HiLT: A Library for Generating Human-in-the-Loop Data Transformation GUIs

Sora Kanosue, Xiaorui Liu, Parker Ziegler, Sarah E. Chasins

Outline

- **Motivation**
- Our intervention: HiLT
- Tour through a HiLT program
- Evaluative user study results

What are Data Transformation GUIs?

	Guided User Interaction	**Open-Ended User Interaction**
Data Transformation	?	Tableau Trifacta Google Sheets
Data Display	Streamlit Mavo Django	mage PI2

Formative Study Setup

- 17 Participants
- 2 hour sessions
- 3 tools: HiLT_0, Streamlit, Django

Research Questions:

- What barriers do programmers face in using standard interface-building tools to develop custom data transformation GUIs?
- What barriers do programmers face in using HiLT_0 to develop custom data transformation GUIs?

What Did We Learn?

Building human-in-the-loop data transformation GUIs is *hard*.

- Complex programs
- Assumptions about
  - Fixed data structures
  - Fixed numbers of datasets
- Difficulty persisting data

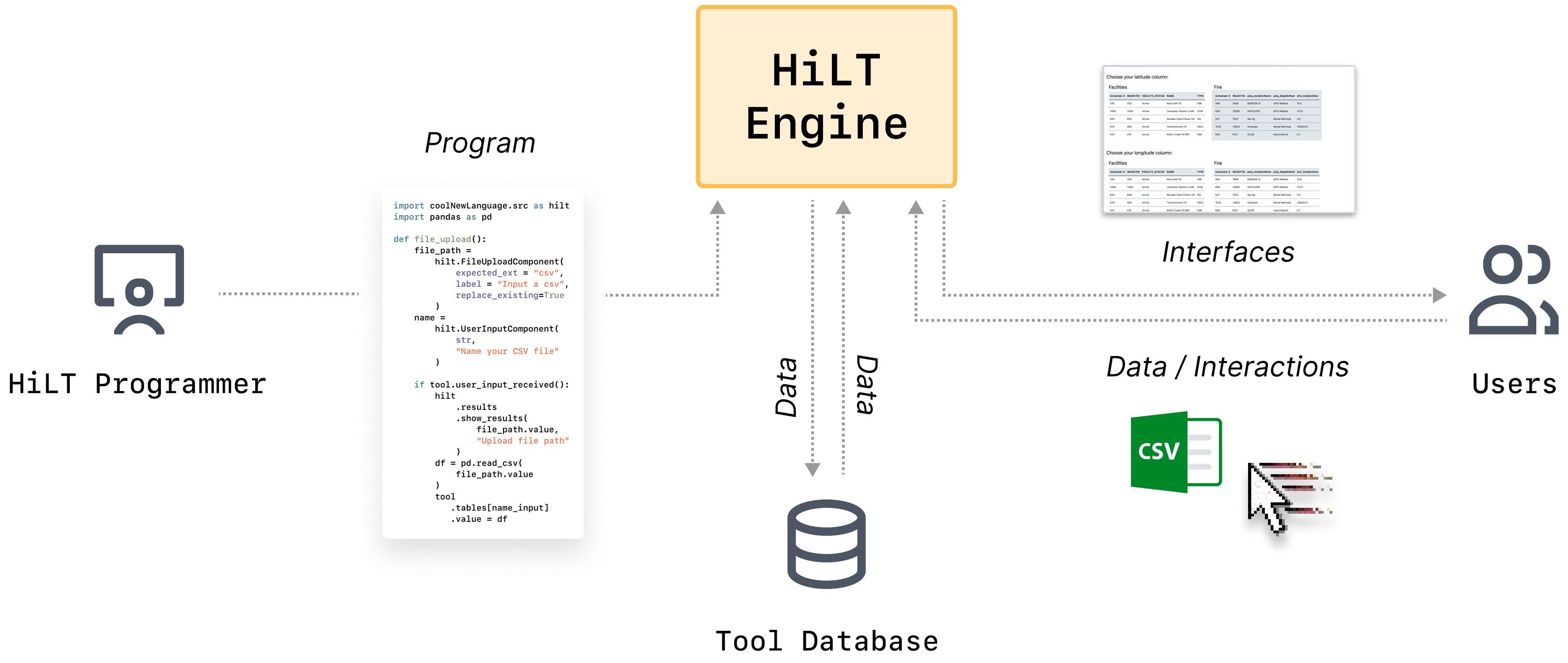
Formative Study → Design Goals

- **Design Goal 1: Guiding the User.** Output GUIs must walk the user through the process of their task; no open-ended exploration
- **Design Goal 2: Importing Data.** Output GUIs must accept different input data shapes, so users can bring their own diverse data
- **Design Goal 3: Transforming Data.** Output GUIs must be able to change the shape of data based on users' interactions

Outline

- Motivation
- **Our intervention: HiLT**
- Tour through a HiLT program
- Evaluative user study results

