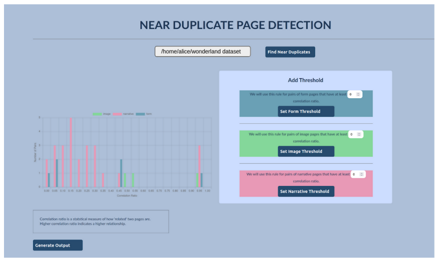
6.1.1 Visual. From block-based programming environments [11, 40] to click- and touch-based interfaces [38, 52], visual programming offers an alternative way to construct programs, aside from traditional text. Our visual interfaces were designed to provide users with an overview of their data through histograms and provide a box representation of each DOT function call in the target program. For interface details, see Figure 1a.

6.1.2 Programming-By-Example. In our PBE interface (Figure 1b), we provide users with side-by-side displays of possible duplicates (files in EDD and pages in NDD). Users either accept or reject each pair, giving us a dataset of duplicates and non-duplicates. Behind the scenes, a simple custom program synthesis ([16]) algorithm uses the labeled examples to generate parameter settings. Users may manually alter the generated parameters if desired.

6.1.3 Text-Based. In the text-based paradigm (Figure 1c), we present our Python domain-specific library through a Jupyter notebook.

6.2 Study Procedures

We conducted a within-subjects, counterbalanced study, observing each participant using each of the three paradigms described above. We assigned participants to either EDD or NDD condition. For each session, after giving consent, participants filled a pre-interview survey which included questions about programming experience



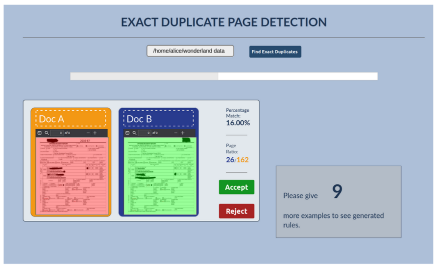
We will use this rule for pairs of form pages that have at least 0 correlation ratio.

We will use this rule for pairs of image pages that have at least 0 correlation ratio.

We will use this rule for pairs of narrative pages that have at least 0 correlation ratio.

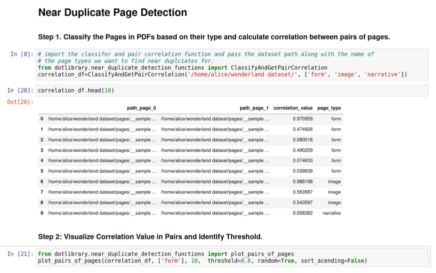
Correlation ratio is a statstical measure of how 'related' two pages are. Higher correlation ratio indicates a higher relationship.

(a) Visual interface for NDD.



Please give 9 more examples to see generated rules.

(b) PBE Interface for EDD.



Near Duplicate Page Detection

Step 1. Classify the Pages in PDFs based on their type and calculate correlation between pairs of pages.

```
In [8]: # import the classifer and pair correlation function and pass the dataset path along with the name of
        # the page types we want to find near duplciates for.
        from dotlibrary.near_duplicate_detection_functions import ClassifyAndGetPairCorrelation
        correlation_df=ClassifyAndGetPairCorrelation('/home/alice/wonderland dataset/', ['form', 'image', 'narrative'])

In [20]: correlation_df.head(10)

Out[20]:
                                  path_page_0                                  path_page_1  correlation_value  page_type
0  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.970959       form
1  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.474926       form
2  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.080018       form
3  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.490259       form
4  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.074633       form
5  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.039939       form
6  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.988198      image
7  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.553587      image
8  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.543597      image
9  /home/alice/wonderland dataset/pages/__sample ...  /home/alice/wonderland dataset/pages/__sample ...           0.008382  narrative
```

Step 2: Visualize Correlation Value in Pairs and Identify Threshold.

```
In [21]: from dotlibrary.near_duplicate_detection_functions import plot_pairs_of_pages
         plot_pairs_of_pages(correlation_df, ['form'], 10,  threshold=0.6, random=True, sort_acending=False)
```

(c) Text-Based Interface for NDD.

Figure 1: (a) Visual Near Duplicate Detection. A histogram shows the distribution of correlation values for pairs of pages. On the right, users can set correlation thresholds for form, image, or narrative page types. (b) PBE Exact Duplicate Detection. For a pair of documents, users can indicate if the pair are duplicates or not. Pages that appear in both documents are colored green. The interface shows the number of pages that matched across the two documents. (c) Text-Based Near Duplicate Detection. A Jupyter Notebook; users write code that uses the DOT library functions.

and exposure to large document dumps. Then, participants watched a tutorial for the first interface they would use. Once they finished the tutorial, we gave participants remote access to the interface and provided the set of tasks. When participants were done with their tasks, we asked them to fill out the NASA Task Load Index (TLX) [18], which measures perceived cognitive load and is a widely used usability scale in human factors research. Once a participant completed the tasks on all three interfaces, we conducted a semi-structured interview to understand their experience. Finally, we debriefed participants about the process.

6.2.1 *Tasks.*

We designed study tasks inspired by real scenarios we observed through our interactions. The scenarios and tasks used in the experiment are in Appendix B. We had two types of tasks for both exact and near duplicate detection:

- **Exploration Task:** We asked participants to explore the interface and write a program they believe would identify duplicates. For EDD, this task came last, meaning participants already had some notion of what values yielded duplicate documents, while for NDD this task came first. We switched the sequence of the exploration task because we wanted to see how participants' behavior might change when they had no prior exposure to the types of parameters they should set.
- **Fixed Task:** In this task, we asked participants to write particular programs the researcher provided which would get different quality outputs of duplicates in the data provided. For both exact and near duplicates, we included two fixed tasks, with varying levels of difficulty.

For EDD, we used reports of police use-of-force and misconduct from a county within the United States. For NDD we used citizen complaints about police use-of-force from one US state public defender's office. Table 2 gives details of data size.

6.2.2 *Participants.*

We received 30 responses to our screening survey which we distributed through professional networks to journalists and public defenders. We conducted 18 sessions in total, selecting users with prior experience in working with large data dumps, with 12 sessions for EDD and 6 sessions for NDD. Participants varied across several axes, including prior work on identifying exact or near duplicates, programming skill, and exposure to particular programming languages. We refer readers to Appendix E for a detailed description of the participants. Throughout the paper, we will refer to participants using the exact duplicate interfaces as PE (e.g., PE0, PE1, . . . , PE11) and participants using the near duplicate interfaces as PN (e.g., PN0, PN1, . . . , PN6).

Table 2: Number of files and total number of pages in the data used for Exact Duplicate Detection (EDD) and Near Duplicate Detection (NDD) in study sessions.

