Advanced Data Management (CSCI 680/490)

Data Transformation

Dr. David Koop

Comma-separated values (CSV) Format

- Comma is a field separator, newlines denote records
  - ```
    a,b,c,d,message
    1,2,3,4,hello
    5,6,7,8,world
    9,10,11,12,foo
    ```
- May have a header (`a,b,c,d,message`), but not required
- No type information: we do not know what the columns are (numbers, strings, floating point, etc.)
  - Default: just keep everything as a string
  - Type inference: Figure out the type to make each column based on values
- What about commas in a value? → double quotes
    ```

Reading/Writing CSV with pandas

- Read: `df = pd.read_csv(<path>)`
- Write: `df.to_csv(<path>)`
- Parameters:
  - `sep` (or `delimiter`): the delimiter (`','`, `' '`, `'\t'`, `'\s+'`)
  - `header`: if `None`, no header
  - `names`: list of header names (e.g. if the file has no header)
  - `skiprows`: number of list of lines to skip

Reading/Writing CSV with DuckDB

- Importing:
  - `read_csv` method with parameters for delimter, header, etc.
  - `read_csv_auto` automatically **infer** these parameters
  - `CREATE TABLE ontime AS SELECT * FROM read_csv_auto('flights.csv');`
- Exporting:
  - Use the `COPY` function
  - `COPY tbl TO 'output.csv' (HEADER, DELIMITER ',');`

JavaScript Object Notation (JSON)

- A format for web data
- Looks very similar to python dictionaries and lists
- Example:

```
- {"name": "Wes",
   "places_lived": ["United States", "Spain", "Germany"],
   "pet": null,
   "siblings": [{"name": "Scott", "age": 25, "pet": "Zuko"},
                {"name": "Katie", "age": 33, "pet": "Cisco"}] }
```

- Only contains literals (no variables) but allows null
- Values: strings, arrays, dictionaries, numbers, booleans, or null
  - Dictionary keys must be strings
  - Quotation marks help differentiate string or numeric values

Parquet

- "Open source, column-oriented data file format designed for efficient data storage and retrieval" [parquet.apache.org]
- Available in multiple languages including python
- Binary format
- Column-oriented: can read a column at a time (e.g. from the cloud)
- Self-describing (schema can be embedded)
- Supports compression

Dataset	Columns	Size on Amazon S3	Data scanned	Cost (1TB = $5)
Data stored as CSV file	4	4TB	4TB	$20
Data stored as GZIP CSV file	4	1TB	1TB	$5
Data stored as Parquet file	4	1TB	0.25TB	$1.25

[T. Spicer]

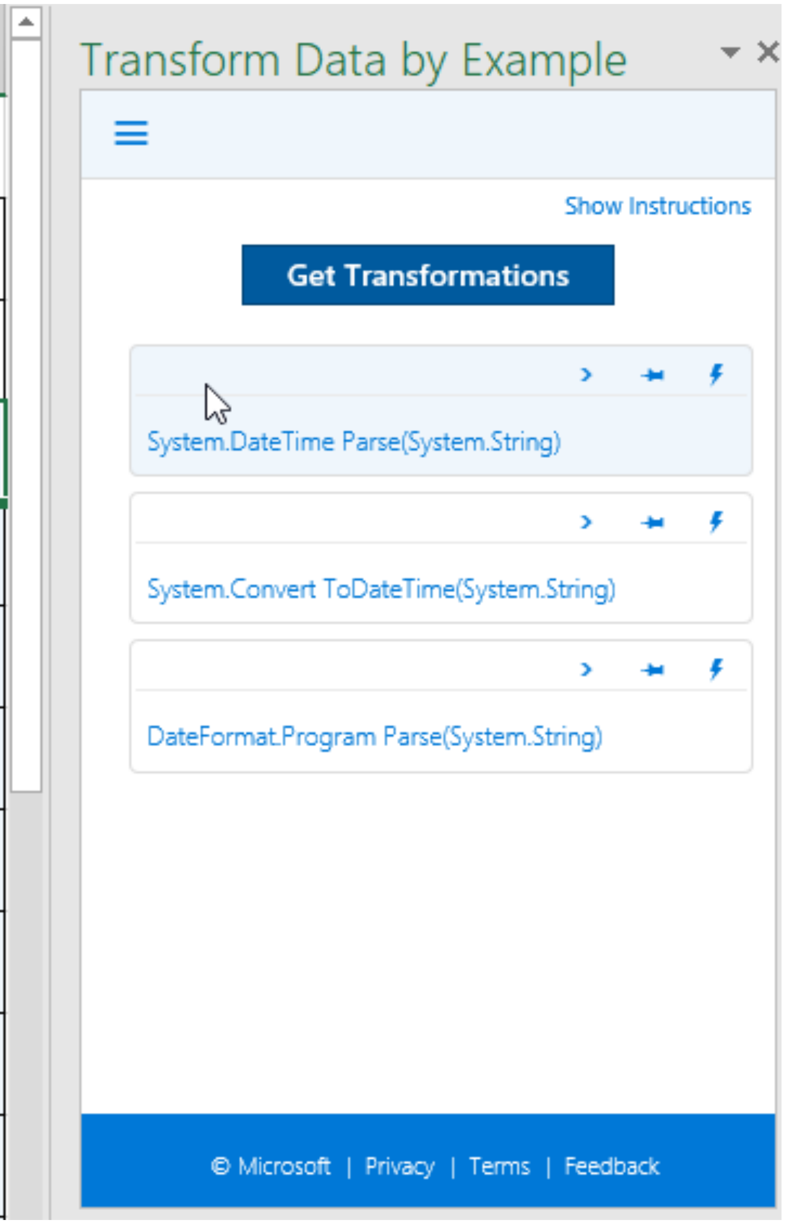
Parquet Support

- Pandas:
  - Install pyarrow
  - `df = pd.read_parquet('input.parquet')`
  - `df.to_parquet('output.parquet')`
- DuckDB
  - `CREATE TABLE new_tbl AS SELECT * FROM read_parquet('input.parquet');`
  - `COPY tbl TO 'output.parquet' (FORMAT PARQUET);`

TDE: Transform Data by Example

C	D
Transaction Date	**output**
Wed, 12 Jan 2011	2011-01-12-Wednesday
Thu, 15 Sep 2011	2011-09-15-Thursday
Mon, 17 Sep 2012	
2010-Nov-30 11:10:41	
2011-Jan-11 02:27:21	
2011-Jan-12	
2010-Dec-24	
9/22/2011	
7/11/2012	
2/12/2012	

C	D
Transaction Date	**output**
Wed, 12 Jan 2011	2011-01-12-Wednesday
Thu, 15 Sep 2011	2011-09-15-Thursday
Mon, 17 Sep 2012	2012-09-17-Monday
2010-Nov-30 11:10:41	2010-11-30-Tuesday
2011-Jan-11 02:27:21	2011-01-11-Tuesday
2011-Jan-12	2011-01-12-Wednesday
2010-Dec-24	2010-12-24-Friday
9/22/2011	2011-09-22-Thursday
7/11/2012	2012-07-11-Wednesday
2/12/2012	2012-02-12-Sunday



[Y. He et al., 2018]

Transform by Pattern (TBP)

- Focus on non-technical users
- More general than Transform by Example
- No need for paired examples
- Use Cases:
  - Auto-Unify: Unify data in different formats
  - Auto-Repair: Fix data quality issues
- Example (Auto-Unify):
  - P_S = <letter>{3}. <digit>{2}, <digit>{4}
  - P_T = <digit>{4}-<digit>{2}-<digit>{2}

S-timestamp	S-phone	S-coordinates
2019-12-23	(425) 882-8080	(38°57'N, 95°15'W)
2019-12-24	(425) 882-8080	(38°61'N, 95°21'W)
2019-12-23	(206) 876-1800	(39°19'N, 95°18'W)
2019-12-24	(206) 876-1800	(39°26'N, 95°23'W)
2019-12-23	(206) 903-8010	(39°42'N, 96°38'W)

R-timestamp	R-phone	R-coordinates
Nov. 16 2019	650-853-1300	N37°31' W122°14'
Nov. 17 2019	650-853-1300	N37°18' W122°19'
Nov. 16 2019	425-421-1225	N37°48' W122°17'
Nov. 17 2019	425-421-1225	N37°60' W123°08'
Nov. 16 2019	650-253-0827	N37°01' W123°72'

[Jin et al.]