HiLT's Interaction Model

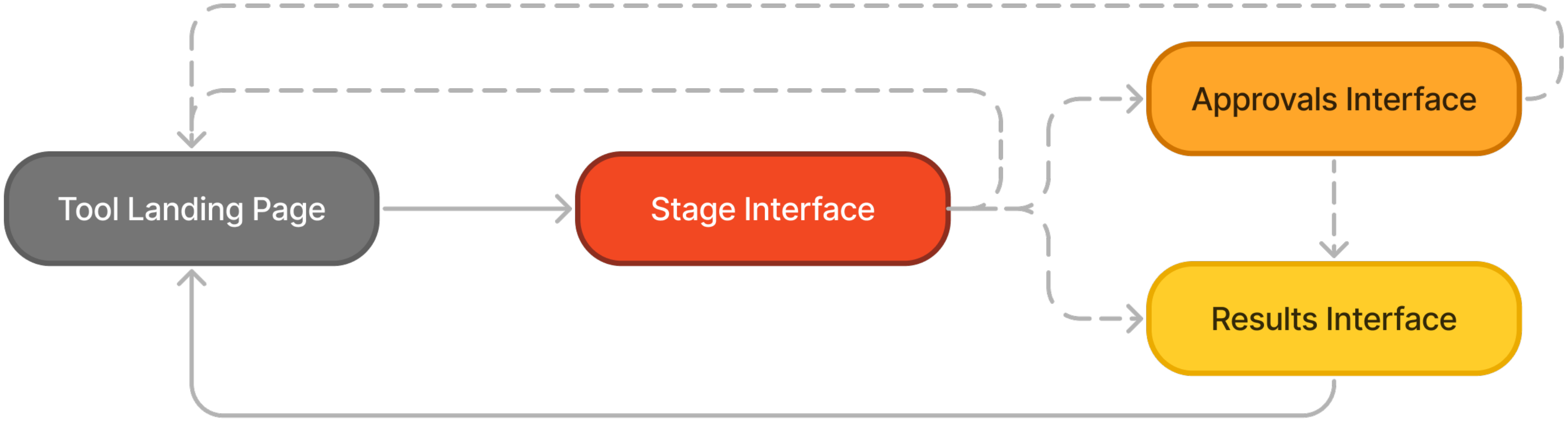


```
import coolNewLanguage.src as hilt
import pandas as pd

def file_upload():
    file_path =
        hilt.FileUploadComponent(
            expected_ext = "csv",
            label = "Input a csv",
            replace_existing=True
        )
    name =
        hilt.UserInputComponent(
            str,
            "Name your CSV file"
        )

    if tool.user_input_received():
        hilt
            .results
            .show_results(
                file_path.value,
                "Upload file path"
            )
        df = pd.read_csv(
            file_path.value
        )
        tool
           .tables[name_input]
           .value = df
```

HiLT Core APIs

- **Tool:** Parent class of a given data transformation GUI
- **Stage:** Represents a single step of a data transformation workflow, typically associated with a single function
- **Component:** Corresponds to a widget on a webpage accepting user input
- **Tables:** Representation of data in a Tool's underlying database
- **Approvals:** Flow for getting user confirmations before committing changes to a database
- **Results:** Gives programmer control of what a user sees after providing input