	Number of Documents	Number of Pages
EDD	1515	62,311
NDD	862	12,400

6.2.3 *Consent and Compensation.*

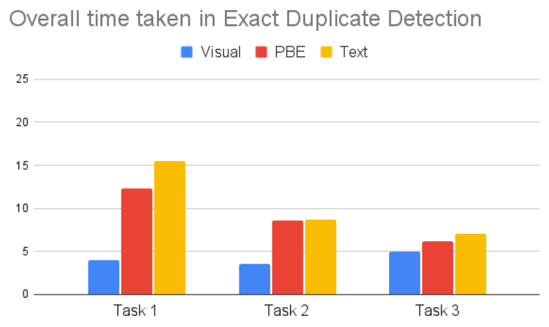
Before participating in the study, each participant signed a consent form in accordance with the UC Berkeley Institutional Review Board. Each participant was compensated $20 per hour of participation, with some participants donating their compensation to a 501(c)(3) organization of their choice.

6.3 Results

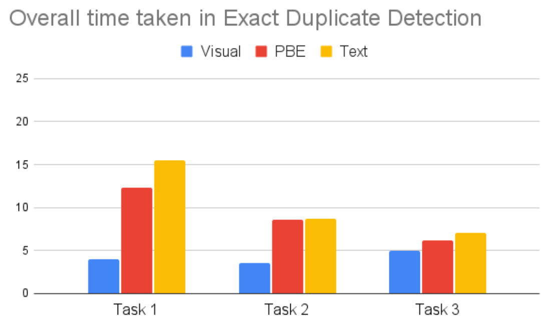
6.3.1 *Participant Performance.*

Overall, during the course of a session, participants became competent with all three programming paradigms, successfully implementing programs using visual, PBE, and text-based interfaces.

Time. To understand variations in task completion time, we conducted statistical significance tests using a one-way ANOVA test with a significance level of $\alpha = 0.05$. Variation in task completion times was statistically significant for: the second (p=0.02) task in EDD and the first (p=0.04) and second (p=0.03) tasks in NDD. Variation in task completion times was *not* statistically significant for: the first (p=0.06) and third (final) task (p=0.0.97) in EDD and the third (final) task (p=0.70) in NDD. See Figure 2 for the time per task;



(a) Average completion time for Exact Duplicate Detection tasks.



(b) Average completion time for Near Duplicate Detection tasks.

Figure 2: Task completion times for participants using Visual, PBE, and Text-Based paradigms. Overall, participants complete tasks quickly with all paradigms by their third task.

see Appendix D for time data broken down according to the order in which participants used each interface.

We observed the highest variation during the first task, which included the time participants took to learn how to use the programming paradigm under test. Although participants were significantly slower to complete the first task using traditional text-based programming (p=0.012), times were approximately even across all paradigms by the third task (Figure 2).

Adherence to researcher-assigned task specifications. Participants' adherence to the task specifications varied across paradigms. In the case of EDD, all participants reached the researcher-assigned goal for the simple fixed task (setting a particular percentage threshold) using the visual and text-based paradigms, while only 50% of the participants set the exact researcher-assigned threshold using PBE. In the second and more difficult task of (i) creating a rule for only a particular range of document sizes and (ii) using a particular number of matched pages as the threshold, participant adherence to the task dropped for all paradigms. Only one participant wrote the assigned program using PBE. In the Visual paradigm, 83.33% of participants used the researcher-provided thresholds, and in the Text-Based paradigm, 66.67%.

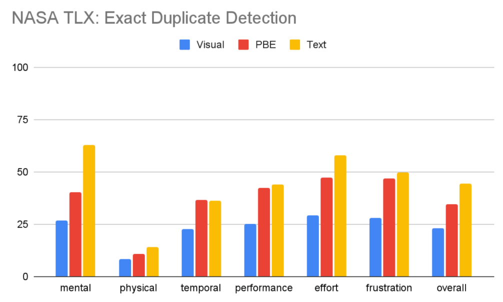
For NDD, for the simple fixed task of setting thresholds for both document types, 100% of participants used the researcher-assigned thresholds with the visual paradigm, followed by 83.33% with PBE and 66.67% with text. Interestingly, for the difficult fixed task (setting a fixed threshold for *only* image pages), participant adherence was highest with PBE at 83.33%, while both visual and text-based paradigms produced adherence rates of 66.67%.

Workload and confidence. Despite participants becoming competent with text-based programming by the ends of their sessions, participants' reported workload was highest for the text-based approach. Figure 3 shows participants reported the highest mental workload for the text-based paradigm. For EDD, the text-based and PBE paradigms elicited approximately even levels of reported frustration, higher than the visual paradigm, but there was no statistically significant difference between the three paradigms (p=0.18). For NDD, text-based elicited more frustration, while PBE was reported least frustrating; this difference was significant (p=0.02). Our results also indicate that participants were least confident about their performance when they used the text-based paradigm (see "performance" in Figure 3); differences in reported confidence in their performance were not statistically significant for EDD (p=0.19) but were significant for NDD (p=0.01). For EDD, the visual paradigm achieved the lowest (best) overall workload score (p=0.02); for NDD, participants gave PBE the lowest overall workload score (p=0.01).

6.3.2 *Perception of Programming Paradigms: Perceived ease and flexibility and the dangers of flawed mental models.*

Participants expressed that they found the visual paradigm tools to be "straightforward" and "easy to use." Visual cues helped them make predictions about program output. For instance, we used shading ranges of a histogram to indicate how much of the input data a given *rule*—a program indicating threshold values the user wants to set—would cover. When participants designed rules to apply to all documents and the shaded region covered the whole histogram, participants (PE0, PE1, PE3, PE5) reasoned that their rule was covering everything in the data, and so they would have many outputs. We observed similar prediction practices for NDD when participants set low thresholds and used the histogram to predict how many documents would fall above those low thresholds. Overall, the visual programming interfaces surfaced information that participants reported using to make predictions about their program behaviors.

