CS 8803-MDS
Human-in-the-loop Data Analytics

Lecture 12

10/03/22

Logistics

Grading

- <= 300 words per answer for paper reviews
- project proposal grades will be released by next week

Project timeline

- project update presentation 10/31
- Assignment 2-6: progress report (1% each)
  - due Fridays 5PM at 10/21, 10/28, 11/4, 11/11, 11/18
- Assignment 7: evaluation plan (5%)
  - due Friday 5PM at 11/18

Next week:

- how to make progress in research?

Today's class

[Finding Related Tables in Data Lakes for Interactive Data Science]

Author: Qiandong, Shen En

Reviewer: Vishnu

Archaeologist: Yanhao

Practioner: Haotian

PORTLAND

ACM SIGMOD/PODS International Conference on Management of Data
June 14-19th, 2020

ACM SIGMOD +PODS 2020

Portland, Oregon USA

SIGMOD PODS

ACM SIGMOD/PODS International Conference on Management of Data June 14 - June 19, 2020 Portland, OR, USA

Welcome

- Homepage
- News

Organization

- Experience Report
- Organization
- SIGMOD PC
- PODS PC

2020 ACM SIGMOD/PODS @ Portland, OR, USA

SIGMOD/PODS 2020 Experience Report now available

Coronavirus Updates

Updates on SIGMOD/PODS 2020 Registration (12 May 2020)

Calls For Submissions

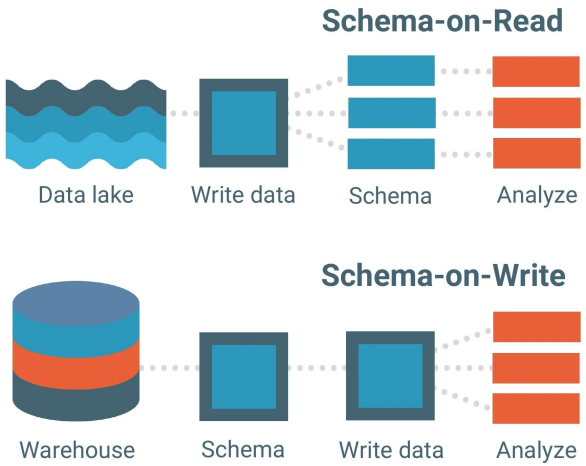
- Important Dates
- Calls for Submissions

PODS Program

- Program Overview
- Detailed Program
- Research Papers
- Keynote Talk

Motivation

- Many data science applications build on data lakes today
- Data Lakes
  - A central repository to store any types of data
  - Schema-on-Read
  - Easy to store data, low storage cost

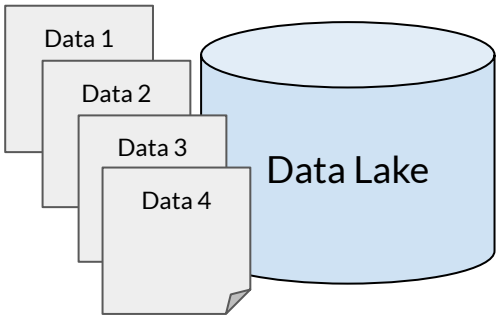


Npujwbujpo

- Data scientists do lots of repeated work every day
  - Importing and cleaning today’s data
  - Data wrangling and exploration on public datasets

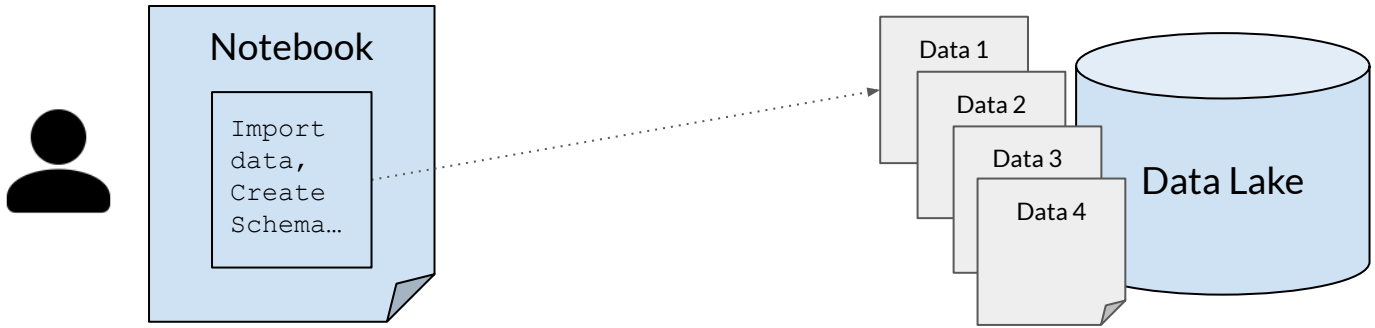
Motivation

- Data scientists do lots of repeated work every day
  - Importing and cleaning today's data
  - Data wrangling and exploration on public datasets