End User Flow



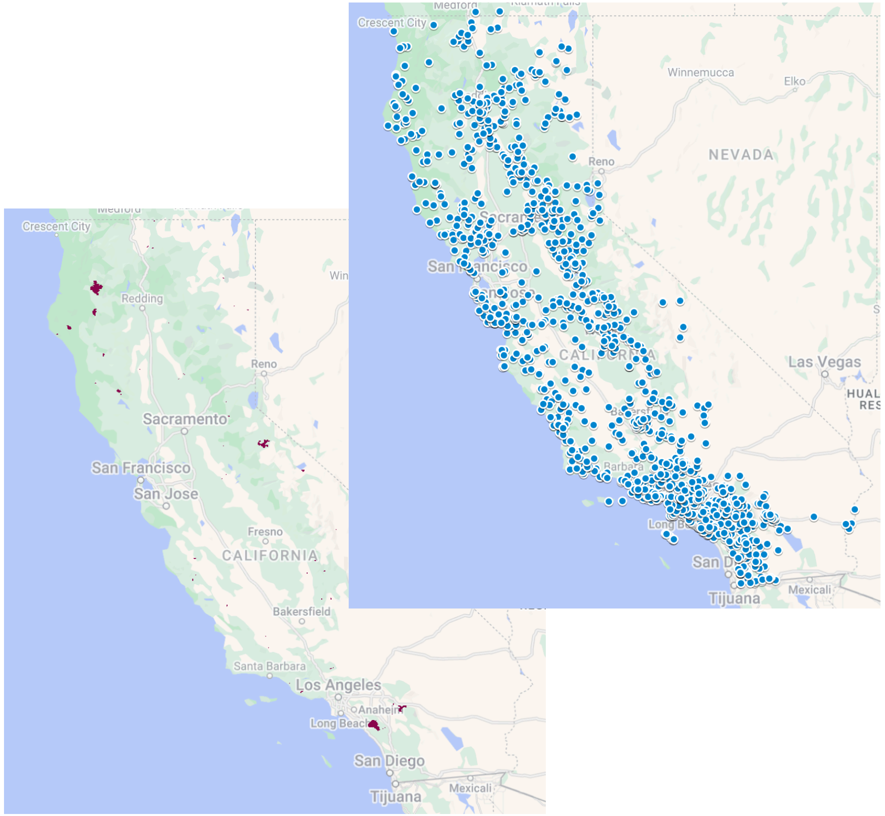
Outline

- Motivation
- Our intervention: HiLT
- **Tour through a HiLT program**
- Evaluative user study results

Evaluative User Study

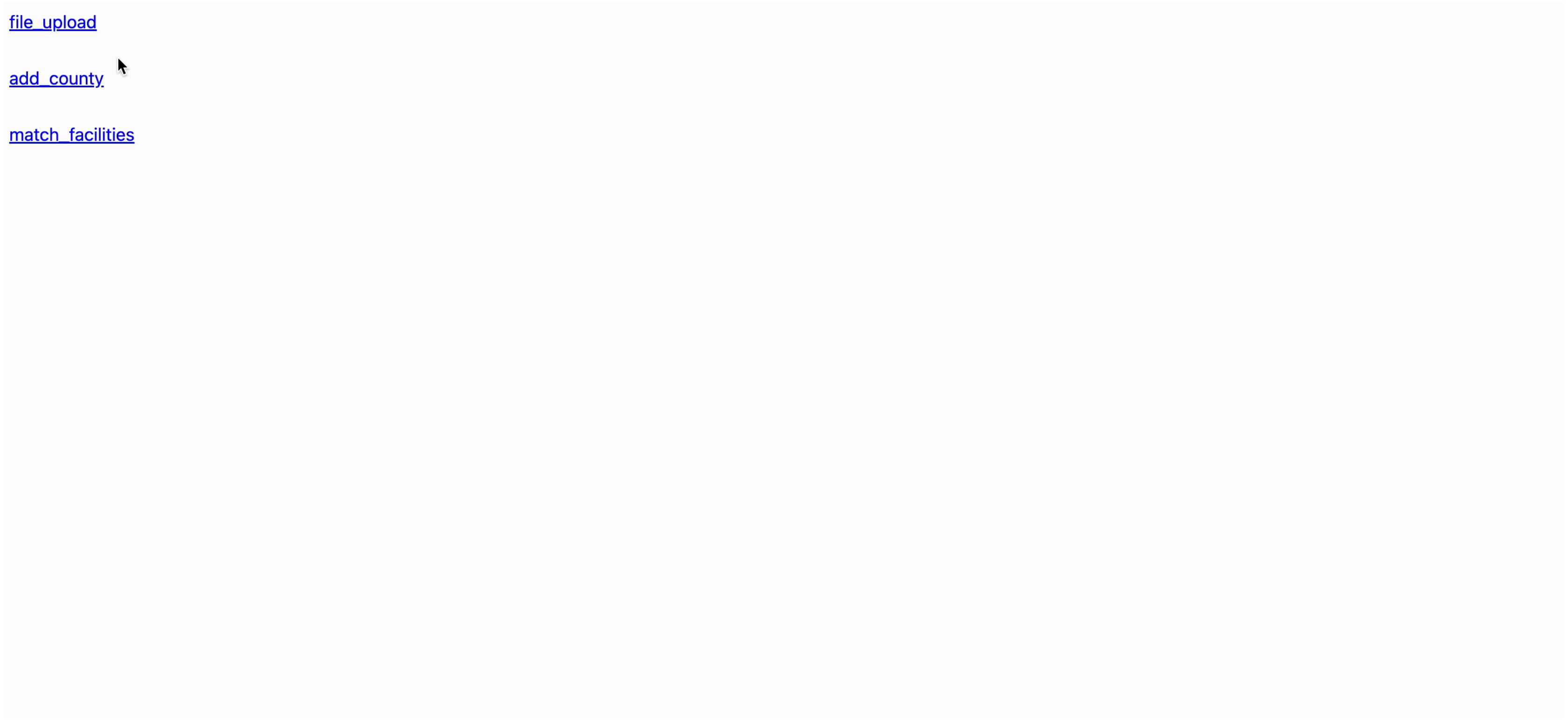
- 16 participants
- Comfortable with Python
- Building a single data transformation GUI broken down into 5 tasks
- 1 hour with HiLT, 1 hour with Streamlit