PBE interfaces prompted participants to speculate about how the underlying synthesis algorithms worked. Some participants correctly assumed that they were seeing rules based on the accumulation of the examples they accepted. One participant (PE2) also mentioned that the rules were useful for understanding what types of examples they were accepting: "The rules also helped me understand what I was doing. It said 100% and 100%, so maybe I was only clicking on things that were a high match." The most common reported misconception in both EDD and NDD was the assumption that the synthesizer would generate thresholds *at or lower than* the minimum score from the participant-accepted pairs (even if the participant had also rejected pairs with higher scores). In fact, the synthesis algorithm used logistic regression on the participant-labeled data points, and thus could choose thresholds *above* the minima. This misinterpretation of the synthesizer's actions led to



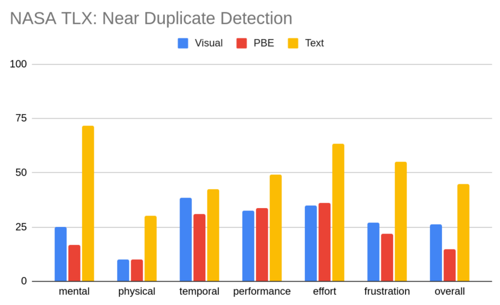


Figure 3: Unweighted NASA TLX scores. For all workload sub-components (e.g., "mental," "physical"), lower is better.

frustrations from participants, in line with previous research on users' mental models of program synthesizers [25]. For instance, when PE0 wanted a rule that matched documents if at least 50% of their pages matched, then accepted an example pair in which the paired documents exhibited a 50% match, they were disappointed that the synthesized rule used 81% as its threshold: "So … why didn't my rule change the condition? Nothing changed in terms of the rules. I thought once the rules appeared, I thought … I thought you take into account choices …"

Participants (PE0, PE1, PE5, PE6, PE7, PE9, PE10, PN0, PN1, PN2, PN3) reported they felt the text-based paradigm gave them more control over what the tool was doing. Interestingly, two participants (PN0 and PE6) used the text-based interface to do something that was not supported in the visual or PBE tools. PN2 also said they felt the text-based interface "makes [them] understand what the program is gonna do. It is easy to tell when you edit something if it is doing what you want it to do."

6.3.3 Programming Errors. Errors in text-based interfaces were mainly caused by copy and paste issues, misspellings, or misunderstanding function return values. When attempting to visualize outputs, most participants (PE2, PE4, PE5, PE9, PE10, PE11, PN0, PN4, PN5) encountered syntax errors when they started variable names with numbers. This issue was common as participants tried to use the name of the file they wanted to visualize as a variable name, and all file names started with the year of the case they were describing. Some participants (PE2, PN0) tried putting the file names in quotes, resulting in another syntax error. PN0 was able to recover by adding "Doc_" before the name of the variable while PE2 gave up. Another common error was from copying and pasting partial code. Participants recovered from this type of error by recopying the cell. We observed that error messages were discouraging for participants using the text-based interface. Participants used phrases like "please, don't yell at me" (PE7) to address the programming interface or "oh, no … it [the programming interface] is mad at me" (PE8) when seeing error messages. In the open-ended interview, PE2 said:

> I just didn't understand because it's not like it says, "you have an extra space..." It just shows you so you have to understand what the code is, you have to know the language before you can … be told what we're doing wrong.

For PBE interfaces, programming errors were usually related to a lack of control over which documents they labeled and the consequent difficulties in getting the desired types of rules. Participants (PE3, PN0, PN5) recovered by restarting the application or by clicking the "Clear Examples" button, which erases the examples the participant had given thus far and allows them to start from scratch. In the visual interfaces, participants made errors such as using a rule specified for *form* page types to set the threshold for *image* page types but used the color cues to recover quickly.

6.3.4 Trends in Programming Practices. Participants tended to tweak existing code rather than writing program from scratch. With text-based programming, all but one of our participants copied from the example notebooks instead of writing code from scratch, which is in line with previous research on *blank-page syndrome* [14, 15]. Interestingly, people with no traditional programming experience were able to accomplish the tasks using text-based programming. By relying on the tutorial videos and example notebooks, four participants who reported no prior exposure to Python or text-based programming were still able to accomplish the given tasks.

6.4 Reflection and Influence on DOT

The formative user study allowed us to observe the programming practices and common programming trends of the audience we aim to serve. Combined with the insights we developed from our co-deign process (Section 7.1), the results of our formative study allowed us to answer the question: Once we design a tool with features informed by collaboration with stakeholders, how can we effectively present the tool to our users? Our formative study led us to believe that it is not the programming paradigm, but rather the abstractions in which we present the features of the tool that mattered. We took the lessons from the relative strengths of each paradigm and combined the histograms from our visual paradigm with the expressiveness of our text-based paradigm to present DOT as a text-based Python library with visualization support. Based on the trends we observed in programming errors, we added features in our tool for error handling by, for instance, describing to users that there are no files in the paths they provided instead of throwing syntax errors.

7 DISCUSSION

As discussed in Section 2.2, computational tools can play an instrumental role in ensuring large disclosed data from LEAs serves its purpose: transparency. Through a long-term, collaborative design process involving stakeholders and computer scientists, our work centers the benefits of co-designing tools for non-computing domains. This section reflects on lessons learned from our co-design process, highlights key takeaways from our user study of programming paradigms, and states the broader implication of building tools for low-resource teams in high-stakes situations.

7.1 Reflections on Co-Designing with Stakeholders

Cross-Team Transfer. In the initial stages of the design, we only interacted with investigative journalists working on police use-of-force and misconduct data. When we later found public defenders working on complaints against police use-of-force, the two audiences of domain experts believed they had different data needs. The first author's (*designer*) outsider perspective combined with the hands-on experience she gained by processing data from both teams allowed us to distill design goals that could accommodate both teams, bridging the perceived difference in data needs.

On Long-Term Engagement. Our design process took 14 months to complete, with frequent regularly scheduled engagements and also on-demand need-driven meetings to resolve design issues. Our deep, long-term engagement allowed us to create a communal, cross-disciplinary team, lowered barriers to communication, and created a shared vision. It also gave us the opportunity to iteratively define our design goals and make design decisions that are in line with our shared mission. Considering the sensitive and complicated nature of the data and implications of such a tool in a high-stakes environment, our priority was making design decisions that address our users' needs. Doing this process justice required a substantial period of time, and we are confident that our solution would have met fewer of the core design goals if we had halted the collaboration earlier.

Benefits Gained from Co-Design. Although our co-design process took considerable time, it allowed us to share intermediate outputs from early prototypes that our users could use in their jobs. Whether to get feedback on a particular output format or to fulfill one-off requests, all of our back-and-forth with the early-stage tool outputs informed the design process while also benefiting the stakeholders. Our collaboration also allowed for the exchange of trainings and workshops across disciplines, which supported both data handling and tool usage.

Beyond Computational Tools. After pushes from lawmakers, activists, journalists, and the public, the data released from LEAs are large and messy (Sub-section 2.2). One possible reason is that LEAs might benefit from withholding the information from the public, and providing it in a way that is time- and labor-consuming as an adversarial strategy. Another reason might be that LEAs themselves are constrained by the money and resources needed to release the data in usable, structured formats. Our tooling has the potential to significantly reduce workload, but our work also taught us that there are limits to what tooling can fix. The current state of released data suggests the need for policy that not only requires data disclosures but also puts restrictions on *how* data is disclosed.