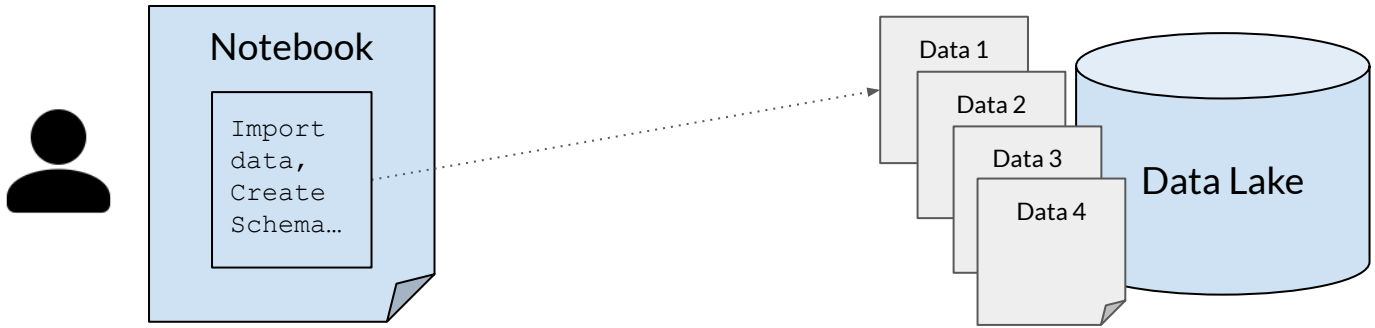
Motivation

- Data scientists do lots of repeated work every day
  - Importing and cleaning today's data
  - Data wrangling and exploration on public datasets
- Same data might already exist



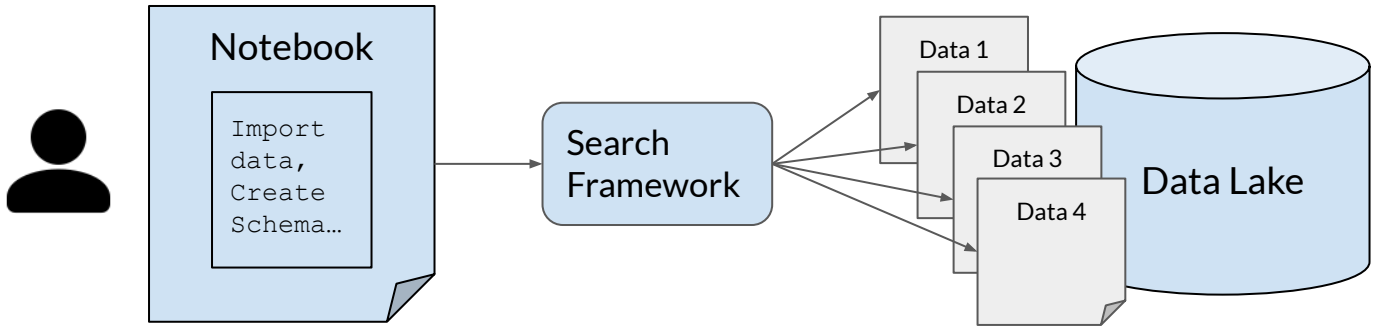
Motivation

- Data scientists do lots of repeated work every day
  - Importing and cleaning today's data
  - Data wrangling and exploration on public datasets
- Same data might already exist
- Data lakes do little to help user find related data
  - Users don't know what data is available, or unable to trust what they find
  - Users reinvent schema - redundant work, data inconsistency



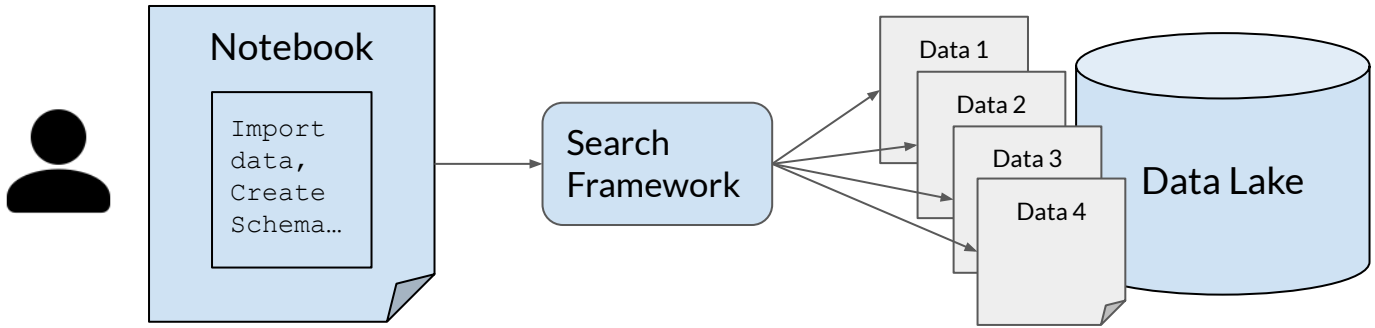
Motivation

- What if we could help users find related data?
  - Promote reusable units of data and processing