TBP Use Cases

- Auto-Unify

S-timestamp	S-phone	S-coordinates
2019-12-23	(425) 882-8080	(38°57'N, 95°15'W)
2019-12-24	(425) 882-8080	(38°61'N, 95°21'W)
2019-12-23	(206) 876-1800	(39°19'N, 95°18'W)
2019-12-24	(206) 876-1800	(39°26'N, 95°23'W)
2019-12-23	(206) 903-8010	(39°42'N, 96°38'W)

R-timestamp	R-phone	R-coordinates
Nov. 16 2019	650-853-1300	N37°31′ W122°14′
Nov. 17 2019	650-853-1300	N37°18′ W122°19′
Nov. 16 2019	425-421-1225	N37°48′ W122°17′
Nov. 17 2019	425-421-1225	N37°60′ W123°08′
Nov. 16 2019	650-253-0827	N37°01′ W123°72′

- Auto-Repair

Date	Opponents
January 12, 1997	Venezuela
February 12, 1997	Peru
April 2, 1997	Colombia
1997-06-04	United States
1997-06-11	Chile
1997-06-14	Ecuador

(a) EN-Wiki: Dates

Year	Artist	Issue Price (BU)
1989	John Mardon	$16.25
1990	D.J. Craig	$16.75
1991	D.J. Craig	$16.75
1992	Karsten Smith	17.50
1993	Stewart Sherwood	$17.50
1994	Ian D. Sparkes	$17.95

(b) EN-Wiki: Currency values

Women's winner	Time
Anikó Kálovics	2:31:24
Lenah Cheruiyot	2:27:02
Lenah Cheruiyot	2:33.44
Emily Kimuria	2:28.42
Jane Ekimat	2:32.08

(c) EN-wiki:time

#	Original air date[1]
12	March 23, 2008
13	March 30, 2008
14	April 6, 2008
15	13 April 2008
16	20 April 2008

(d) EN-Wiki: Date

[Jin et al.]

TBP Programs and Triples

Table 1: An example repository of TBP programs (P_s, P_t, T), where each line is a TBP program. The first three programs can be used to auto-unify the two tables shown in Figure 2.

TBP-id	Source-pattern (P_s)	Target-pattern (P_t)	(T)
TBP-1	<letter>{3}. <digit>{2}, <digit>{4}	<digit>{4}-<digit>{2}-<digit>{2}	…
TBP-2	(<digit>{3}) <digit>{3}-<digit>{4}	<letter>{3}-<digit>{3}-<digit>{4}	…
TBP-3	(<digit>+°<num>'<letter>{1}, <digit>+°<num>'<letter>{1})	<letter>{1}<digit>+°<num>' <letter>{1}<digit>+°<num>'	…
…	…	…	…
TBP-7	<digit>{4}/<digit>{2}/<digit>{2}	<letter>{3} <digit>{2}	…
TBP-8	<num> kg	<num> lb	…
TBP-9	<num> lb	<num> lb <num> oz	…
…	…	…	…
TBP-15	<num> kg	<num>公斤	…
TBP-16	<letter>+ de <digit>{4}	<digit>{4}	…
…	…	…	…

CCT-id	Input-column (C)	Output-column (C')	Program (T)
CCT-1	(C_1) "Born" = {"02/22/1732", "10/30/1735", … }	(C'_1) "Date of birth" = {"February 22, 1732", … }	Listing 1
CCT-2	(C_2) "Date of birth" = {"February 22, 1732", … }	(C'_2) "Born" = {"02/22/1732", "10/30/1735", … }	…
CCT-3	(C_3) "Died" = {"02/14/1799", "07/04/1826", … }	(C'_3) "Date of birth" = {"February 22, 1732", … }	…
CCT-4	(C_4) "Date" = {"11/01/2019", "12/01/2019", … }	(C'_4) "Date-2" = {"November 01, 2019", … }	Listing 1
…	…	…	…
CCT-9	(C_9) "Name" = {"Washington, George", "Adam, John", … }	(C'_9) "Date of birth" = {"February 22, 1732", … }	$\emptyset$
…	…	…	…

[Jin et al.]

Learning TBP Programs

- User Logs
  - Similar to Search Engines
  - (Privacy Issues)
- Tables
  - Find common tables whose rows can be linked
  - Link Wikipedia tables across languages
  - Obtain different data formats and abbreviations that can be used as patterns

[Jin et al.]

TBP Learning from Tables

T_1

Name	#	Born	Died
Washington, George	USA President (1)	02/22/1732	12/14/1799
Adams, John	USA President (2), VP (1)	10/30/1735	07/04/1826
Jefferson, Thomas	USA President (3), VP (2)	04/13/1743	07/04/1826
Madison, James	USA President (4)	03/16/1751	06/28/1836
Monroe, James	USA President (5)	04/28/1758	07/04/1851

T_2

Date of birth	President	Birthplace	State† of birth
February 22, 1732	George Washington	Westmoreland County	Virginia†
October 30, 1735	John Adams	Braintree	Massachusetts†

T_3

30.	**George Washington**	–	57y, 10d	22.02.1732	14.12.1799
31.	**John Quincy Adams**	Nat-Rep	57y, 7m, 20d	11.07.1767	23.02.1848
32.	**Thomas Jefferson**	Dem-Rep	57y, 10m, 18d	13.04.1743	04.07.1826
33.	**James Madison**	Dem-Rep	57y, 11m, 15d	16.03.1751	28.06.1836
34.	**James Monroe**	Dem-Rep	58y, 10m, 3d	28.04.1758	04.07.1831

T_4

1.	George Washington	Virginia	Feb. 22, 1732	Dec. 14, 1797
3.	Thomas Jefferson	Virginia	Apr. 13, 1743	July 4, 1826
4.	James Madison	Virginia	Mar. 16, 1751	June 28, 1836
6.	John Quincy Adams	Massachusetts	July 11, 1767	Feb. 23, 1848

T_5

	Name and (party)[1]	Term	State of birth	Born	Died	Religion[2]	Age at inaug.	Age at death
1.	Washington (F)[3]	1789–1797	Va.	2/22/1732	12/14/1799	Episcopalian	57	67
2.	J. Adams (F)	1797–1801	Mass.	10/30/1735	7/4/1826	Unitarian	61	90

T_6

PRESIDENT	BIRTH DATE	BIRTH PLACE	DEATH DATE	LOCATION OF DEATH
George Washington	Feb 22, 1732	Westmoreland Co., Va.	Dec 14, 1799	Mount Vernon, Va.
John Adams	Oct 30, 1735	Quincy, Mass.	July 4, 1826	Quincy, Mass.

[Jin et al.]

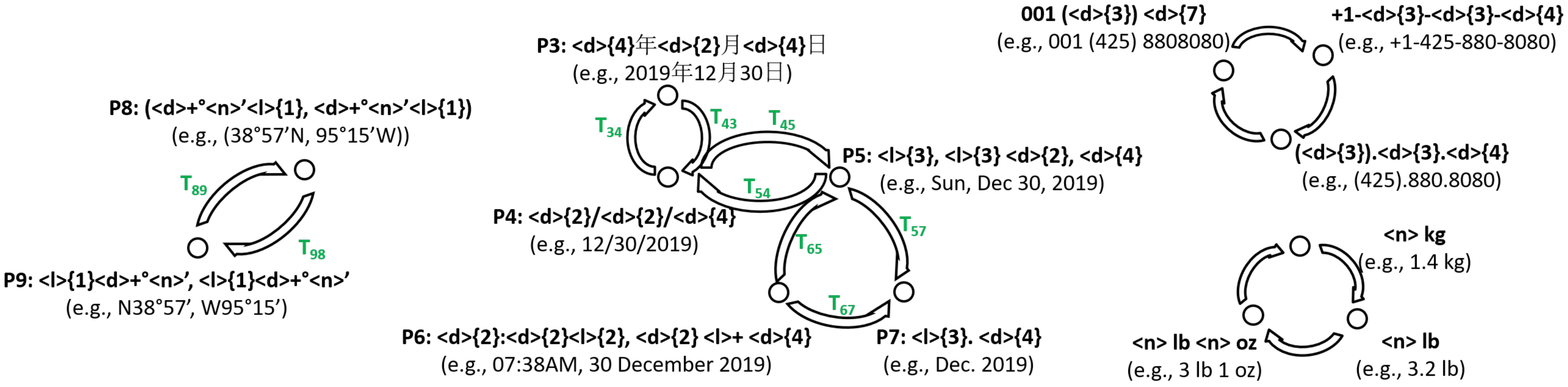
Generating Patterns

- Generate potential regex patterns
- Want more general patterns
- (`<digits>/<digits>/<digits>` vs. `<digits>/<digits>/17<digits>`)
- Can be too general: `<num><symbol><num><symbol><num>`
- Want high "coverage" and high "accuracy"
-

[Jin et al.]