Evaluative Study - Context



1. Data upload
2. Data augmentation with counties
3. User approvals for counties
4. Data matching by county
5. User approvals for matches

```
def coords_to_county(lat: str, lon: str):
    import json
    with open('counties.json') as f:
        counties = json.load(f)
    return counties[f'{lat}, {lon}']

tool = hilt.Tool('Wildfires')

def file_upload():
    file_path = hilt.FileUploadComponent(expected_ext = "csv", label = "Input a CSV file", replace_existing=True)
    name_input = hilt.UserInputComponent(str, "Name your CSV file: ")
    if tool.user_input_received():
        df = pd.read_csv(file_path.value)
        tool.tables[name_input.value] = df
        hilt.results.show_results((file_path.value, "Uploaded file path: ”))

def add_county():
    lat_selector = hilt.ColumnSelectorComponent("Choose your latitude column:")
    lon_selector = hilt.ColumnSelectorComponent("Choose your longitude column:")
    if tool.user_input_received():
        lat_col = lat_selector.column_names[0]
        lon_col = lon_selector.column_names[0]
        df = tool.tables[lat_selector.table_name]
        table_name = lat_selector.table_name
        df['COUNTY'] = df.apply(lambda row: coords_to_county(row[lat_col], row[lon_col]), axis=1)
        tool.tables[table_name] = df
        hilt.approvals.get_user_approvals()
        hilt.results.show_results((tool.tables[table_name], "Added counties to table: ”))
                                                    .
                                                    .
                                                    .
tool.add_stage('file_upload', file_upload)
tool.add_stage('add_county', add_county)
tool.run()
```



file_upload

add_county

match_facilities

```
def coords_to_county(lat: str, lon: str):
    import json
    with open('counties.json') as f:
        counties = json.load(f)
    return counties[f'{lat}, {lon}']

tool = hilt.Tool('Wildfires')

def file_upload():
    file_path = hilt.FileUploadComponent(expected_ext = "csv", label = "Input a CSV file", replace_existing=True)
    name_input = hilt.UserInputComponent(str, "Name your CSV file: ")
    if tool.user_input_received():
        df = pd.read_csv(file_path.value)
        tool.tables[name_input.value] = df
        hilt.results.show_results((file_path.value, "Uploaded file path: ”))

def add_county():
    lat_selector = hilt.ColumnSelectorComponent("Choose your latitude column:")
    lon_selector = hilt.ColumnSelectorComponent("Choose your longitude column:")
    if tool.user_input_received():
        lat_col = lat_selector.column_names[0]
        lon_col = lon_selector.column_names[0]
        df = tool.tables[lat_selector.table_name]
        table_name = lat_selector.table_name
        df['COUNTY'] = df.apply(lambda row: coords_to_county(row[lat_col], row[lon_col]), axis=1)
        tool.tables[table_name] = df
        hilt.approvals.get_user_approvals()
        hilt.results.show_results((tool.tables[table_name], "Added counties to table: ”))
                                                            .
                                                            .
                                                            .
tool.add_stage('file_upload', file_upload)
tool.add_stage('add_county', add_county)
tool.run()
```

```python
def coords_to_county(lat: str, lon: str):
    import json
    with open('counties.json') as f:
        counties = json.load(f)
    return counties[f'{lat}, {lon}']

tool = hilt.Tool('Wildfires')

def file_upload():
    file_path = hilt.FileUploadComponent(expected_ext = "csv", label = "Input a CSV file", replace_existing=True)
    name_input = hilt.UserInputComponent(str, "Name your CSV file: ")
    if tool.user_input_received():
        df = pd.read_csv(file_path.value)
        tool.tables[name_input.value] = df
        hilt.results.show_results((file_path.value, "Uploaded file path: ”))

def add_county():
    lat_selector = hilt.ColumnSelectorComponent("Choose your latitude column:")