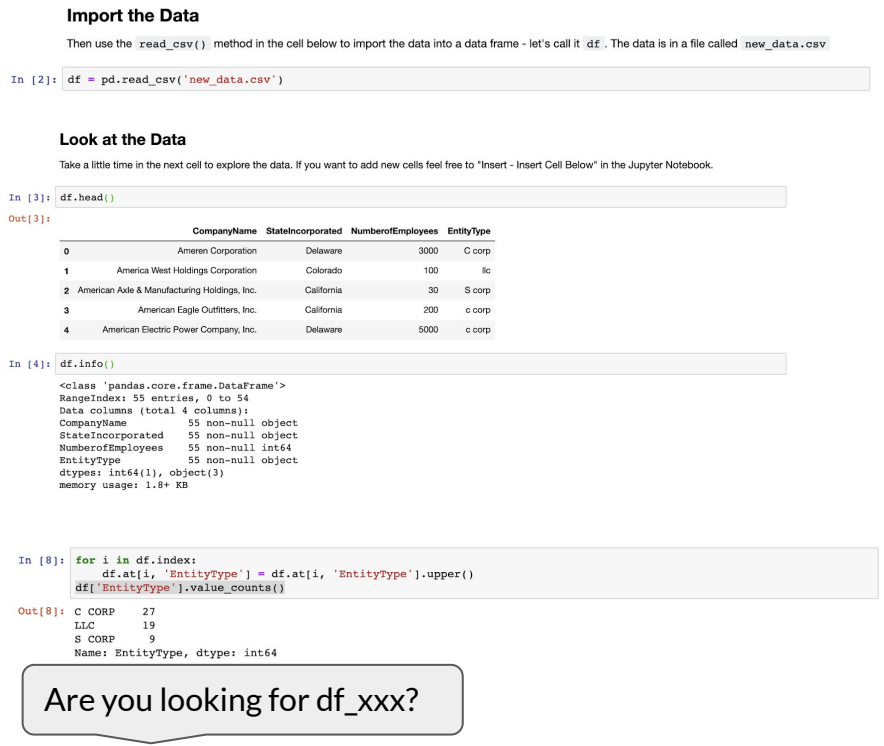
Motivation

- What if we could help users find related data?
  - Promote reusable units of data and processing
- Goal: Build a search framework to find related tables



Motivation

- Based on Jupyter Notebook
- Considering
  - **source data**
  - **cells**
  - **intermediate data**
- Github Copilot for suggesting tables



Import the Data

Then use the `read_csv()` method in the cell below to import the data into a data frame - let's call it `df` . The data is in a file called `new_data.csv`

```
In [2]: df = pd.read_csv('new_data.csv')
```

Look at the Data

Take a little time in the next cell to explore the data. If you want to add new cells feel free to "Insert - Insert Cell Below" in the Jupyter Notebook.

```
In [3]: df.head()
```

Out[3]:

	CompanyName	StateIncorporated	NumberofEmployees	EntityType
0	Ameren Corporation	Delaware	3000	C corp
1	America West Holdings Corporation	Colorado	100	llc
2	American Axle & Manufacturing Holdings, Inc.	California	30	S corp
3	American Eagle Outfitters, Inc.	California	200	c corp
4	American Electric Power Company, Inc.	Delaware	5000	c corp

```
In [4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55 entries, 0 to 54
Data columns (total 4 columns):
CompanyName          55 non-null object
StateIncorporated    55 non-null object
NumberofEmployees    55 non-null int64
EntityType           55 non-null object
dtypes: int64(1), object(3)
memory usage: 1.8+ KB
```

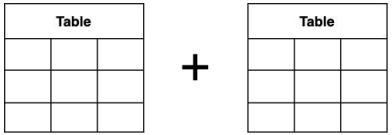
```
In [8]: for i in df.index:
            df.at[i, 'EntityType'] = df.at[i, 'EntityType'].upper()
        df['EntityType'].value_counts()
```

```
Out[8]: C CORP    27
        LLC       19
        S CORP     9
        Name: EntityType, dtype: int64
```

Are you looking for df_xxx?

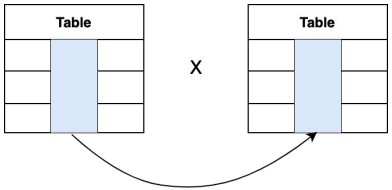
Use Cases

- Augmenting training / validating data
  - Union of two tables
- Linking data
  - Join of two tables
- Extracting machine learning features
  - Adding new columns to the original table
- Cleaning data
  - Fill in missing values in the original table



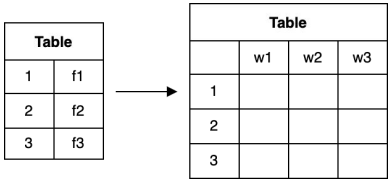
Use Cases

- Augmenting training / validating data
  - Union of two tables
- Linking data
  - Join of two tables
- Extracting machine learning features
  - Adding new columns to the original table
- Cleaning data
  - Fill in missing values in the original table



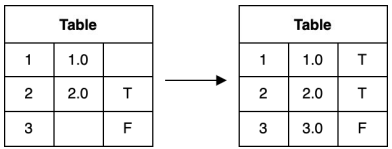
Use Cases

- Augmenting training / validating data
  - Union of two tables
- Linking data
  - Join of two tables
- Extracting machine learning features
  - Adding new columns to the original table
- Cleaning data
  - Fill in missing values in the original table