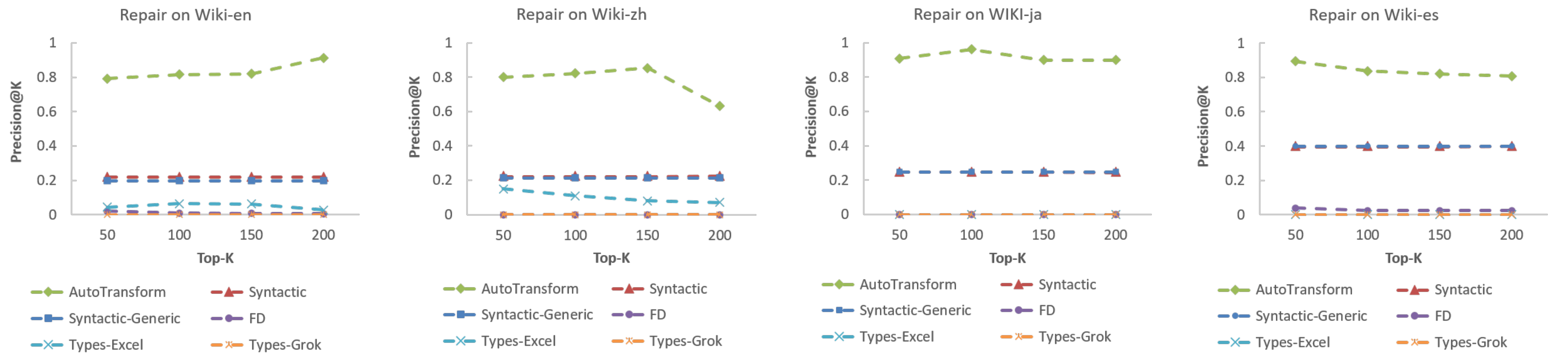
Graph Pattern Relationships

- Lossless inverses: can go back and forth
- Triangular equivalent programs: applying one transformation on a column matches the output of apply two other transformations in sequence



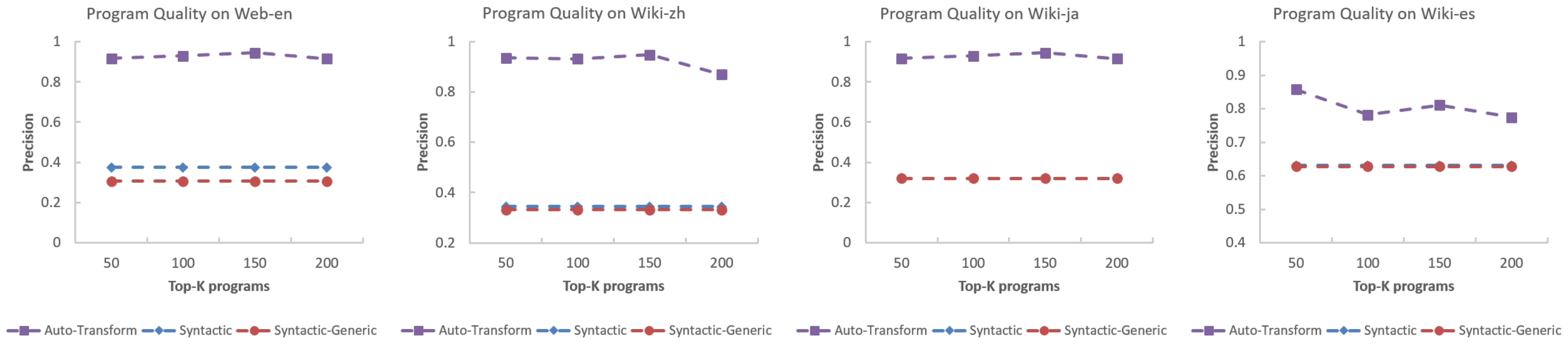
[Jin et al.]

Experiment Results



(a) Wiki-en (b) Wiki-zh (c) Wiki-ja (d) Wiki-es

Figure 10: Quality of repairs on Wiki-en, Wiki-zh, Wiki-ja, Wiki-es, using TBP programs learned from corresponding corpus.



(a) Web-en (b) Wiki-zh (c) Wiki-ja (d) Wiki-es

Figure 11: Quality of TBP programs produced on Web-en, Wiki-zh, Wiki-ja, Wiki-es, respectively.

[Jin et al.]

Questions/Discussion?

Questions/Discussion?

- Strong focus on dates in examples
- Does this help the analyst who has specific types of data and formats?
- How does this relate to programmatic means of wrangling?

Test 1

- Monday, Feb. 27
- In-class, 9:30-10:45am
- Format:
  - Multiple Choice
  - Free Response
- Information will be posted online

Data Transformation

Pandas Transformations

- Split: str.split
- Fold/Unfold: stack/unstack
- Merge, join, and concatenate documentation:
  - https://pandas.pydata.org/pandas-docs/stable/merging.html

Tidy Data

- Dataset contain values: quantitative and categorical/qualitative
- Value is either:
  - **variable**: all values that measure the same underlying attribute
  - **observation**: all values measured on the same unit across attributes

[H. Wickham, 2014]

Three Ways to Present the Same Data

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Initial Data

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Transpose

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Tidy Data

[H. Wickham, 2014]

Tidy Data Principles

- **Tidy Data**: Codd's 3rd Normal Form (Databases)
  1. Each variable forms a column
  2. Each observation forms a row
  3. Each type of observational unit forms a table (DataFrame)
- Other structures are **messy data**

[H. Wickham, 2014]

Tidy Data

- Benefits:
  - Easy for analyst to extract variables
  - Works well for vectorized programming
- Organize variables by their role
  - Fixed variables: describe experimental design, known in advance
  - Measured variables: what is measured in study
- Variables also known as dimensions and measures

[H. Wickham, 2014]

Messy Dataset Problems

- Column headers are values, not variable names
- Multiple variables are stored in one column
- Variables are stored in both rows and columns
- Multiple types of observational units are stored in the same table
- A single observational unit is stored in multiple tables

Problem: Column Headers are Values

Income and Religion, Pew Forum

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

[H. Wickham, 2014]

Problem: Column Headers are Values

Income and Religion, Pew Forum

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Variables: religion, income, frequency

[H. Wickham, 2014]

Solution: Melt Data

- Turn columns into rows
- One or more columns become rows under a new column (`column`)
- Values become a new column (`value`)
- After melt, data is **molten**
- AKA pivot_longer
- **Inverse** of pivot

row	a	b	c
A	1	4	7
B	2	5	8
C	3	6	9

(a) Raw data

row	column	value
A	a	1
B	a	2
C	a	3
A	b	4
B	b	5
C	b	6
A	c	7
B	c	8
C	c	9

(b) Molten data

[H. Wickham, 2014]

Solution: Molten Data

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Original

religion	income	freq
Agnostic	<$10k	27
Agnostic	$10-20k	34
Agnostic	$20-30k	60
Agnostic	$30-40k	81
Agnostic	$40-50k	76
Agnostic	$50-75k	137
Agnostic	$75-100k	122
Agnostic	$100-150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Molten (first 10 rows)

[H. Wickham, 2014]

Melting: Billboard Top Hits

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

Table 7: The first eight Billboard top hits for 2000. Other columns not shown are `wk4`, `wk5`, ..., `wk75`.