7.2 Future Work and Programming Practices

From our user study and our interaction with users, we observe that contrary to popular belief, non-technical experts are able to use text-based coding paradigms if the tools are designed to support their needs. Our user study shows participants are able to accomplish their tasks in any of the provided paradigms once they are familiar with the interface (i.e., arrived at the third task), regardless of the user experience with the paradigm. However, there was a gap in reported vs. observed performance when using the text-based interfaces (Section 6.3.1) indicating that research in improving the learnability and approachability of text-based programming could improve accessibility of programming tools.

Qualitative analysis of our user study results suggests a variety of jumping off points for future research on programming tools for non-computing experts. Here we highlight three themes:

- We observed that our visual interfaces' histograms allowed users to learn about datasets and reason about how their choices could affect program outputs. Designers may be able to build on this insight to provide users with visualizations that give an overview of program inputs and program state, to provide information that would have otherwise been available only via users' active attempts to acquire it. **Visual programming gives tool designers an opportunity to offer information by default that programmers might not think to uncover themselves.**
- In the context of the programming-by-example paradigm, we observed that participants could interact with their data and learn about the range of appropriate parameter values as they gave examples. For use cases that require low-level control over program details (adherence to a fixed program), PBE may be an unnatural fit relative to textual or visual paradigms. However, designers may be able to use elements of the PBE interaction to aid users in developing their specifications. **The PBE paradigm puts the focus on the data rather than the program structure.**
- Within the text-based paradigm, two participants were able to import outside functions and use their previous knowledge to try to achieve goals that the visual and PBE interfaces did not support. Our subjective perception of participant behaviors included the observation that participants tended to explore more freely when using the text-based paradigms. **We observed that text-based programming offers low-level control and flexibility to explore outside of the designer-provided abstractions.**

7.3 Bigger Picture: Impact of Organized and Manageable Data

Discrimination and lack of police accountability within the United States policing system [51, 60, 61] continues to tarnish the public's image of LEAs [24, 46]. Various stakeholders call for LEA transparency to increase trust and hold institutions and systems accountable [23, 43]. When LEAs share disorganized and messy data, they delay the transparency that legislators, advocates, and the public have demanded. Supporting data recipients to easily navigate and extract useful information can support journalists fighting for transparency, defense attorneys fighting for justice for their clients, and advocates demanding an equitable justice system.

One goal of our work is to present lessons learned from long-term co-designing process with stakeholders dealing with large data dumps in hopes that our reflections could be transferred to other domains where practitioners deal with large, disclosed datasets. Our aim is to provide insights for designers working on building computational tools for domain experts, particularly in high-stakes, low-resource settings. Additionally, our work focuses on disclosures from Law Enforcement Agencies, and provides an open source tool for practitioners working to hold such institutions accountable.

We will end our discussion with a direct quote from **California Senate Bill 1421 (Peace Officers: Release of Records)**.

> Concealing crucial public safety matters such as officer violations of civilians' rights, or inquires into deadly use-of-force incidents, undercuts the public faith in the legitimacy of law enforcement, makes it harder for tens of thousands of hardworking peace officers to do their jobs and endangers public safety.

Limitations. Our user study used a small participant pool. Future work could conduct similar but larger-scale comparisons with more participants, more domains, more tools from the paradigms under test, or more programming paradigms. Our user study is not an evaluation, but rather informs our design via information about the skill-sets of our users. However, the foundation of our work lies in the co-design approach, which informed most of the decisions about functionality as well as the user interaction model design. Our co-design process mainly involved two sets of users: investigative journalists and public defenders. Future work could expand beyond LEA-stakeholder interaction and include the public's needs.

8 CONCLUSION

Efforts by a variety of stakeholders have led to data disclosures from Law Enforcement Agencies for the sake of transparency and accountability. However, data released in these processes can be large, messy, and disorganized. Processing them takes time from already resource-constrained teams, such as public defenders and investigative journalists, working towards justice and transparency. This work presents a cross-disciplinary co-design approach to building a document organization tool for non-technical domain experts working on police use-of-force and misconduct data. Calling for the release of already clean and organized data from LEAs is one avenue for improving transparency, but we show an alternative—supporting teams who receive and analyse the released data. We argue for empowering such teams by (i) co-creating design goals that align with their work practices and (ii) building tools that allow them to accomplish their tasks by complimenting their existing skills. We also provide insights into the use of three programming paradigms—visual, programming-by-example, and text-based—to understand users' interactions with our tool in different paradigms. We show that with user-centered design and a deep understanding of stakeholders' design goals, it is possible to meet users where they are in terms of technical skills necessary for programming.

Across the criminal justice domain, large and messy document dumps can slow or hinder justice. Together with stakeholders, we can build tools to support processing such data and alleviate problems in vital journalistic coverage, the work of public defenders, and advocates' fights for an equitable and fair justice system.

ACKNOWLEDGMENTS

This work is supported in part by NSF grants FW-HTF 2129008, CA-HDR 1936731, and CA-HDR 2033558, as well as by gifts from Google, Microsoft, and Sigma Computing. Sarah E. Chasins is a Chan Zuckerberg Biohub Investigator. Hellina Hailu Nigatu is a SIGHPC Computational and Data Science Fellow. We thank Julie Ciccolini and the rest of the NACDL Full Disclosure Project team, all the journalists and public defenders with whom we worked over the course of the co-design process, and all our research study participants. We would also like to thank members and friends of PLAIT lab for feedback on this work.

REFERENCES

[1] [n. d.]. 34 U.S. Code § 10534 - James Guelff and Chris McCurley Body Armor Act of 2002 34 U.S. Code § 10534 - James Guelff and Chris McCurley Body Armor Act of 2002. https://www.law.cornell.edu/uscode/text/34/10534#c_2

[2] 2021. The public defense project. https://www.ischool.berkeley.edu/sites/default/files/sproject_attachments/the_public_defense_project_final_report_2021.pdf

[3] Arko Banerjee and Arun K. Pujari. 2014. Representative Based Document Clustering. In *Advanced Computing, Networking and Informatics- Volume 1 (Smart Innovation, Systems and Technologies)*, Malay Kumar Kundu, Durga Prasad Mohapatra, Amit Konar, and Aruna Chakraborty (Eds.). Springer International Publishing, Cham, 403–411. https://doi.org/10.1007/978-3-319-07353-8_47

[4] Sarah E. Chasins, Maria Mueller, and Rastislav Bodik. 2018. Rousillon: Scraping Distributed Hierarchical Web Data. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, Berlin Germany, 963–975. https://doi.org/10.1145/3242587.3242661

[5] Jennifer Cobbe, Michelle Seng Ah Lee, and Jatinder Singh. 2021. Reviewable Automated Decision-Making: A Framework for Accountable Algorithmic Systems. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*. Association for Computing Machinery, New York, NY, USA, 598–609. https://doi.org/10.1145/3442188.3445921