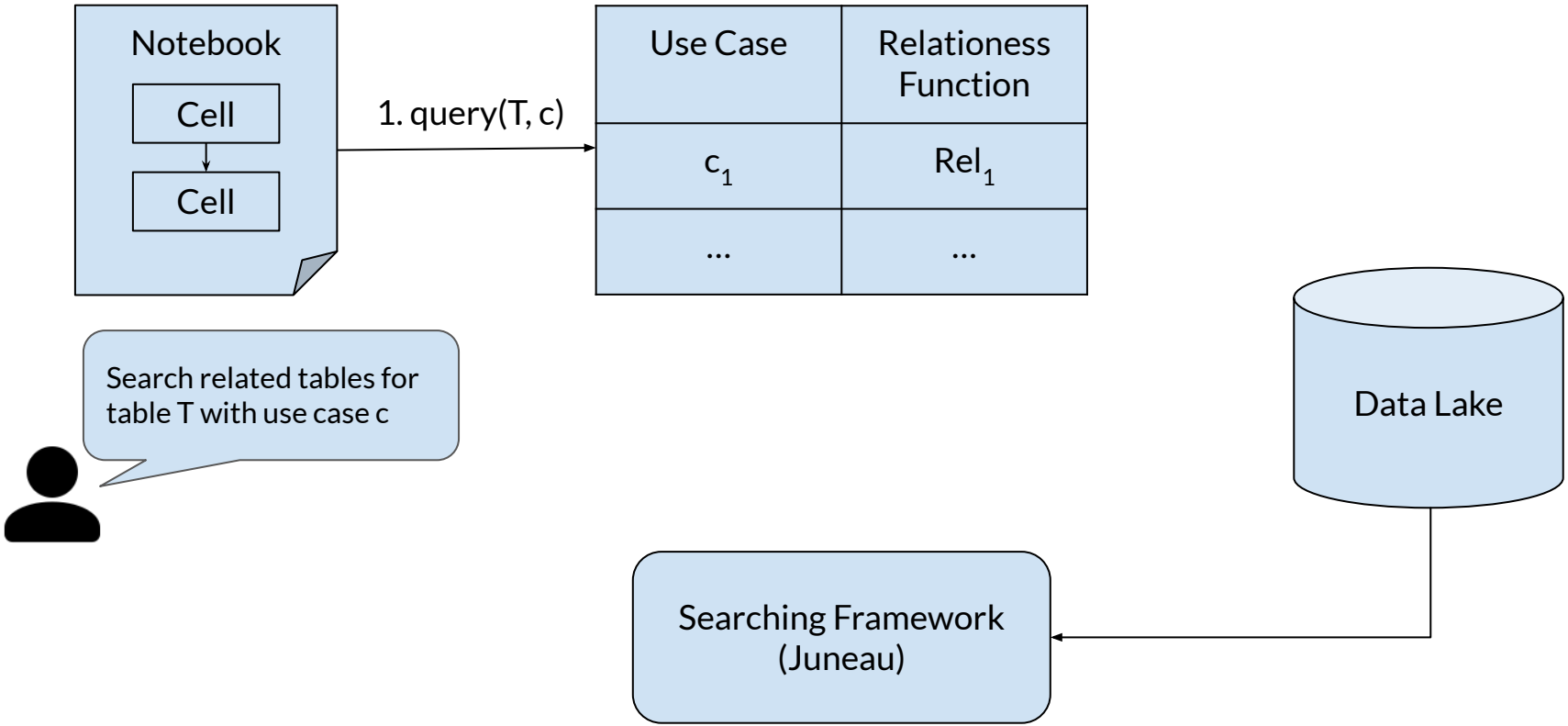


Use Cases

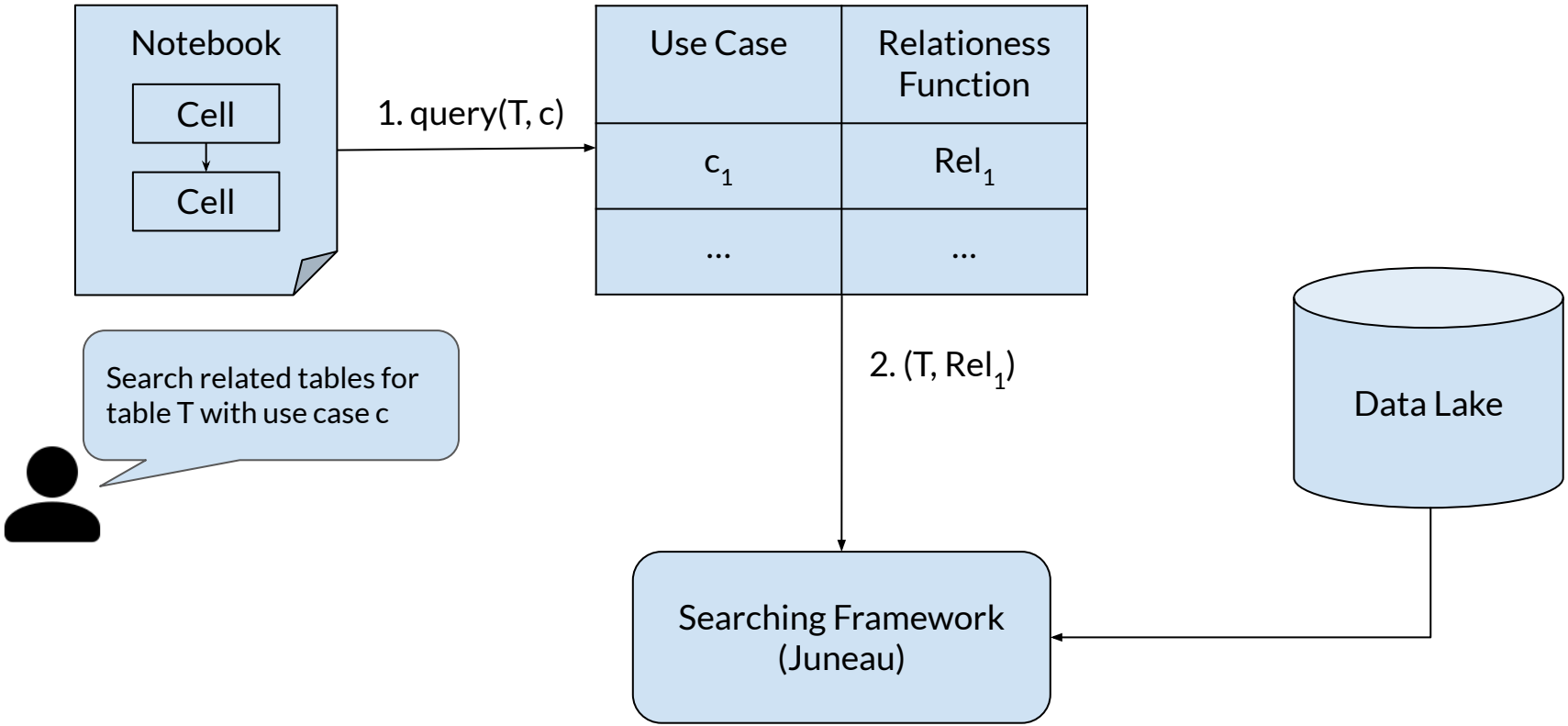
- Augmenting training / validating data
  - Union of two tables
- Linking data
  - Join of two tables
- Extracting machine learning features
  - Adding new columns to the original table
- Cleaning data
  - Fill in missing values in the original table