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

[Wickham, 2014]

Melting

- Pandas also has a melt function:

```
In [41]: cheese = pd.DataFrame({'first' : ['John', 'Mary'],
   ....:                        'last' : ['Doe', 'Bo'],
   ....:                        'height' : [5.5, 6.0],
   ....:                        'weight' : [130, 150]})
   ....:

In [42]: cheese
Out[42]:
  first  height last  weight
0  John     5.5  Doe     130
1  Mary     6.0   Bo     150

In [43]: cheese.melt(id_vars=['first', 'last'])
Out[43]:
  first last variable  value
0  John  Doe   height    5.5
1  Mary   Bo   height    6.0
2  John  Doe   weight  130.0
3  Mary   Bo   weight  150.0

In [44]: cheese.melt(id_vars=['first', 'last'], var_name='quantity')
Out[44]:
  first last quantity  value
0  John  Doe   height    5.5
1  Mary   Bo   height    6.0
2  John  Doe   weight  130.0
3  Mary   Bo   weight  150.0
```

Problem: Multiple variables stored in one column

Tuberculosis Data, World Health Organization

country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
AD	2000	0	0	1	0	0	0	0	—	—
AE	2000	2	4	4	6	5	12	10	—	3
AF	2000	52	228	183	149	129	94	80	—	93
AG	2000	0	0	0	0	0	0	1	—	1
AL	2000	2	19	21	14	24	19	16	—	3
AM	2000	2	152	130	131	63	26	21	—	1
AN	2000	0	0	1	2	0	0	0	—	0
AO	2000	186	999	1003	912	482	312	194	—	247
AR	2000	97	278	594	402	419	368	330	—	121
AS	2000	—	—	—	—	1	1	—	—	—

[H. Wickham, 2014]

Problem: Multiple variables stored in one column

Tuberculosis Data, World Health Organization

country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
AD	2000	0	0	1	0	0	0	0	—	—
AE	2000	2	4	4	6	5	12	10	—	3
AF	2000	52	228	183	149	129	94	80	—	93
AG	2000	0	0	0	0	0	0	1	—	1
AL	2000	2	19	21	14	24	19	16	—	3
AM	2000	2	152	130	131	63	26	21	—	1
AN	2000	0	0	1	2	0	0	0	—	0
AO	2000	186	999	1003	912	482	312	194	—	247
AR	2000	97	278	594	402	419	368	330	—	121
AS	2000	—	—	—	—	1	1	—	—	—

Two variables in columns: age and sex

[H. Wickham, 2014]

Solution: Melting + Splitting

country	year	column	cases
AD	2000	m014	0
AD	2000	m1524	0
AD	2000	m2534	1
AD	2000	m3544	0
AD	2000	m4554	0
AD	2000	m5564	0
AD	2000	m65	0
AE	2000	m014	2
AE	2000	m1524	4
AE	2000	m2534	4
AE	2000	m3544	6
AE	2000	m4554	5
AE	2000	m5564	12
AE	2000	m65	10
AE	2000	f014	3

(a) Molten data

country	year	sex	age	cases
AD	2000	m	0-14	0
AD	2000	m	15-24	0
AD	2000	m	25-34	1
AD	2000	m	35-44	0
AD	2000	m	45-54	0
AD	2000	m	55-64	0
AD	2000	m	65+	0
AE	2000	m	0-14	2
AE	2000	m	15-24	4
AE	2000	m	25-34	4
AE	2000	m	35-44	6
AE	2000	m	45-54	5
AE	2000	m	55-64	12
AE	2000	m	65+	10
AE	2000	f	0-14	3

(b) Tidy data

[H. Wickham, 2014]

Problem: Variables stored in both rows & columns

Mexico Weather, Global Historical Climatology Network

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

[H. Wickham, 2014]

Problem: Variables stored in both rows & columns

Mexico Weather, Global Historical Climatology Network

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

Variable in columns: day; Variable in rows: tmax/tmin

[H. Wickham, 2014]

Pivot

- Sometimes, we have data that is given in "long" format and we would like "wide" format (AKA pivot_wider)
- Long format: column names are data values…
- Wide format: more like spreadsheet format
- Example:

```
        date     item     value
0 1959-03-31  realgdp  2710.349
1 1959-03-31     infl     0.000
2 1959-03-31    unemp     5.800
3 1959-06-30  realgdp  2778.801
4 1959-06-30     infl     2.340
5 1959-06-30    unemp     5.100
6 1959-09-30  realgdp  2775.488
7 1959-09-30     infl     2.740
8 1959-09-30    unemp     5.300
9 1959-12-31  realgdp  2785.204
```

```
.pivot('date', 'item', 'value')
```

```
item        infl   realgdp  unemp
date
1959-03-31  0.00  2710.349    5.8
1959-06-30  2.34  2778.801    5.1
1959-09-30  2.74  2775.488    5.3
1959-12-31  0.27  2785.204    5.6
1960-03-31  2.31  2847.699    5.2
```

[W. McKinney, Python for Data Analysis]

Solution: Melting + Pivot

id	date	element	value
MX17004	2010-01-30	tmax	27.8
MX17004	2010-01-30	tmin	14.5
MX17004	2010-02-02	tmax	27.3
MX17004	2010-02-02	tmin	14.4
MX17004	2010-02-03	tmax	24.1
MX17004	2010-02-03	tmin	14.4
MX17004	2010-02-11	tmax	29.7
MX17004	2010-02-11	tmin	13.4
MX17004	2010-02-23	tmax	29.9
MX17004	2010-02-23	tmin	10.7

(a) Molten data

id	date	tmax	tmin
MX17004	2010-01-30	27.8	14.5
MX17004	2010-02-02	27.3	14.4
MX17004	2010-02-03	24.1	14.4
MX17004	2010-02-11	29.7	13.4
MX17004	2010-02-23	29.9	10.7
MX17004	2010-03-05	32.1	14.2
MX17004	2010-03-10	34.5	16.8
MX17004	2010-03-16	31.1	17.6
MX17004	2010-04-27	36.3	16.7
MX17004	2010-05-27	33.2	18.2

(b) Tidy data

[H. Wickham, 2014]

Getting Lost in Transformations

Bureau of I.A.	
Regional Director	Numbers
Niles C.	Tel: (800)645-8397
	Fax: (907)586-7252
Jean H.	Tel: (918)781-4600
	Fax: (918)781-4604
Frank K.	Tel: (615)564-6500
	Fax: (615)564-6701

Original Table

Split+Delete

Niles C.	Tel	(800)645-8397
	Fax	(907)586-7252
Jean H.	Tel	(918)781-4600
	Fax	(918)781-4604
Frank K.	Tel	(615)564-6500
	Fax	(615)564-6701

Intermediate Table

Unfold

	Tel	Fax
Niles C.	(800)645-8397	
		(615)564-6701
Jean H.	(918)781-4600	
Frank K.	(615)564-6500	

Problem Table

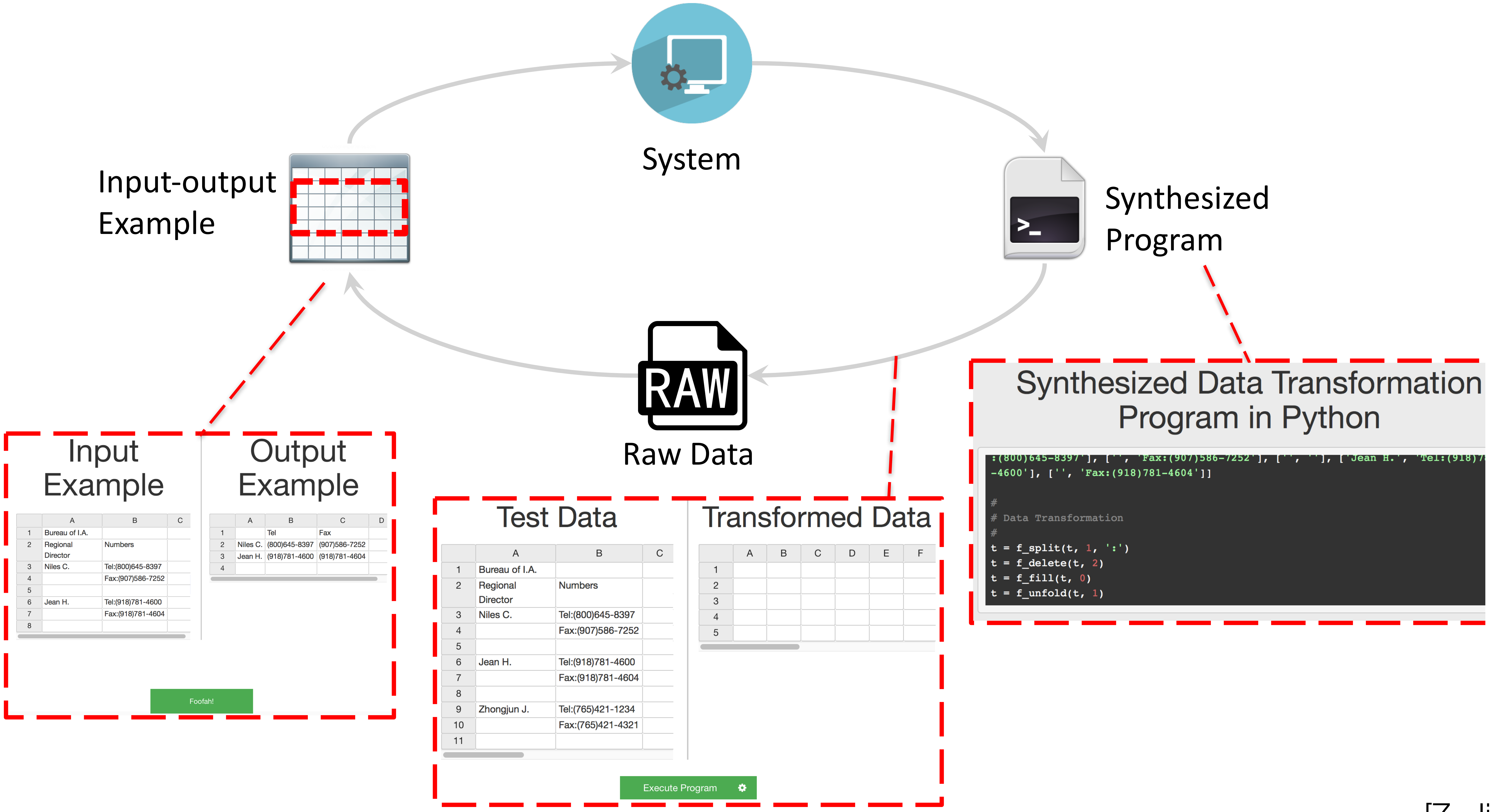
Fill+ Unfold

	Tel	Fax
Niles C.	(800)645-8397	(907)586-7252
Jean H.	(918)781-4600	(918)781-4604
Frank K.	(615)564-6500	(615)564-6701

Desired Solution

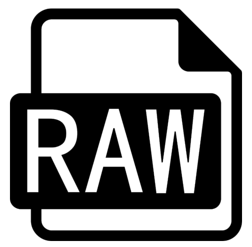
[Z. Jin et al., 2017]

Foofah Design: Programming by Example



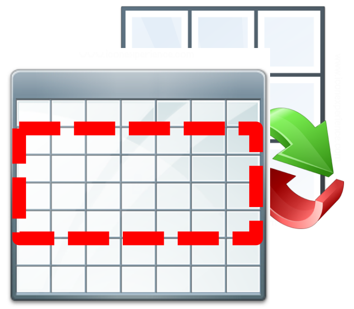
[Z. Jin et al., 2017]

Input, Output, and Transformations



Raw Data:

- A grid of values, i.e., spreadsheets
- "Somewhat" structured - must have some regular structure or is automatically generated.