    lon_selector = hilt.ColumnSelectorComponent("Choose your longitude column:")
    if tool.user_input_received():
        lat_col = lat_selector.column_names[0]
        lon_col = lon_selector.column_names[0]
        df = tool.tables[lat_selector.table_name]
        table_name = lat_selector.table_name
        df['COUNTY'] = df.apply(lambda row: coords_to_county(row[lat_col], row[lon_col]), axis=1)
        tool.tables[table_name] = df
        hilt.approvals.get_user_approvals()
        hilt.results.show_results((tool.tables[table_name], "Added counties to table: ”))
                                                            .
                                                            .
                                                            .
tool.add_stage('file_upload', file_upload)
tool.add_stage('add_county', add_county)
tool.run()
```

```
def coords_to_county(lat: str, lon: str):
    import json
    with open('counties.json') as f:
        counties = json.load(f)
    return counties[f'{lat}, {lon}']

tool = hilt.Tool('Wildfires')

def file_upload():
    file_path = hilt.FileUploadComponent(expected_ext = "csv", label = "Input a CSV file", replace_existing=True)
    name_input = hilt.UserInputComponent(str, "Name your CSV file: ")
    if tool.user_input_received():
        df = pd.read_csv(file_path.value)
        tool.tables[name_input.value] = df
        hilt.results.show_results((file_path.value, "Uploaded file path: ”))

def add_county():
    lat_selector = hilt.ColumnSelectorComponent("Choose your latitude column:")
    lon_selector = hilt.ColumnSelectorComponent("Choose your longitude column:")
    if tool.user_input_received():
        lat_col = lat_selector.column_names[0]
        lon_col = lon_selector.column_names[0]
        df = tool.tables[lat_selector.table_name]
        table_name = lat_selector.table_name
        df['COUNTY'] = df.apply(lambda row: coords_to_county(row[lat_col], row[lon_col]), axis=1)
        tool.tables[table_name] = df
        hilt.approvals.get_user_approvals()
        hilt.results.show_results((tool.tables[table_name], "Added counties to table: ”))
                                                    .
                                                    .
                                                    .

tool.add_stage('file_upload', file_upload)
tool.add_stage('add_county', add_county)
tool.run()
```

```
def coords_to_county(lat: str, lon: str):
    import json
    with open('counties.json') as f:
        counties = json.load(f)
    return counties[f'{lat}, {lon}']

tool = hilt.Tool('Wildfires')

def file_upload():
    file_path = hilt.FileUploadComponent(expected_ext = "csv", label = "Input a CSV file", replace_existing=True)
    name_input = hilt.UserInputComponent(str, "Name your CSV file: ")
    if tool.user_input_received():
        df = pd.read_csv(file_path.value)
        tool.tables[name_input.value] = df
        hilt.results.show_results((file_path.value, "Uploaded file path: ”))

def add_county():
    lat_selector = hilt.ColumnSelectorComponent("Choose your latitude column:")
    lon_selector = hilt.ColumnSelectorComponent("Choose your longitude column:")
    if tool.user_input_received():
        lat_col = lat_selector.column_names[0]
        lon_col = lon_selector.column_names[0]
        df = tool.tables[lat_selector.table_name]
        table_name = lat_selector.table_name
        df['COUNTY'] = df.apply(lambda row: coords_to_county(row[lat_col], row[lon_col]), axis=1)
        tool.tables[table_name] = df
        hilt.approvals.get_user_approvals()
        hilt.results.show_results((tool.tables[table_name], "Added counties to table: ”))
                                                       .
                                                       .
                                                       .
tool.add_stage('file_upload', file_upload)
tool.add_stage('add_county', add_county)
tool.run()
```