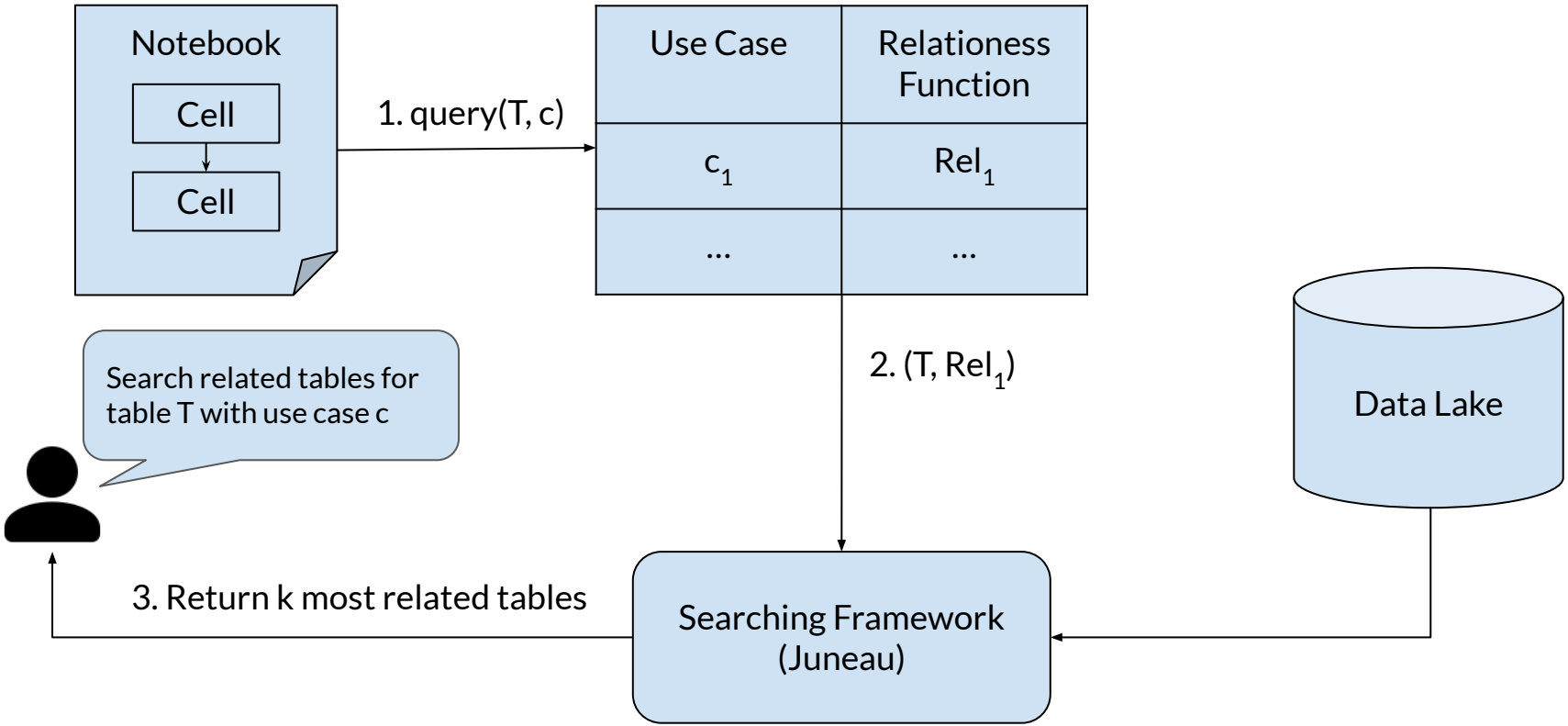
Problem Statement



Problem Statement



Problem Statement



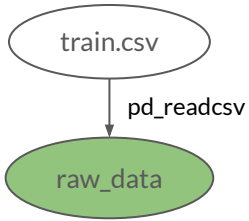
An Example - Feature Extraction

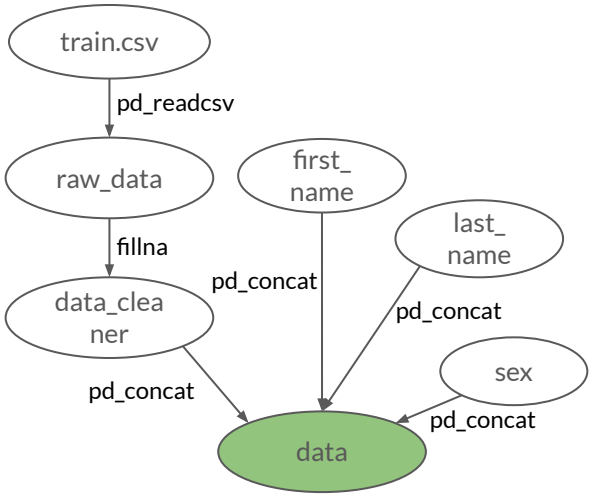
- Feature extraction
  - Produce tables that preserve inputs' keys and other columns, but add new columns

An Example - Feature Extraction

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Fare	Embarked	First Name	Last Name	AgeBand
0	1	0	3	Braund, Mr. Owen Harris	0	1	0	7.2500	S	Braund	Owen Harris	(16.0, 32.0]
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	0	71.2833	C	Cumings	John Bradley (Florence Briggs Thayer)	(32.0, 48.0]
2	3	1	3	Heikkinen, Miss. Laina	1	0	0	7.9250	S	Heikkinen	Laina	(16.0, 32.0]