User Input:

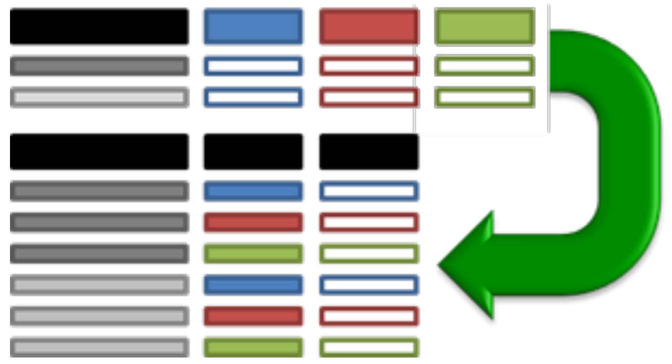
- Sample from raw data
- Transformed view of the sample

Program to synthesize:

- A loop-free Potter's Wheel [2] program

Transformations Targeted:

1. Layout transformation



2. String transformation

05-16-2017
05-17-2017
…

→

05/16/2017
05/17/2017
…

[Z. Jin et al., 2017]

Foofah Solution

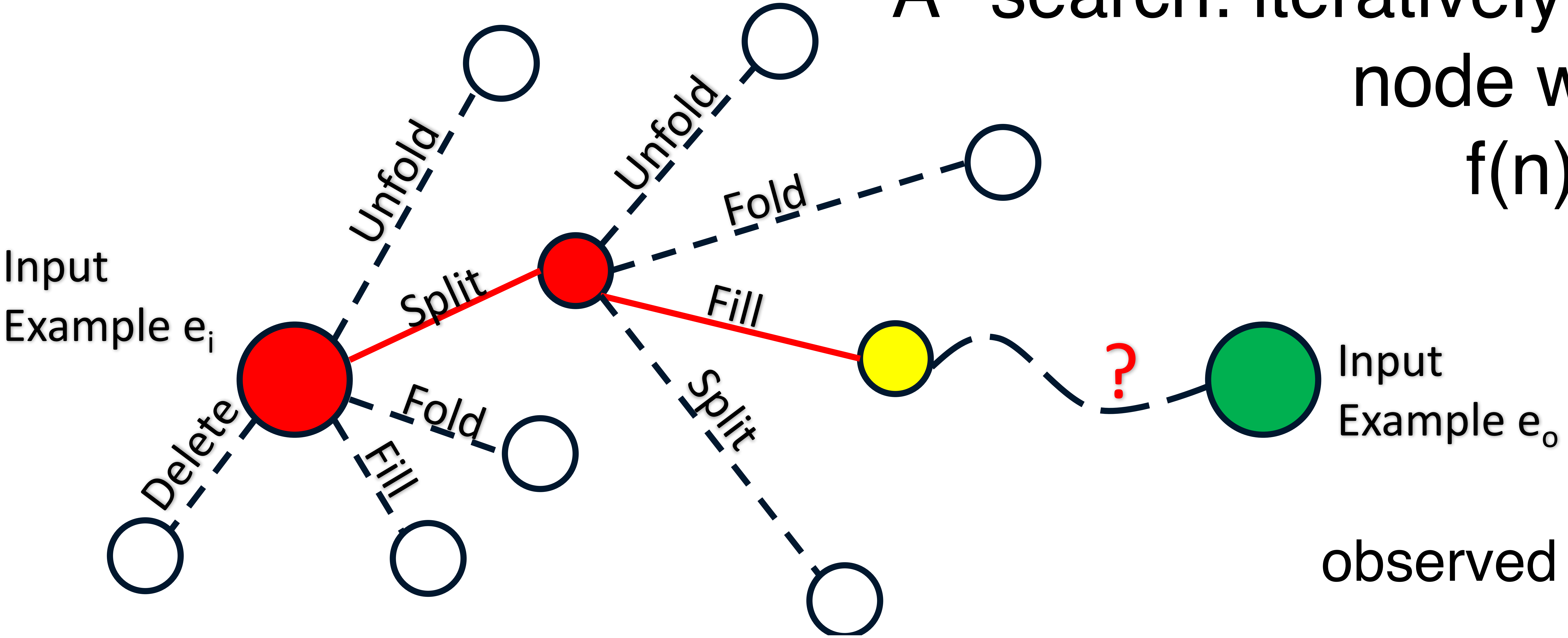
A search problem solved by A* algorithm

edges: operation

nodes: different views of the data

A* search: iteratively explore the node with min f(n)

$f(n) = g(n) + h(n)$



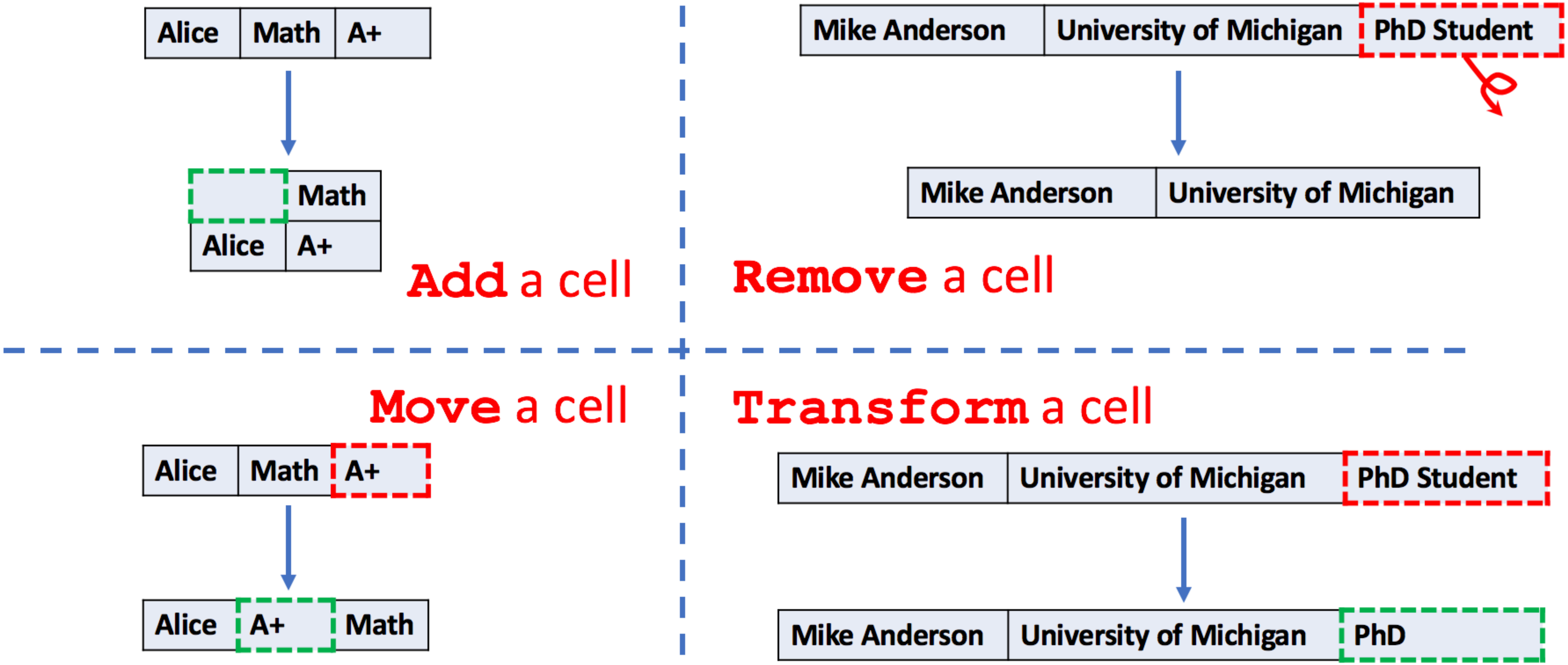
estimated distance

observed distance

[Z. Jin et al., 2017]

Need a Heuristic Function to Prune

Most transformations are composed of cell-based operations



[Z. Jin et al., 2017]

Table Edit Distance

- Akin to Graph Edit Distance
- Count the number of operations required to transform one table to another
- Use Add/Remove/Modify + Move

Table Edit Distance (TED) Definition:
The cost of transforming Table T_1 to Table T_2 using the cell-level operators Add/Remove/Move/Transform cell.