[6] European Commission. 2019. Ethics guidelines for trustworthy AI | Shaping Europe's digital future. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[7] Robin Cura, Amélie Beaumont, Jean-Samuel Beuscart, Samuel Coavoux, Noé Latreille de Fozières, Brenda Le Bigot, Yann Renisio, Manuel Moussallam, and Thomas Louail. 2022. Uplifting Interviews in Social Science with Individual Data Visualization: the case of Music Listening. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. ACM, New Orleans LA USA, 1–9. https://doi.org/10.1145/3491101.3503553

[8] PAUL M. LEONARDI CYNTHIA STOHL, MICHAEL STOHL. 2016. Managing Opacity: Information Visibility and the Paradox of Transparency in the Digital Age. (2016). https://ijoc.org/index.php/ijoc/article/view/4466/1530

[9] Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. 2021. Cooperative AI: machines must learn to find common ground. *Nature* 593, 7857 (May 2021), 33–36. https://doi.org/10.1038/d41586-021-01170-0 Bandiera_abtest: a Cg_type: Comment Number: 7857 Publisher: Nature Publishing Group Subject_term: Machine learning, Computer science, Society, Technology, Sociology, Human behaviour.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. http://arxiv.org/abs/1810.04805 arXiv:1810.04805 [cs].

[11] Mark Dorling and Dave White. 2015. Scratch: A Way to Logo and Python. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, Kansas City Missouri USA, 191–196. https://doi.org/10.1145/2676723.2677256

[12] Therese Enarsson, Lena Enqvist, and Markus Naarttijärvi. 2022. Approaching the human in the loop – legal perspectives on hybrid human/algorithmic decision-making in three contexts. *Information & Communications Technology Law* 31, 1 (Jan. 2022), 123–153. https://doi.org/10.1080/13600834.2021.1958860

[13] David Enrich. 2022. How Abbott Kept Sick Babies From Becoming a Scandal. *The New York Times* (Sept. 2022). https://www.nytimes.com/2022/09/06/business/abbott-baby-formula-lawsuits-jones-day.html

[14] Olivier Goletti, Kim Mens, and Felienne Hermans. 2021. Tutors' Experiences in Using Explicit Strategies in a Problem-Based Learning Introductory Programming Course. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. ACM, Virtual Event Germany, 157–163. https://doi.org/10.1145/3430665.3456348

[15] N Guibert, L Guittet, and P Girard. [n. d.]. A STUDY OF THE EFFICIENCY OF AN ALTERNATIVE PROGRAMMING PARADIGM TO TEACH THE BASICS OF PROGRAMMING. https://www.lias-lab.fr/publications/7164/2005-WCCE-Guibert.pdf

[16] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. 2017. *Program synthesis*. Number 4.2017, 1-2 in Foundations and trends in programming languages. Now Publishers, Hanover, MA Delft.

[17] Kyle Wm Hall, Adam J. Bradley, Uta Hinrichs, Samuel Huron, Jo Wood, Christopher Collins, and Sheelagh Carpendale. 2020. Design by Immersion: A Transdisciplinary Approach to Problem-Driven Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (Jan. 2020), 109–118. https://doi.org/10.1109/TVCG.2019.2934790 arXiv:1908.00559 [cs].

[18] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Advances in Psychology*, Peter A. Hancock and Najmedin Meshkati (Eds.). Human Mental Workload, Vol. 52. North-Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[19] Gillian R. Hayes. 2011. The relationship of action research to human-computer interaction. *ACM Transactions on Computer-Human Interaction* 18, 3 (July 2011), 1–20. https://doi.org/10.1145/1993060.1993065

[20] Chris Hess and Sarah E. Chasins. 2022. Informing Housing Policy through Web Automation: Lessons for Designing Programming Tools for Domain Experts. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3491101.3503575

[21] Matt-Heun Hong, Lauren A. Marsh, Jessica L. Feuston, Janet Ruppert, Jed R. Brubaker, and Danielle Albers Szafir. 2022. Scholastic: Graphical Human-AI Collaboration for Inductive and Interpretive Text Analysis. In *The 35th Annual ACM Symposium on User Interface Software and Technology*. ACM, Bend OR USA, 1–12. https://doi.org/10.1145/3526113.3545681

[22] Matthew Horton, Janet C. Read, Emanuela Mazzone, Gavin Sim, and Daniel Fitton. 2012. School friendly participatory research activities with children. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. ACM, Austin Texas USA, 2099–2104. https://doi.org/10.1145/2212776.2223759

[23] Invisible Institute. [n. d.]. Citizens Police Data Project. https://invisible.institute/police-data

[24] Brian A Jackson. 2015. Strengthening Trust Between Police and the Public in an Era of Increasing Transparency. (Oct. 2015).

[25] Dhanya Jayagopal, Justin Lubin, and Sarah E Chasins. 2022. Exploring the Learnability of Program Synthesizers by Novice Programmers. (2022), 15.

[26] Mike Jeffries Jon Swords, Kye Askins and Catherine Butcher. 2013. Geographic visualisation: lessons for learning and teaching. https://doi.org/10.11120/plan.2013.00001

[27] Amy J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. 2011. The state of the art in end-user software engineering. *Comput. Surveys* 43, 3 (April 2011), 1–44. https://doi.org/10.1145/1922649.1922658

[28] Brendan Lawlor and Roy D Sleator. 2021. The roles of code in biology - Brendan Lawlor, Roy D Sleator, 2021. https://journals.sagepub.com/doi/full/10.1177/00368504211010570

[29] Kurt Lewin. 1946. Action Research and Minority Problems. *Journal of Social Issues* 2, 4 (1946), 34–46. https://doi.org/10.1111/j.1540-4560.1946.tb02295.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-4560.1946.tb02295.x.

[30] Nina McCurdy, Jason Dykes, and Miriah Meyer. 2016. Action Design Research and Visualization Design. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*. ACM, Baltimore MD USA, 10–18. https://doi.org/10.1145/2993901.2993916

[31] Vivek Mehta, Seema Bawa, and Jasmeet Singh. 2021. WEClustering: word embeddings based text clustering technique for large datasets. *Complex & Intelligent Systems* 7, 6 (Dec. 2021), 3211–3224. https://doi.org/10.1007/s40747-021-00512-9

[32] Carina Miller. [n. d.]. THE PARADOX OF S.B. 1421: A NEW TOOL TO SHED LIGHT ON POLICE MISCONDUCT AND A PERVERSE INCENTIVE TO COVER IT UP. ([n. d.]). https://www.swlaw.edu/sites/default/files/2021-06/49SwLRev537.pdf

[33] Michael J. Muller and Sarah Kuhn. 1993. Participatory design. *Commun. ACM* 36, 6 (June 1993), 24–28. https://doi.org/10.1145/153571.255960

[34] Brad Myers, Sun Young Park, Yoko Nakano, Greg Mueller, and Amy Ko. 2008. How designers design and program interactive behaviors. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*. 177–184. https://doi.org/10.1109/VLHCC.2008.4639081 ISSN: 1943-6106.