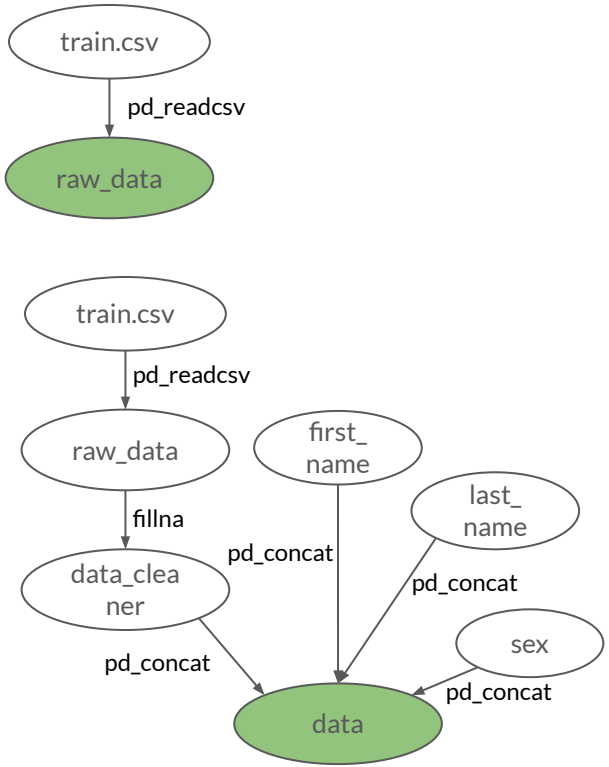


Feature Extraction - Table Overlap

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN

Row Overlap

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Fare	Embarked	First Name	Last Name	AgeBand
0	1	0	3	Braund, Mr. Owen Harris	0	1	0	7.2500	S	Braund	Owen Harris	(16.0, 32.0]
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	0	71.2833	C	Cumings	John Bradley (Florence Briggs Thayer)	(32.0, 48.0]
2	3	1	3	Heikkinen, Miss. Laina	1	0	0	7.9250	S	Heikkinen	Laina	(16.0, 32.0]

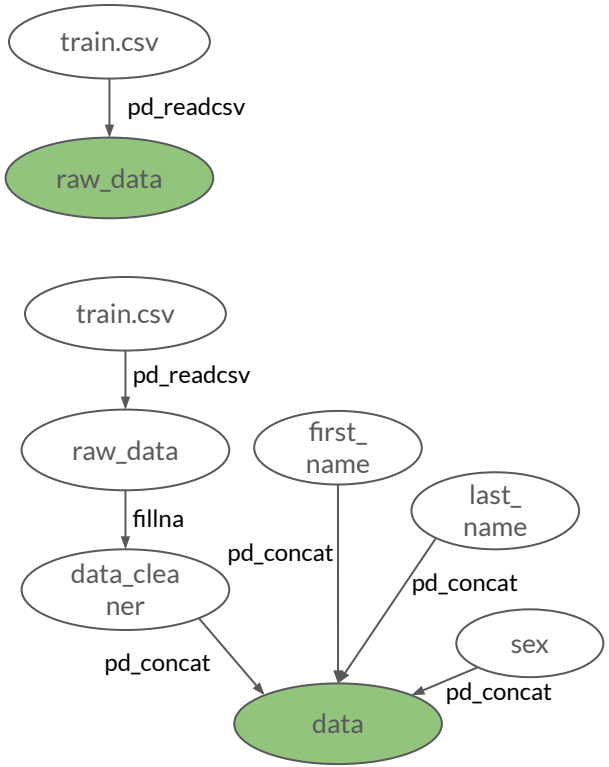


Feature Extraction - Table Overlap

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN

Schema Overlap

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Fare	Embarked	First Name	Last Name	AgeBand
0	1	0	3	Braund, Mr. Owen Harris	0	1	0	7.2500	S	Braund	Owen Harris	(16.0, 32.0]
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	0	71.2833	C	Cumings	John Bradley (Florence Briggs Thayer)	(32.0, 48.0]
2	3	1	3	Heikkinen, Miss. Laina	1	0	0	7.9250	S	Heikkinen	Laina	(16.0, 32.0]

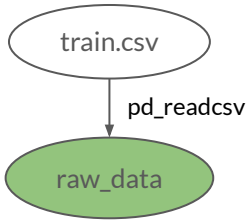


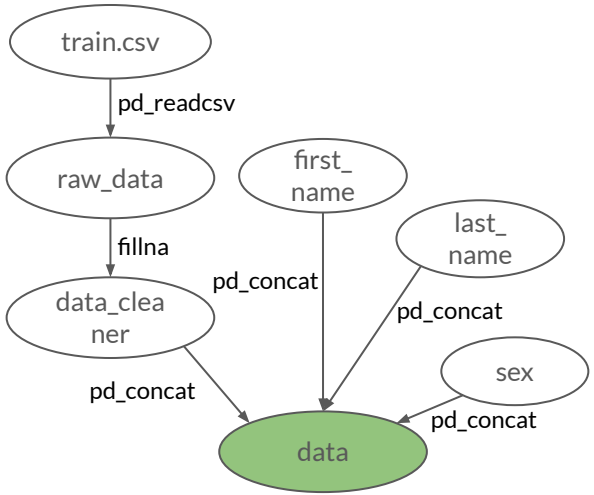
Feature Extraction - New Information

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN

New Columns

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Fare	Embarked	First Name	Last Name	AgeBand
0	1	0	3	Braund, Mr. Owen Harris	0	1	0	7.2500	S	Braund	Owen Harris	(16.0, 32.0]
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	0	71.2833	C	Cumings	John Bradley (Florence Briggs Thayer)	(32.0, 48.0]
2	3	1	3	Heikkinen, Miss. Laina	1	0	0	7.9250	S	Heikkinen	Laina	(16.0, 32.0]