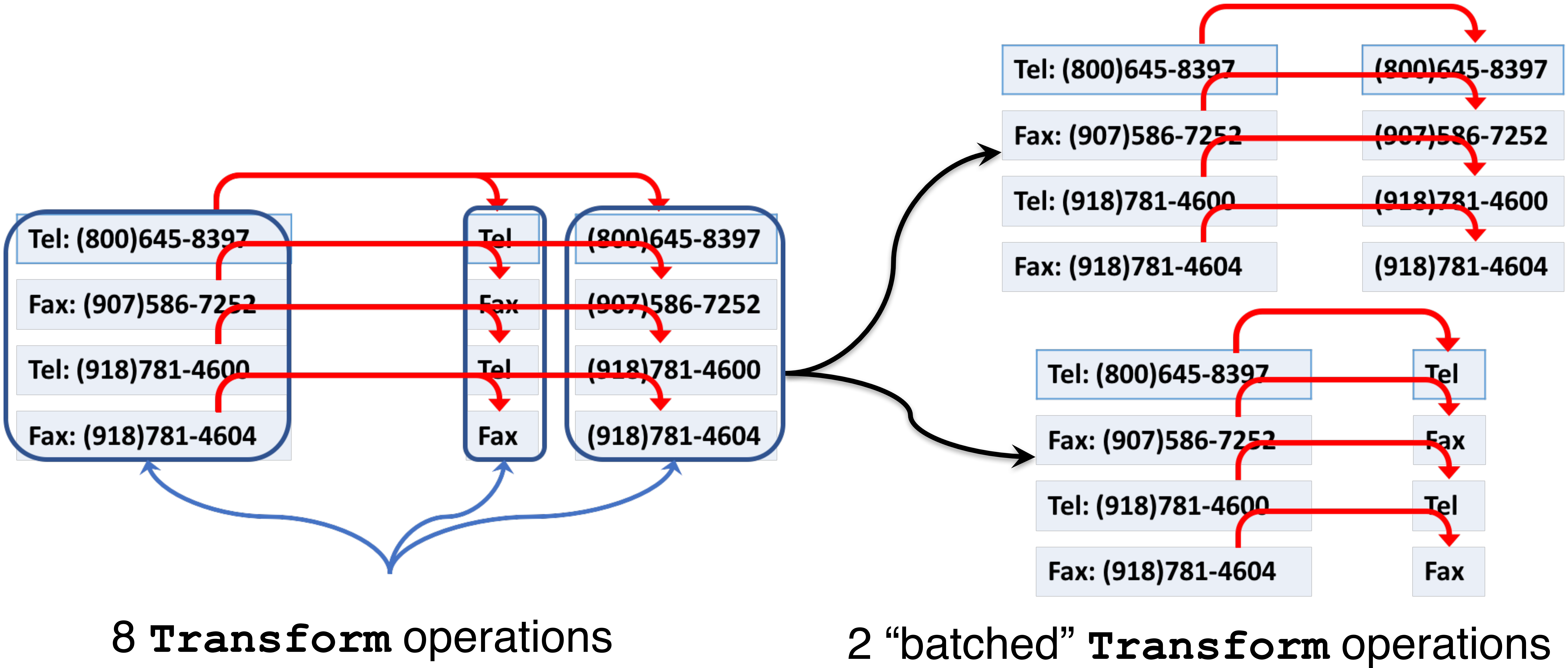
$$\mathrm{TED}(T_1, T_2) = \min_{(p_1, \ldots, p_k) \in P(T_1, T_2)} \sum_{i=0}^{k} cost(p_i)$$

- $P(T_1, T_2)$: Set of all "paths" transforming T_1 to T_2 using cell-level operators

[Z. Jin et al., 2017]

Table Edit Distance Batch

Batch the geometrically-adjacent cell-level operations of the same type



[Z. Jin et al., 2017]

Geometric Patterns Used to Batch

Pattern	Formulation (X is a table edit operator)	Related Operators
Horizontal to Horizontal	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i + 1), (x_j, y_j + 1)), \ldots\}$	Delete(Possibly)
Horizontal to Vertical	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i + 1), (x_j + 1, y_j)), \ldots\}$	Fold, Transpose
Vertical to Horizontal	$\{X((x_i, y_i), (x_j, y_j)), X((x_i + 1, y_i), (x_j, y_j + 1)), \ldots\}$	Unfold,Transpose
Vertical to Vertical	$\{X((x_i, y_i), (x_j, y_j)), X((x_i + 1, y_i), (x_j + 1, y_j)), \ldots\}$	Move, Copy, Merge, Split, Extract, Drop
One to Horizontal	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i), (x_j, y_j + 1)), \ldots\}$	Fold(Possibly), Fill(Possibly)
One to Vertical	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i), (x_j + 1, y_j)), \ldots\}$	Fold, Fill
Remove Horizontal	$\{X((x_i, y_i)), X((x_i, y_i + 1)), \ldots\}$	Delete
Remove Vertical	$\{X((x_i, y_i)), X((x_i + 1, y_i)), \ldots\}$	Drop, Unfold

[Z. Jin et al., 2017]

Other Pruning Rules

- Global:
  - Missing Alphanumerics: check that character maintained
  - No effect: meaningless operation
  - Introducing Novel Symbols: check that no new characters added
- Property-specific:
  - Generating Empty Columns
  - Null in Column

AutoSuggest

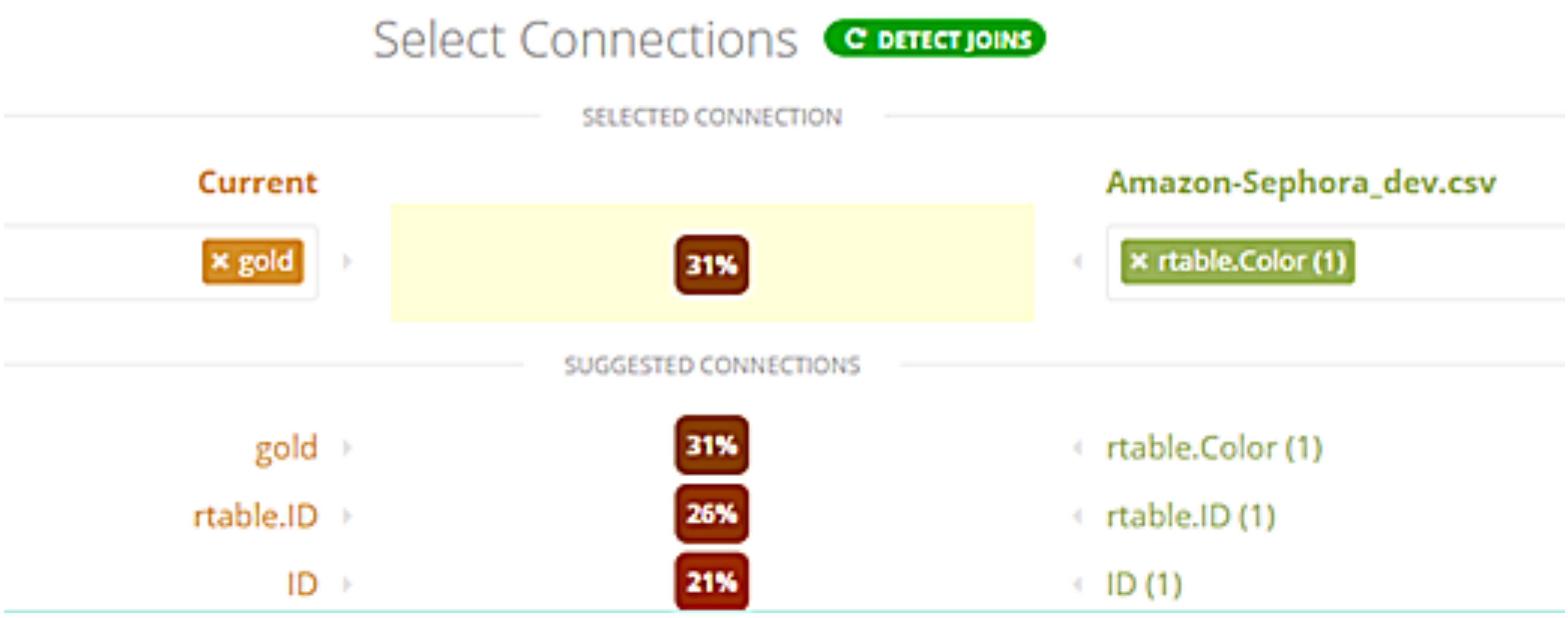
Comments/Questions?