[35] Jonah Newman and Geoff Hing. 2017. Chicago Reporter. https://www.chicagoreporter.com/99reforms/

[36] nij. 2020. Overview of Police Use of Force. https://nij.ojp.gov/topics/articles/overview-police-use-force

[37] Zheng-Yu Niu, Dong-Hong Ji, and Chew-Lim Tan. 2004. Document clustering based on cluster validation. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM '04)*. Association for Computing Machinery, New York, NY, USA, 501–506. https://doi.org/10.1145/1031171.1031267

[38] Vesa Norilo and Alejandro Olarte. 2020. A Visual Programming Interface for Digital Luthiery: Implementing Circuits with Veneer. *Computer Music Journal* 44, 4 (Dec. 2020), 8–25. https://doi.org/10.1162/comj_a_00578

[39] Chris Norval, Kristin Cornelius, Jennifer Cobbe, and Jatinder Singh. 2022. Disclosure by Design: Designing information disclosures to support meaningful transparency and accountability. In *2022 ACM Conference on Fairness, Accountability, and Transparency*. ACM, Seoul Republic of Korea, 679–690. https://doi.org/10.1145/3531146.3533133

[40] Erik Pasternak, Rachel Fenichel, and Andrew N. Marshall. 2017. Tips for creating a block language with blockly. In *2017 IEEE Blocks and Beyond Workshop (B&B)*. 21–24. https://doi.org/10.1109/BLOCKS.2017.8120404

[41] Lindsay Poirier. 2022. Accountable Data: The Politics and Pragmatics of Disclosure Datasets. In *2022 ACM Conference on Fairness, Accountability, and Transparency*. ACM, Seoul Republic of Korea, 1446–1456. https://doi.org/10.1145/3531146.3533201

[42] Washington Post. 2023. Police shootings database 2015-2023: Search by race, age, department. https://www.washingtonpost.com/investigations/interactive/2022/police-shootings-database-2015-2022-search-by-race-age-department/

[43] The Policing Project. 2019. Grounding Grassroots Advocacy: Salt Lake City Activists Organize to Tackle Wide-Ranging Reforms. https://www.policingproject.org/news-main/2019/3/26/grounding-grassroots-advocacy

[44] Ronald L. Rivest. 1992. *The MD5 Message-Digest Algorithm*. Request for Comments RFC 1321. Internet Engineering Task Force. https://doi.org/10.17487/RFC1321 Num Pages: 21.

[45] M.B. Rosson, J. Ballin, and J. Rode. 2005. Who, what, and how: a survey of informal and professional Web developers. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. 199–206. https://doi.org/10.1109/VLHCC.2005.73 ISSN: 1943-6106.

[46] Michael S. Schmidt. 2015. A Call to Better Track Police Use of Guns. *The New York Times* (Jan. 2015). https://www.nytimes.com/2015/01/16/us/politics/holder-urges-better-data-for-shootings-involving-police.html

[47] Michael Sedlmair, Miriah Meyer, and Tamara Munzner. 2012. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec. 2012), 2431–2440. https://doi.org/10.1109/TVCG.2012.213 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[48] Cathrine Seidelin, Yvonne Dittrich, and Erik Grönvall. 2020. Co-designing Data Experiments: Domain Experts' Exploration and Experimentation with self-selected Data Sources. In *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*. ACM, Tallinn Estonia, 1–11. https://doi.org/10.1145/3419249.3420152

[49] Logan Stapleton, Min Hun Lee, Diana Qing, Marya Wright, Alexandra Chouldechova, Ken Holstein, Zhiwei Steven Wu, and Haiyi Zhu. 2022. Imagining new futures beyond predictive systems in child welfare: A qualitative study with impacted stakeholders. In *2022 ACM Conference on Fairness, Accountability, and Transparency*. ACM, Seoul Republic of Korea, 1162–1177. https://doi.org/10.1145/3531146.3533177

[50] Michael Steinbach, George Karypis, and Vipin Kumar. [n. d.]. A Comparison of Document Clustering Techniques. ([n. d.]).

[51] Eric A. Stewart, Eric P. Baumer, Rod K. Brunson, and Ronald L. Simons. 2009. Neighborhood Racial Context and Perceptions of Police-Based Racial Discrimination Among Black Youth*. *Criminology* 47, 3 (2009), 847–887. https://doi.org/10.1111/j.1745-9125.2009.00159.x

[52] Kathryn T Stolee. [n. d.]. Kodu Language and Grammar Specification. ([n. d.]).

[53] Ram Subramanian and Leily Arzi. 2021. State Policing Reforms Since George Floyd's Murder | Brennan Center for Justice. https://www.brennancenter.org/our-work/research-reports/state-policing-reforms-george-floyds-murder

[54] S Thirumaran and R Nagarajan. 2021. Split and rule algorithm for documents clustering in big data of research articles on Google scholar. *IOP Conference Series: Materials Science and Engineering* 1070, 1 (Feb. 2021), 012068. https://doi.org/10.1088/1757-899X/1070/1/012068

[55] Anna Trunk, Hendrik Birkel, and Evi Hartmann. 2020. On the current state of combining human and artificial intelligence for strategic organizational decision making. *Business Research* 13, 3 (Nov. 2020), 875–919. https://doi.org/10.1007/s40685-020-00133-x

[56] Rachel B. Warren and Niloufar Salehi. 2022. Trial by File Formats: Exploring Public Defenders' Challenges Working with Novel Surveillance Data. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW1 (March 2022), 1–26. https://doi.org/10.1145/3512914

[57] Elliott Wen, Tharindu Indrajith Kaluarachchi, Shamane Siriwardhana, Vanessa Tang, Mark Billinghurst, Robert W. Lindeman, Richard Yao, James Lin, and

Suranga Nanayakkara. 2022. VRhook: A Data Collection Tool for VR Motion Sickness Research. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22)*. Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3526113.3545656

[58] Cara Wilson, Margot Brereton, Bernd Ploderer, and Laurianne Sitbon. 2019. Co-Design Beyond Words: 'Moments of Interaction' with Minimally-Verbal Children on the Autism Spectrum. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–15. https://doi.org/10.1145/3290605.3300251

[59] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2022. LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding. http://arxiv.org/abs/2012.14740 arXiv:2012.14740 [cs].

[60] Katharine H. Zeiders, Adriana J. Umaña-Taylor, Stefanie Martinez-Fuentes, Kimberly A. Updegraff, Sara Douglass Bayless, and Laudan B. Jahromi. 2021. Latina/o youths' discrimination experiences in the U.S. Southwest: Estimates from three studies. *Applied Developmental Science* 25, 1 (2021), 51–61. https://doi.org/10.1080/10888691.2018.1527695 arXiv:https://doi.org/10.1080/10888691.2018.1527695 PMID: 33716491.