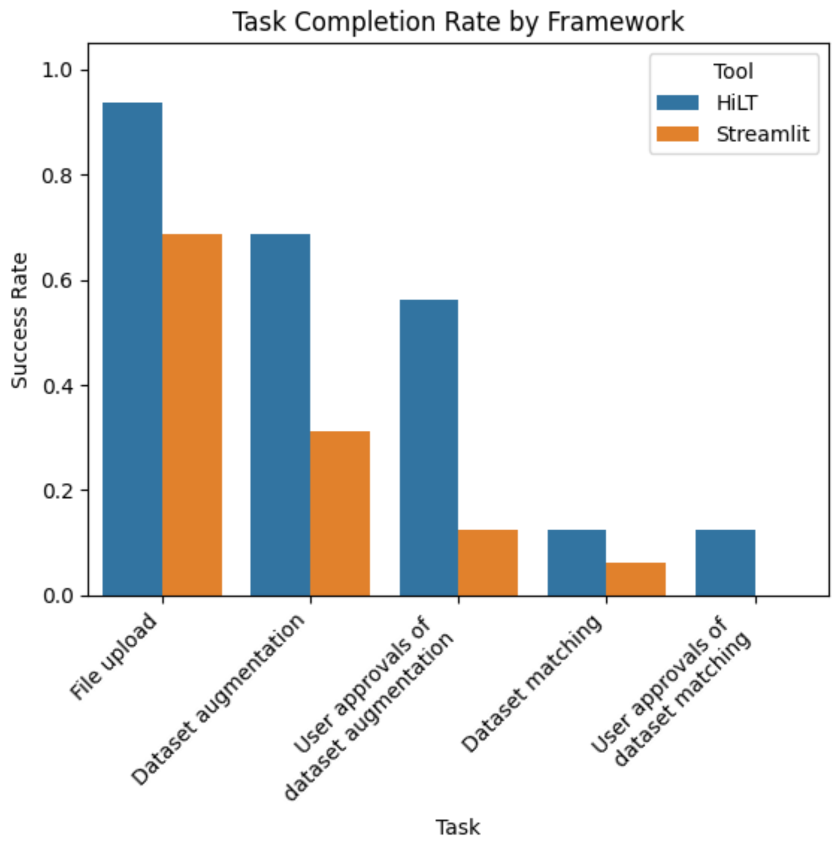
```
def coords_to_county(lat: str, lon: str):
    import json
    with open('counties.json') as f:
        counties = json.load(f)
    return counties[f'{lat}, {lon}']

tool = hilt.Tool('Wildfires')

def file_upload():
    file_path = hilt.FileUploadComponent(expected_ext = "csv", label = "Input a CSV file", replace_existing=True)
    name_input = hilt.UserInputComponent(str, "Name your CSV file: ")
    if tool.user_input_received():
        df = pd.read_csv(file_path.value)
        tool.tables[name_input.value] = df
        hilt.results.show_results((file_path.value, "Uploaded file path: ”))

def add_county():
    lat_selector = hilt.ColumnSelectorComponent("Choose your latitude column:")
    lon_selector = hilt.ColumnSelectorComponent("Choose your longitude column:")
    if tool.user_input_received():
        lat_col = lat_selector.column_names[0]
        lon_col = lon_selector.column_names[0]
        df = tool.tables[lat_selector.table_name]
        table_name = lat_selector.table_name
        df['COUNTY'] = df.apply(lambda row: coords_to_county(row[lat_col], row[lon_col]), axis=1)
        tool.tables[table_name] = df
        hilt.approvals.get_user_approvals()
        hilt.results.show_results((tool.tables[table_name], "Added counties to table: ”))
                                                        .
                                                        .
                                                        .
tool.add_stage('file_upload', file_upload)
tool.add_stage('add_county', add_county)
tool.run()
```

```
def coords_to_county(lat: str, lon: str):
    import json
    with open('counties.json') as f:
        counties = json.load(f)
    return counties[f'{lat}, {lon}']

tool = hilt.Tool('Wildfires')

def file_upload():
    file_path = hilt.FileUploadComponent(expected_ext = "csv", label = "Input a CSV file", replace_existing=True)
    name_input = hilt.UserInputComponent(str, "Name your CSV file: ")
    if tool.user_input_received():
        df = pd.read_csv(file_path.value)
        tool.tables[name_input.value] = df
        hilt.results.show_results((file_path.value, "Uploaded file path: ”))

def add_county():
    lat_selector = hilt.ColumnSelectorComponent("Choose your latitude column:")
    lon_selector = hilt.ColumnSelectorComponent("Choose your longitude column:")
    if tool.user_input_received():
        lat_col = lat_selector.column_names[0]
        lon_col = lon_selector.column_names[0]
        df = tool.tables[lat_selector.table_name]
        table_name = lat_selector.table_name
        df['COUNTY'] = df.apply(lambda row: coords_to_county(row[lat_col], row[lon_col]), axis=1)
        tool.tables[table_name] = df
        hilt.approvals.get_user_approvals()
        hilt.results.show_results((tool.tables[table_name], "Added counties to table: ”))
                                                            .
                                                            .
                                                            .
tool.add_stage('file_upload', file_upload)
tool.add_stage('add_county', add_county)
tool.run()
```