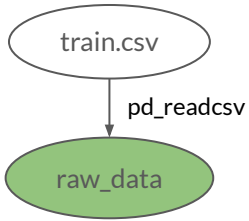


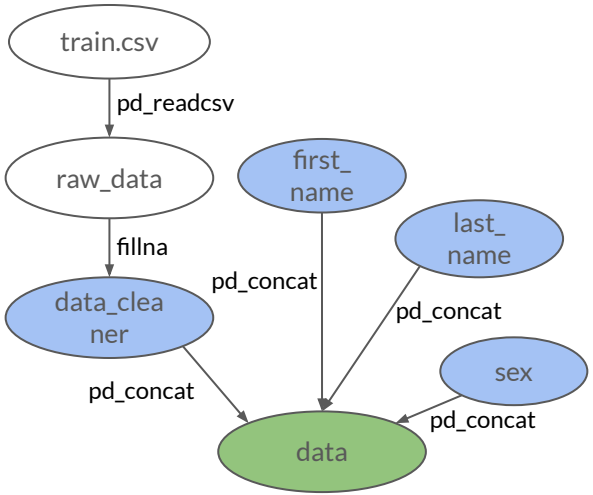
Feature Extraction - New Information

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN

Provenance

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Fare	Embarked	First Name	Last Name	AgeBand
0	1	0	3	Braund, Mr. Owen Harris	0	1	0	7.2500	S	Braund	Owen Harris	(16.0, 32.0]
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	0	71.2833	C	Cumings	John Bradley (Florence Briggs Thayer)	(32.0, 48.0]
2	3	1	3	Heikkinen, Miss. Laina	1	0	0	7.9250	S	Heikkinen	Laina	(16.0, 32.0]





Full Suite of Basic Measures

1. Row/Column Match
   - Measures the row overlap and column overlap between two tables S and T
2. Row/Column Information Gain
   - Measures the new row rate and new column rate of T compared to query table S
3. Shared Provenance
   - Measures the purpose/workflow similarity of two tables S and T based on variable dependencies
4. Null Value Decrement
   - Measures the decrease in null value in T compared to query table S
5. Description Similarity
   - Measures the similarity of table metadata such as problem type, text about the workflow, etc.

Full Suite of Basic Measures

Use Cases	Row Match	Column Match	Row Info Gain	Column Info Gain	Shared Provenance	Null Value Decrement	Description Similarity
Data Augmentation		⬆️⬆️	⬆️⬆️		⬆️⬆️		⬆️
Feature Extraction	⬆️⬆️			⬆️⬆️	⬇️⬇️		⬆️
Data Cleaning	⬆️⬆️	⬆️⬆️			⬆️	⬆️	⬆️
Linkable Data	⬆️⬆️						⬆️

⬆️: the higher the better
⬇️: the lower the better

Top-K Search: Threshold Algorithm

Example: Assume our goal is to find the top-2 (k = 2) candidate tables T_j

Top-K Search: Threshold Algorithm

Step 1: For each measure M_i, order the candidate tables T_j in descending order of scores

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

Top-K Search: Threshold Algorithm

Step 2: Perform sequential scan of the tables, set a threshold t = sum of the scores in the scan

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 1 + 0.8 + 0.8 = 2.6$

Top-K Search: Threshold Algorithm

Step 3: At the same time, for each T_j appearing in the scan, calculate their total scores

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

+

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

+

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

→ $t = 1 + 0.8 + 0.8 = 2.6$

$s_1 = 1 + 0.3 + 0.2 = 1.5$

Top-K Search: Threshold Algorithm

Step 3: At the same time, for each T_j appearing in the scan, calculate their total scores

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

+

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

+

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 1 + 0.8 + 0.8 = 2.6$

$s_1 = 1 + 0.3 + 0.2 = 1.5$

$s_2 = 0.8 + 0.8 + 0 = 1.6$

Top-K Search: Threshold Algorithm

Step 3: At the same time, for each T_j appearing in the scan, calculate their total scores

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

+

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

+

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 1 + 0.8 + 0.8 = 2.6$

$s_1 = 1 + 0.3 + 0.2 = 1.5$

$s_2 = 0.8 + 0.8 + 0 = 1.6$

$s_4 = 0.3 + 0.2 + 0.8 = 1.3$

Top-K Search: Threshold Algorithm

Step 3.5: We only need the top-2 tables, so we only maintain the top-2 total scores

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 1 + 0.8 + 0.8 = 2.6$

$s_2 = 1.6$

$s_1 = 1.5$

~~$s_4 = 0.3 + 0.2 + 0.8 = 1.3$~~

Top-K Search: Threshold Algorithm

Step 4: Repeat Step 2 and Step 3 by:

(1) continuing the scan, (2) updating the threshold, (3) scale total scores of new tables

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

Updated threshold

$t = 0.8 + 0.7 + 0.6 = 2.1$

$s_3 = 1.8$

$s_2 = 1.6$

Updated top-2 total scores:

1. Calculate score for T_3
2. $s_3 = 1.8$ is larger than $s_1 = 1.5$

Top-K Search: Threshold Algorithm

Step 5: Stop when all the top-2 total scores maintained ≥ threshold

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 0.5 + 0.3 + 0.2 = 1$

$s_3 = 1.8$

$s_2 = 1.6$

We have found our top-2 tables

Top-K Search: Threshold Algorithm

Step 5: Stop when all the top-2 total scores maintained ≥ threshold

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 0.5 + 0.3 + 0.2 = 1$

$s_3 = 1.8$

$s_2 = 1.6$

We have found our top-2 tables

Why?