[61] Katharine H. Zeiders, Adriana J. Umaña-Taylor, Selena Carbajal, and Alexandria Pech. 2021. Police discrimination among Black, Latina/x/o, and White adolescents: Examining frequency and relations to academic functioning. *Journal of Adolescence* 90, 1 (July 2021), 91–99. https://doi.org/10.1016/j.adolescence.2021.06.001

[62] Yu Zhang, Yun Wang, Haidong Zhang, Bin Zhu, Siming Chen, and Dongmei Zhang. 2022. OneLabeler: A Flexible System for Building Data Labeling Tools. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–22. https://doi.org/10.1145/3491102.3517612

A PAGE TYPE CLASSIFICATION THROUGH FINE-TUNING

We fine-tuned LayOutLMv2 with around 6000 pages of manually annotated data for sequence classification. We found LayOutLMv2 to perform well for our datasets as an open sourced document processing AI model. The training accuracy of our fine-tuning efforts was 83.78%, validation accuracy was 84.11%, and test accuracy was 83.52%. Finetuning the model took 6 hours for model training and 6 hours for data labeling.

B USER-STUDY TASKS

Here, we present the content of the tasks document that the participants saw during the think-out-loud user study sessions. We present the tasks for both Exact Duplicate and Near Duplicate Detection. The Scenarios and the the tasks are inspired by real scenarios we observed from our interactions. We have two tasks: Fixed tasks where the participants are asked to write a specific program assigned by the researcher and Exploration tasks where the participants are given the option to explore the interface and write a program they want. Further details about the tasks is in Section 6.2. We have anonymized the specific locations (specific County and District Police Complaints Office) for the purpose of this paper.

B.1 Exact Duplicate Detection

Scenario. You are in a team of investigative journalists who are interested in police misconduct cases. You just got data dumps from a County in the US with several case files in PDF form. You are aware that there are several PDF files that are exact copies of each other. Additionally, you know that some of the files are wholly or partially included in larger PDF files. There are also files that share standard forms/statements.

Data. There are over 1500 PDF files with a total of 62,000 pages with pages per PDF ranging from 1 to 829 pages.

Path to dataset. : /home/user/user study dataset/

Fixed Tasks.

- Your team wants to look at case files where over half the pages in one document appears in another document. Write a program that will find duplicate documents where more than 50% of pages in one document appear in the other.
- The county manager just contacted you and told you that they have a 100 page manuscript for training officers that is included in almost all of the larger case-files. Write a program which will allow you to identify documents where at least 100 of the pages in one document are found in another document for documents with a number of pages at least 150 and at most 500.

Exploration Task.

- Write a program that you would be interested in working with and believe will get the duplicate documents in this dataset.

B.2 Near Duplicate Detection

Scenario. You are in a team of public defenders who are investigating complaints of police misconduct cases. You just got data dumps from a US District's Police Complaints Office with several complaints in PDF format. You are aware that there are several PDF files that have near duplicate pages due to difference in level and type of redaction, difference in type of scanning, notes taken on top of PDFs etc.

Data. There are over 862 PDF files with a total of 12,400 pages. A single PDF could have pages of type: 'form', 'image', 'narrative', 'interview', or 'other'. In this task, we are interested in **form** and **image** page types.

Path to dataset. : /home/user/user study dataset/

Exploration Task.

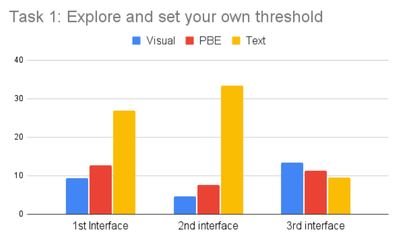
- You just received the data dump described above and your team is trying to find near duplicate pages so you can organize your files. Explore the different correlation values in your dataset and set a threshold you believe will get the near duplicates for 'form' and 'image' page types.

Fixed Tasks.

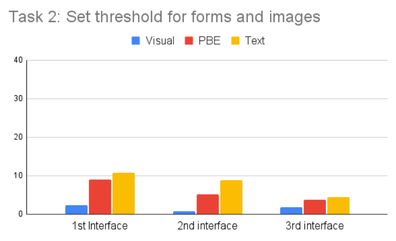
- Your teammate has done the exploration for you! (Bless their heart) Set the threshold for 'form' to be 0.89 and 'image' 0.95.
- You are told that you have far more images that are near duplicates than any other page type. Set a correlation threshold of 0.92 for 'image' page type.

C DOT FUNCTIONS

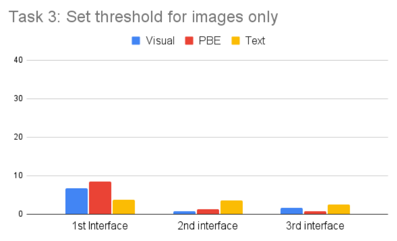
Here, we give an overview of the functions that are present in DOT. We present the functions, their return values and a description of what they do. We have dedicated one table for each of the three features of DOT: Data Cleaning in Table 4, Data Extraction in Table 5, and Data Extraction in Table 6.



(a) Time taken by participants for exploration task.

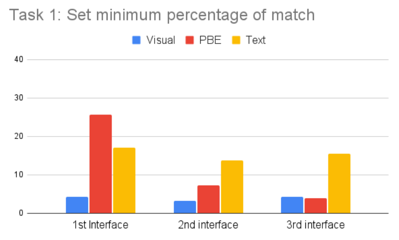


(b) Time taken by participants for fixed task to set correlation threshold for image and form pages.

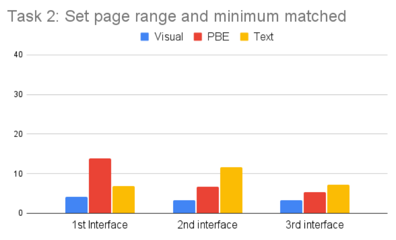


(c) Time taken by participants for fixed task of setting correlation threshold for just image pages.

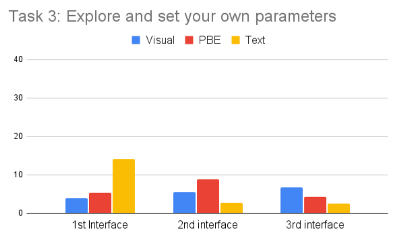
Figure 4: Bar charts showing the average time taken by participants in Near Duplicate Detection. The x-axis of each histogram shows whether the interface was the first, second, or third interface in the sequence a set of participants saw.



(a) Time taken by participants for fixed task of setting percentage threshold.



(b) Time taken by participants for fixed task to set file page range and number of matched pages.



(c) Time taken by participants for exploration task.