Outline

- Motivation
- Our intervention: HiLT
- Tour through a HiLT program
- **Evaluative user study results**

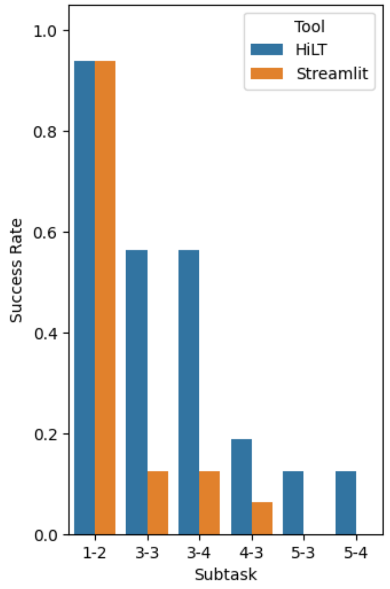
Task Completion Rate



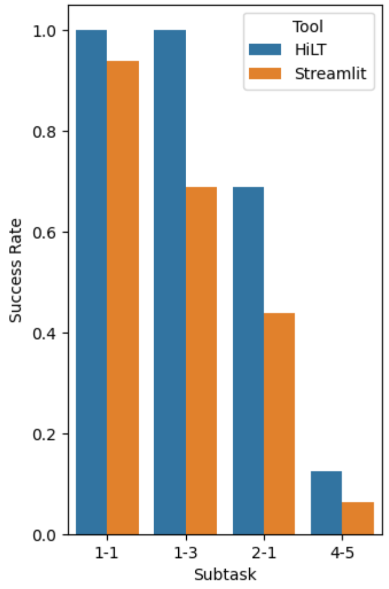
Subtask Completion Rate

Grouped By Design Goals 1–3

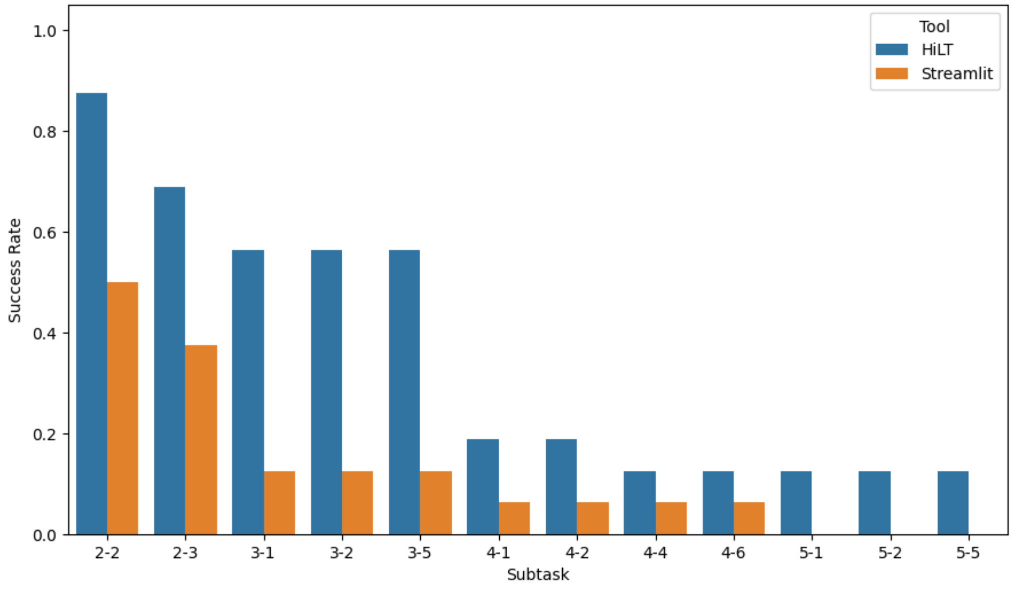
Guiding the User



Importing Data

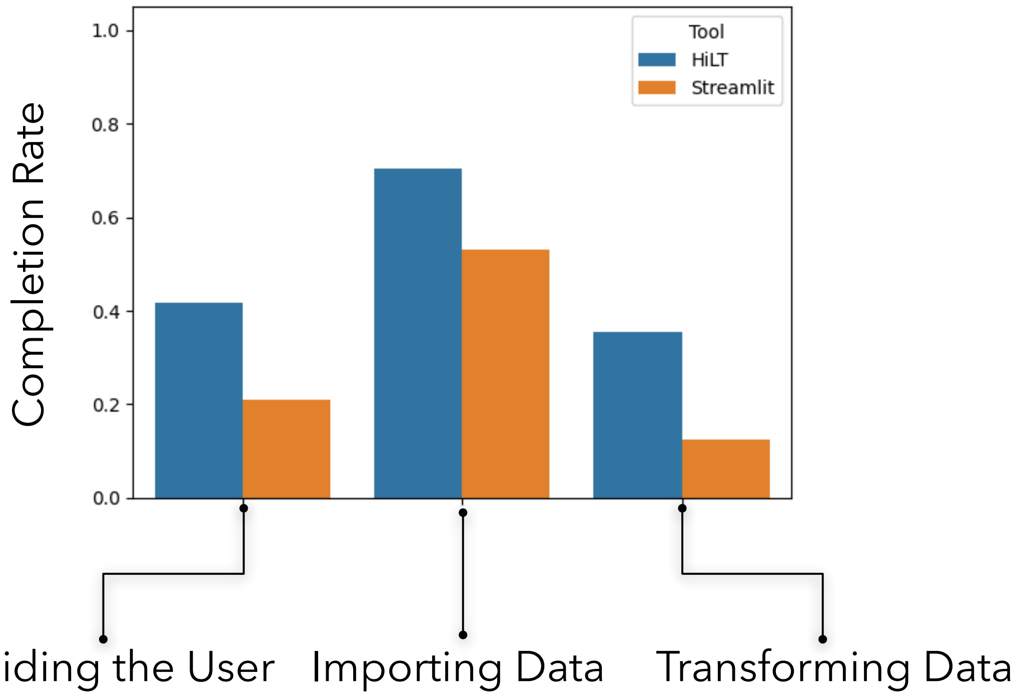


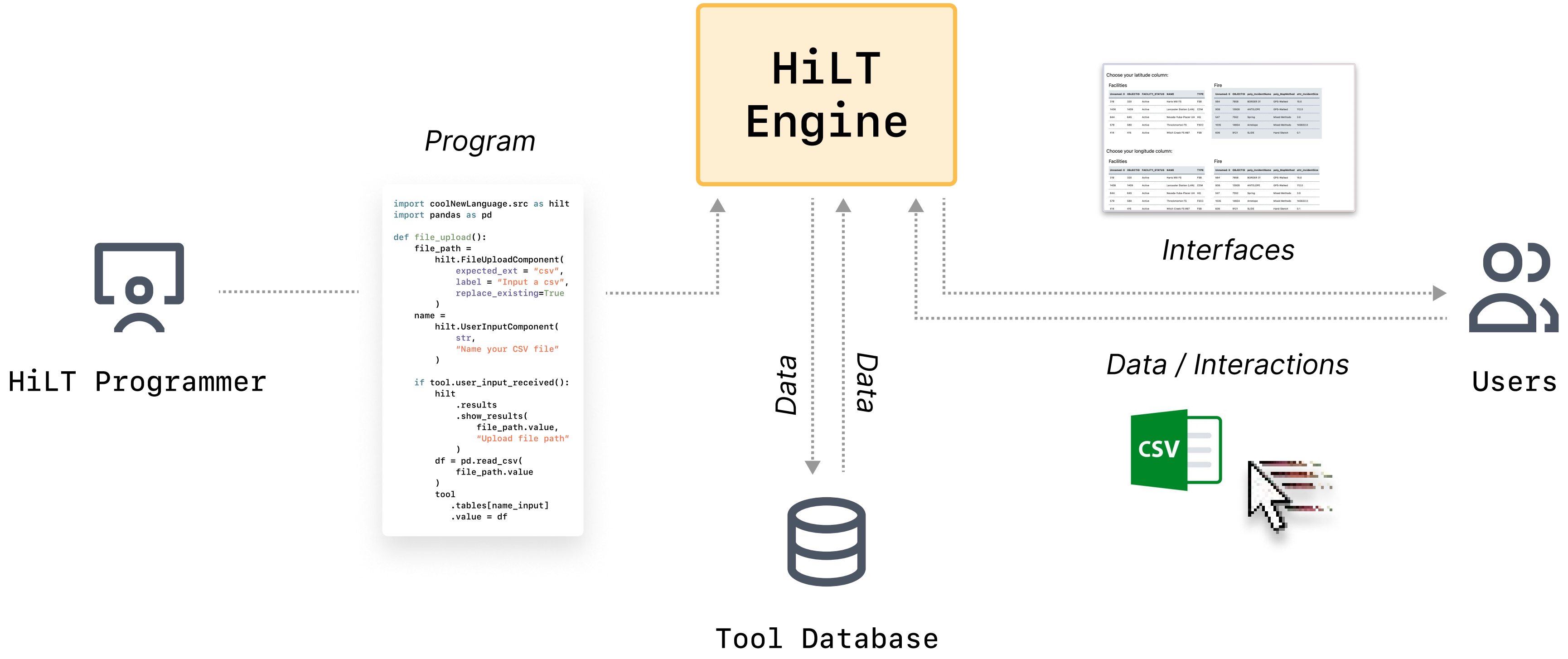
Transforming Data



Subtask Completion Rate

Grouped By Design Goals 1-3





```
import coolNewLanguage.src as hilt
import pandas as pd

def file_upload():
    file_path =
        hilt.FileUploadComponent(
            expected_ext = "csv",
            label = "Input a csv",
            replace_existing=True
        )
    name =
        hilt.UserInputComponent(
            str,
            "Name your CSV file"
        )

    if tool.user_input_received():
        hilt
            .results
            .show_results(
                file_path.value,
                "Upload file path"
            )
        df = pd.read_csv(
            file_path.value
        )
        tool
           .tables[name_input]