Goal

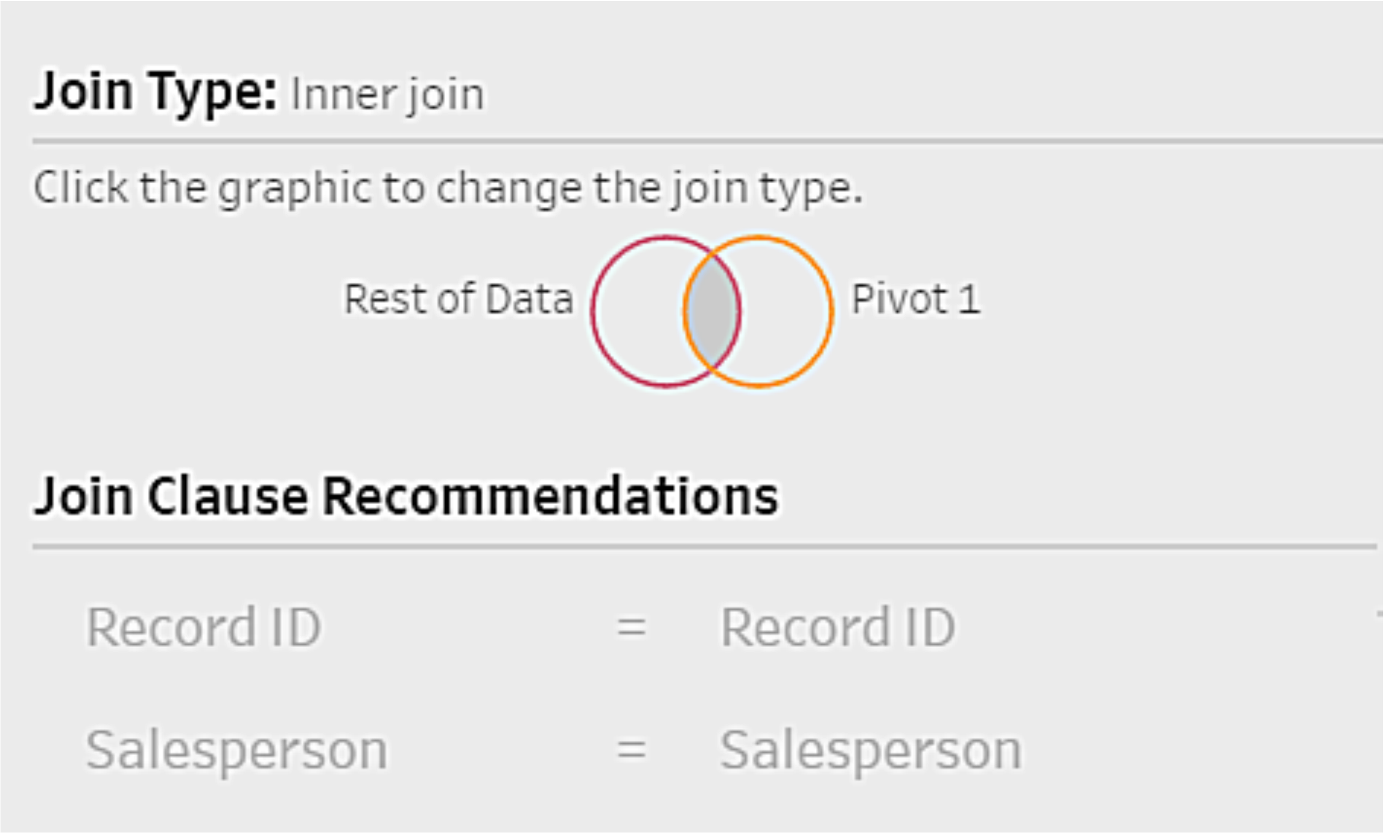
- Automate "Complex" Data Preparation steps
- Focus on frame transformations (not per-cell transformations)
- Learn from Jupyter Notebooks
- Use **interactive** methods to help users select from top-k options

[C. Yan & Y. He]

Join Wizards



(a) Paxata



(b) Tableau Prep

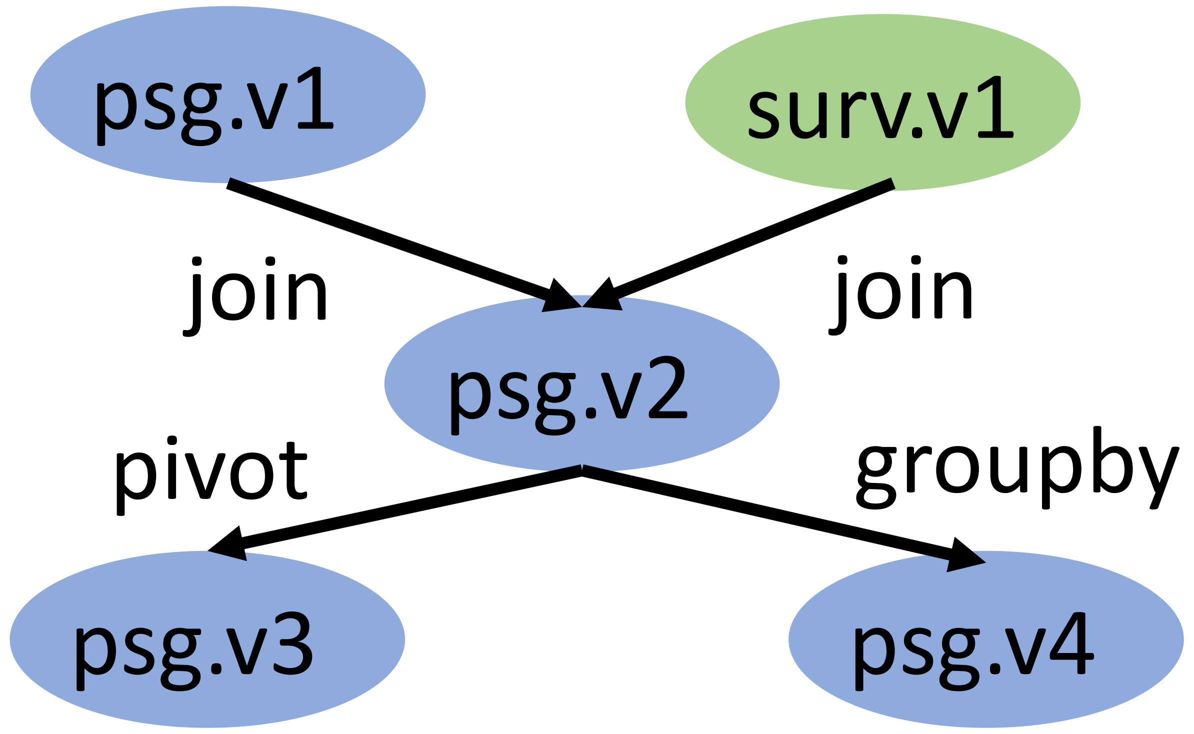
(c) Trifacta

[C. Yan & Y. He]

Programmatic Operators

- Crawl, reapply, and analyze data piplelines from Jupyter+pandas
- 7 API calls: concat, dropna, fillna, groupby, melt, merge, pivot

Logical Operator	Join	Pivot	Unpivot	Groupby	Relationalize JSON
Pandas Operator	merge[17]	pivot[18]	melt[16]	groupby[14]	json_normalize[15]
#nb crawled w/ the operator	209.9K	68.9K	16.8K	364.3K	8.3K




```
import pandas as pd

psg=pd.read_csv('passenger_data.csv')
surv=pd.read_csv('survive.csv')
psg=psg.merge(surv,on='PassengerId',
          how='left')
psg.pivot(header=['Survived, Pclass'],
          index='Sex', aggrfunc='count')
psg.groupby('Sex',aggrfunc='count')
```

[C. Yan & Y. He]

Recommendation Tasks

- Single-Operator Prediction: Given two tables and an operation, decide how to best apply the operation (what are the parameters)
- Next-Operator Prediction: Given all operations performed so far, predict the next one

[C. Yan & Y. He]

Join Prediction

- Predict columns
  - Use features of columns: value-overlap, "left-ness", statistics
- Predict join type
  - Inner join is the default (also 78% of cases in data)
  - "Central" table vs. "filtering"

[C. Yan & Y. He]

Pivot/Unpivot

- Pivot is hard to get right
  - Index
  - Header
  - Aggregation Function
  - Aggregation Columns
- Use GroupBy Prediction
- Look for NULLs and use **affiinity**
- Affinity-Maximizing Pivot Table
- Unpivot requires **compatibility**

Sector	Ticker	Company	Year	Quarter	Market Cap	Revenue
Aerospace	AJRD	AEROJET ROCKETD	2006	Q1	1442.67	472.07
Aerospace	AJRD	AEROJET ROCKETD	2006	Q2	1514.80	489.22
...	...	...	...	...	...	...