Figure 5: Histograms showing the average time taken by participants in Exact Duplicate Detection. The x-axis of each histogram shows whether the interface was the first, second, or third interface in the sequence a set of participants saw.

D FORMATIVE STUDY TASK COMPLETION TIME

In this section, we include detailed breakdowns of participant task completion time, with times separated according to whether participants saw each programming paradigm first, second, or third in their sessions. See Figure 4 for details of Near Duplicate Detection times. See Figure 5 for details of Exact Duplicate Detection times.

E USER STUDY PARTICIPANTS

In Table 3, we present full list of participants and details collected from a pre-interview survey on field they are currently working in, their programming experience, other programming tools they have used prior to this study, the amount of time that has elapsed since they wrote their first program, their comfort in writing programs on a scale of 1-to-5, and weather or not they identify as a programmer.

Table 3: Self-reported information from user study participants. We had a total of 18 participants in our study. Only two self identified as a programmer, regardless of their programming experience. The highest level of comfort for writing programs was 3 (n=5) and the most frequent was 1 (n=7).

Participant	Field	Programming Experience	Programming Tools Used	Time since first program	Programming comfort (1-5)	Identify as
PE0	Journalism	Self-taught	Excel, Jupyter Notebook	2 months	1	non-programmer
PE1	Journalism	Self-taught	Excel, Jupyter Notebook, Tabeleau, R	20 years	2	non-programmer
PE2	Journalism	No Experience	Excel, Jupyter Notebook	NA	1	non-programmer
PE3	Communications	No Experience	Excel	NA	1	non-programmer
PE4	Journalism	Self-taught	Excel, Jupyter Notebook	1 day	3	non-programmer
PE5	Journalism	Self-taught	Excel, Jupyter Notebook	3 years	3	non-programmer
PE6	Data Reporter	Formally trained	Excel, Jupyter Notebook,Tabeleau	6 months	2	non-programmer
PE7	Journalism	Self-taught	Excel, Jupyter Notebook,Tabeleau	3 years	3	programmer
PE8	Journalism	Self-taught	Excel, Jupyter Notebook,Tabeleau, Google Collab	20 years	2	non-programmer
PE9	Journalism	No Experience	Excel, Tabeleau	NA	1	non-programmer
PE10	Journalism	No Experience	Excel, Jupyter Notebook,Tabeleau	1 year	2	non-programmer
PE11	Journalism	No Experience	Excel	1 day	1	non-programmer
PN0	Journalism	Self-taught	Excel, Jupyter Notebook, Tableau, R Studio, command line, Google collab, GitHub	20 years	2	non-programmer
PN1	Social Justice, Director	Self-taught	Excel, Jupyter Notebook, Tabeleau	7 years	3	programmer
PN2	Economics	Self-taught	Excel, Jupyter Notebook	2 years	2	non-programmer
PN3	Defense Investigator	No Experience	Excel	NA	1	non-programmer
PN4	Investigative Specialist	No Experience	Excel	NA	1	non-programmer
PN5	Journalism	Self-taught with some formal training	Excel, Jupyter Notebook,Tabeleau	3 years	3	programmer

Table 4: Data Cleaning Functions. Includes functions for both Exact Duplicate and Near Duplicate Detection.

Function Name	Return Value	Description
HashPages(dataset_path, num_rep)	DataFrame	Hashes all the pages in the given path and returns a DataFrame with information about pages with the same hash value. Users can control how many replicated pages a single page should have using *num_rep*.
threshold_by_percent(file_df, min_percent_value, max_percent_value, min_page_in_file, max_page_in_file)	Dictionary	Filters the files from the *file_df* based on the percentage of pages in one document found in another. Users set the parameters to control how many pages the file needs to have and where the percent threshold should be.
threshold_by_matched_pages(file_df, min_page_in_file_match, max_page_in_file_match, min_page_in_file, max_page_in_file)	Dictionary	Filters the files from the *file_df* based on the number pages in one document found in another. Users set the parameters to control how many pages the file needs to have and how many of those pages need to be shared across the pair of matched files.
print_duplicate_info(List[conditions], output_path)	None	Displays a list of pairs of files that match the passed conditions along with what percent and how many pages in each file are found in the other. Users can specify where to save the printed output.
ClassifyAndGetPairCorrelation(dataset_path, List[page_type])	DataFrame	Classifies each page using fine-tuned LayOutLMv2 and computes pair wise correlation value for the indicated page type.
print_near_duplicate_information(condition)	None	Displays a list of pairs of files that have pages matching the passed correlation threshold along with how many pages in each document are near duplicates in its pair.
visualize_file_pairs(path_one, path_two)	Tuple	Displays pair of files specified in the passed paths with matching pages colored green and non-matching pages colored in red.

Table 5: Data Extraction Functions.

Function Name	Return Value	Description
extract_with_bbox(List[bbox],encoded_dataset, processor, lable_df, reg_expression)	List[String]	Searches for String matching the given Regular Expression within the provided bounding box within every page found in the lable_df. lable_df has information about pages with a particular format (form, narrative, interview, image, handwriting)
plot_extraction_suggestions(pages, path, file_name)	None	Displays a where the bounding box specified falls on the pages specified so users can adjust bbox location if needed.
get_named_entity(file_df, entity_type, page_type)	List[String]	Uses BERT to extracted Named Entities passed to the function within the pages that have the *page_type* specified.
clean_entity_list(stop_words, List[entities], minLen, maxLen, maxFreq)	List[String]	Takes the list of extracted entities and performs normalization of date format, limiting String length, and removing stop words.

Table 6: Data Organization Functions.

Function Name	Return Value	Description
set_entities_doc(page_df, doc_df)	DataFrame	Sets extracted values from each pages found in *page_df* to each document and presents a DataFrame.
highlight_text(doc_df, column, List[entities])	List[int]	Returns indexes of rows in the passed dataframe where the names and dates match and highlights the *text* in each of the rows.
highlight_rows(doc_df, column, List[casenumbers])	List[int]	Returns indexes of rows in the passed dataframe where case numbers match and highlights those *rows*.
get_eligible_case(List[Object[Case]], doc_df, similarity_score)	Tuple	Find documents with matching entities for passed to the function and return their index. For case numbers, since OCR might make character mistakes, users can specify a String Edit distance threshold using *similarity_score*.
split_large_files(doc_df, extracted_info_df, min_page_limit)	DataFrame	Break up large PDFs with page number above the limit passed based on the provided page type and add index on the file names to distinguish they are split.
save_case_groups(List[indicies])	None	Saves the information about each of the files found in the rows indicated by the passed indices into a Python Object which also stores the extracted entities from each file in the group.
output_organized_cases(List[Object[case]])	None	Displays the final list of group of files belonging to individual case along with the entities from each of the files in each group.