           .value = df
```

Choose your latitude column:

Facilities

Unnamed: 0	OBJECTID	FACILITY_STATUS	NAME	TYPE
319	320	Active	Harts Mill FS	FSB
1408	1409	Active	Lancaster Station (LAN)	COM
644	645	Active	Nevada-Yuba-Placer UH	HQ
579	580	Active	Throckmorton FS	FSCC
414	415	Active	Witch Creek FS #87	FSB

Fire

Unnamed: 0	OBJECTID	poly_IncidentName	poly_MapMethod	attr_IncidentSize
564	7858	BORDER 31	GPS-Walked	15.0
938	13926	ANTELOPE	GPS-Walked	112.0
547	7552	Spring	Mixed Methods	3.0
1035	14654	Antelope	Mixed Methods	145632.0
636	9121	SLIDE	Hand Sketch	0.1

Choose your longitude column:

Facilities

Unnamed: 0	OBJECTID	FACILITY_STATUS	NAME	TYPE
319	320	Active	Harts Mill FS	FSB
1408	1409	Active	Lancaster Station (LAN)	COM
644	645	Active	Nevada-Yuba-Placer UH	HQ
579	580	Active	Throckmorton FS	FSCC
414	415	Active	Witch Creek FS #87	FSB

Fire

Unnamed: 0	OBJECTID	poly_IncidentName	poly_MapMethod	attr_IncidentSize
564	7858	BORDER 31	GPS-Walked	15.0
938	13926	ANTELOPE	GPS-Walked	112.0
547	7552	Spring	Mixed Methods	3.0
1035	14654	Antelope	Mixed Methods	145632.0
636	9121	SLIDE	Hand Sketch	0.1

Email: sorakanosue@berkeley.edu