Aerospace	BA	BOEING CO	2006	Q1	343.41	210.66
...	...	...	...	...	...	...
Utilities	YORW	YORK WATER CO	2008	Q4	600.19	271.73

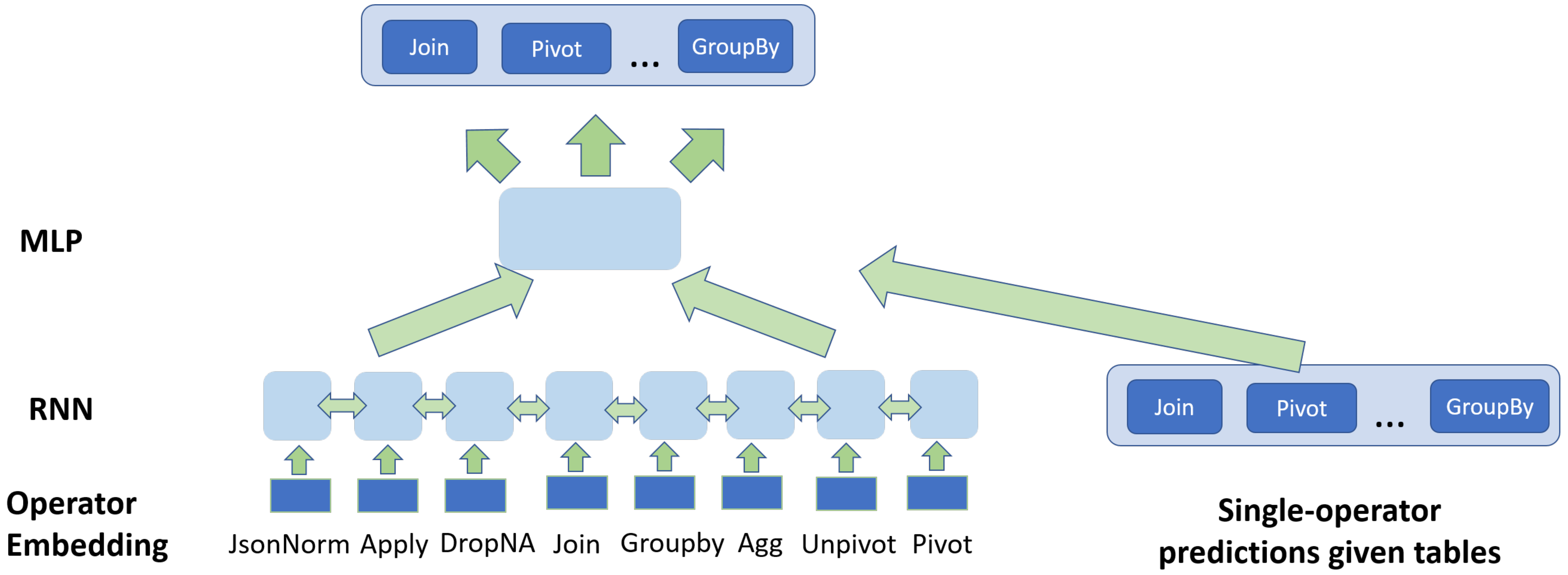
Sector	**Ticker**	**Company**	**2006**	**2007**	**2008**
Aerospace	AJRD	AEROJET ROCKETD	6218.09	6342.45	7088.62
	ATRO	ASTRONICS CORP	1050.97	1071.99	1198.11
Business Services	HHS	HARTE-HANKS INC	2473.75	2523.22	2820.07
	NCMI	NATL CINEMEDIA	856.92	874.06	976.89
Consumer Staples	YTEN	TIELD10 BIOSCI	533.13	543.79	607.77
	...	...	...	...	...
Utilities	YORW	YORK WATER CO	1902.37	1940.42	2168.70

Ticker	**Company**	**Year**	**Aerospace**	**Business Services**	**...**	**Utilities**
AJRD	AEROJET ROCKETD	2006	6218.09	NULL	...	NULL
AJRD	AEROJET ROCKETD	2007	6342.45	NULL	...	NULL
AJRD	AEROJET ROCKETD	2008	7088.62	NULL	...	NULL
ATRO	ASTRONICS CORP	2006	1050.97	NULL	...	NULL
...	...	...	...	...	...	...
HHS	HARTE-HANKS INC	2006	NULL	2473.75	...	NULL
...	...	...	...	...	...	...
YORW	YORK WATER CO	2008	NULL	NULL	...	2168.7

[C. Yan & Y. He]

Predict Next Operator

- Two Signals:
  - Use past information (latent sequential connections)
  - Use table characteristics



[C. Yan & Y. He]

Evaluation

- Data
  - Jupyter Notebooks with working operations

operator	join	pivot	unpivot	groupby	normalize JSON
#nb crawled	209.9K	68.9K	16.8K	364.3K	8.3K
#nb sampled	80K	68.9K	16.8K	80K	8.3K
#nb replayed	12.6K	16.1K	5.7K	9.6K	3.2K
#operator replayed	58.3K	79K	7.2K	70.9K	4.3K
#operator post-filtering	11.2K	7.7K	2.9K	8.9K	1.9K

- Metrics:
  - Precision@K: Proportion of relevant results in the top K
  - NDCG@K (Normalized Discounted Cumulative Gain): ratio of relevance to ideal relevance on a per item basis

[C. Yan & Y. He]

Results

method (all data)	prec@1	prec@2	ndcg@1	ndcg@2
AUTO-SUGGEST	**0.89**	**0.92**	**0.89**	**0.93**
ML-FK	0.84	0.87	0.84	0.87
PowerPivot	0.31	0.44	0.31	0.48
Multi	0.33	0.4	0.33	0.41
Holistic	0.57	0.63	0.57	0.65
max-overlap	0.53	0.61	0.53	0.63
method (sampled data)	prec@1	prec@2	ndcg@1	ndcg@2
AUTO-SUGGEST	**0.92**	-	**0.92**	-
Vendor-A	0.76	-	0.76	-
Vendor-C	0.42	-	0.42	-
Vendor-B	0.33	-	0.33	-

Table 3: Evaluation of Join column prediction. (Top) methods from the literature, evaluated on all test data. (Bottom): Comparisons with commercial systems on a random sample of 200 cases.

feature	left-ness	val-range-overlap	distinct-val-ratio	val-overlap
importance	**0.35**	**0.35**	**0.11**	**0.05**
feature	single-col-candidate	col-val-types	table-stats	sorted-ness
importance	**0.04**	**0.01**	**0.01**	**0.01**

Table 4: Importance of Feature Groups for Join

method	prec@1
AUTO-SUGGEST	**0.88**
Vendor-A	0.78

Table 5: Join type prediction.

[C. Yan & Y. He]

Results

method	full-accuracy	Rand-Index (RI)
AUTO-SUGGEST	**77%**	**0.87**
Affinity	42%	0.56
Type-Rules	19%	0.55
Min-Emptiness	46%	0.70
Balanced-Cut	14%	0.55

Table 8: Pivot: splitting index/header columns.

method	full accuracy	column precision	column recall	column F1
AUTO-SUGGEST	**67%**	**0.93**	**0.96**	**0.94**
Pattern-similarity	21%	0.64	0.46	0.54
Col-name-similarity	27%	0.71	0.53	0.61
Data-type	44%	0.87	0.92	0.89
Contiguous-type	46%	0.80	0.83	0.81

Table 9: Unpivot: Column prediction.

operator	groupby	join	concat	dropna	fillna	pivot	unpivot
percentage	33.3%	27.6%	12.2%	10.8%	9.6%	4.1%	2.4%

Table 10: Distribution of operators in data flows.

method	prec@1	prec@2	recall@1	recall@2
AUTO-SUGGEST	**0.72**	**0.79**	**0.72**	**0.85**
RNN	0.56	0.68	0.56	0.77
N-gram model	0.40	0.53	0.40	0.66
Single-Operators	0.32	0.41	0.32	0.50
Random	0.23	0.35	0.24	0.42

Table 11: Precision for next operator prediction.

[C. Yan & Y. He]