Top-K Search: Threshold Algorithm

Step 5: Stop when all the top-2 total scores maintained ≥ threshold

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 0.5 + 0.3 + 0.2 = 1$

$s_3 = 1.8$

$s_2 = 1.6$

We have found our top-2 tables

Why?

If there were any table that should have been in the top-k list, we would have seen it during the sequential scan

Top-K Search: Threshold Algorithm

Challenge:

Scores requiring relation mapping are much more expensive to compute

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 0.5 + 0.3 + 0.2 = 1$

$s_3 = 1.8$

$s_2 = 1.6$

Top-K Search: Threshold Algorithm

Challenge: e.g. To calculate Row Overlap, we need to know how the table headers map to each other

Scores requiring relation mapping are much more expensive to compute

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

$t = 0.5 + 0.3 + 0.2 = 1$

$s_3 = 1.8$

$s_2 = 1.6$

Computational Challenges

- Relation mappings are expensive
  - For two tables with size (R x C), computing table overlaps: $O(RC^2)$
- Top-k search makes it worse
  - For N tables, top-k search: $O(NRC^2)$

Top-K Search: Juneau's Threshold Algorithm

Example: Assume M_4 is much more expensive to compute than M_1, M_2, and M_3.

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

M_4	Score

Top-K Search: Juneau's Threshold Algorithm

Example: Assume M_4 is much more expensive to compute than M_1, M_2, and M_3.

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

M_4	Score

Instead of pre-compute all scores for M_4, we do it on-demand

Top-K Search: Juneau's Threshold Algorithm

Trick: In each sequential scan, use the Upper Bound of M_4 instead of the individual M_4 scores.

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

M_4	Score
UB	0.9

$t = 0.1 + 0.8 + 0.8 + UB = 3.5$

Top-K Search: Juneau's Threshold Algorithm

Trick: Calculate individual M_4 scores on-demand when covered by the scan

M_1	Score
T_1	1
T_2	0.8
T_3	0.5
T_4	0.3
T_5	0.1

M_2	Score
T_2	0.8
T_3	0.7
T_1	0.3
T_4	0.2
T_5	0.1

M_3	Score
T_4	0.8
T_3	0.6
T_1	0.2
T_5	0.1
T_2	0

M_4	Score
UB	0.9
T_1	0.6
T_2	0.5
T_4	0.3

$t = 0.1 + 0.8 + 0.8 + UB = 3.5$

$s_1 = 1 + 0.3 + 0.2 + 0.6 = 2.1$

$s_2 = 0.8 + 0.8 + 0 + 0.5 = 2.1$

$s_4 = 0.3 + 0.2 + 0.8 + 0.3 = 1.6$

Speeding up relation mapping

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
590	591	0	3	Rintamaki, Mr. Matti	male	35.0	0	0	STON/O 2. 3101273
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224

1. Data Profiles
2. Compositional Profile
3. Workflow Graph

Matcher: f_{name}
Neighbors:
Sex, Age, ...
Value Space:
Rintamaki...

Matcher: f_{sex}
Neighbors:
Name, Age, ...
Value Space:
male, female, ...

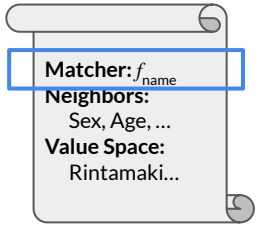
Matcher: f_{age}
Neighbors:
Name, Sex, ...
Value Space:
[1, 120]

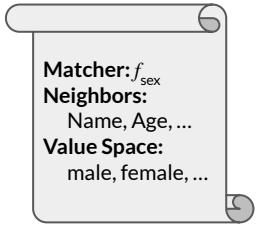
Speeding up relation mapping

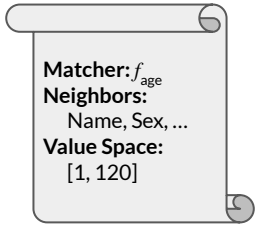
1. Data Profiles
2. Compositional Profile
3. Workflow Graph

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
590	591	0	3	Rintamaki, Mr. Matti	male	35.0	0	0	STON/O 2. 3101273
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224

Matcher tests whether a column belong to the domain





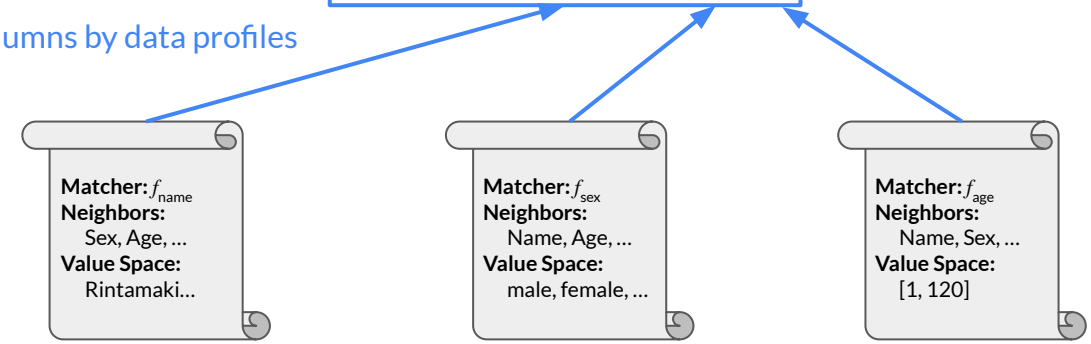


Speeding up relation mapping

1. Data Profiles
2. Compositional Profile
3. Workflow Graph

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
590	591	0	3	Rintamaki, Mr. Matti	male	35.0	0	0	STON/O 2. 3101273
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224

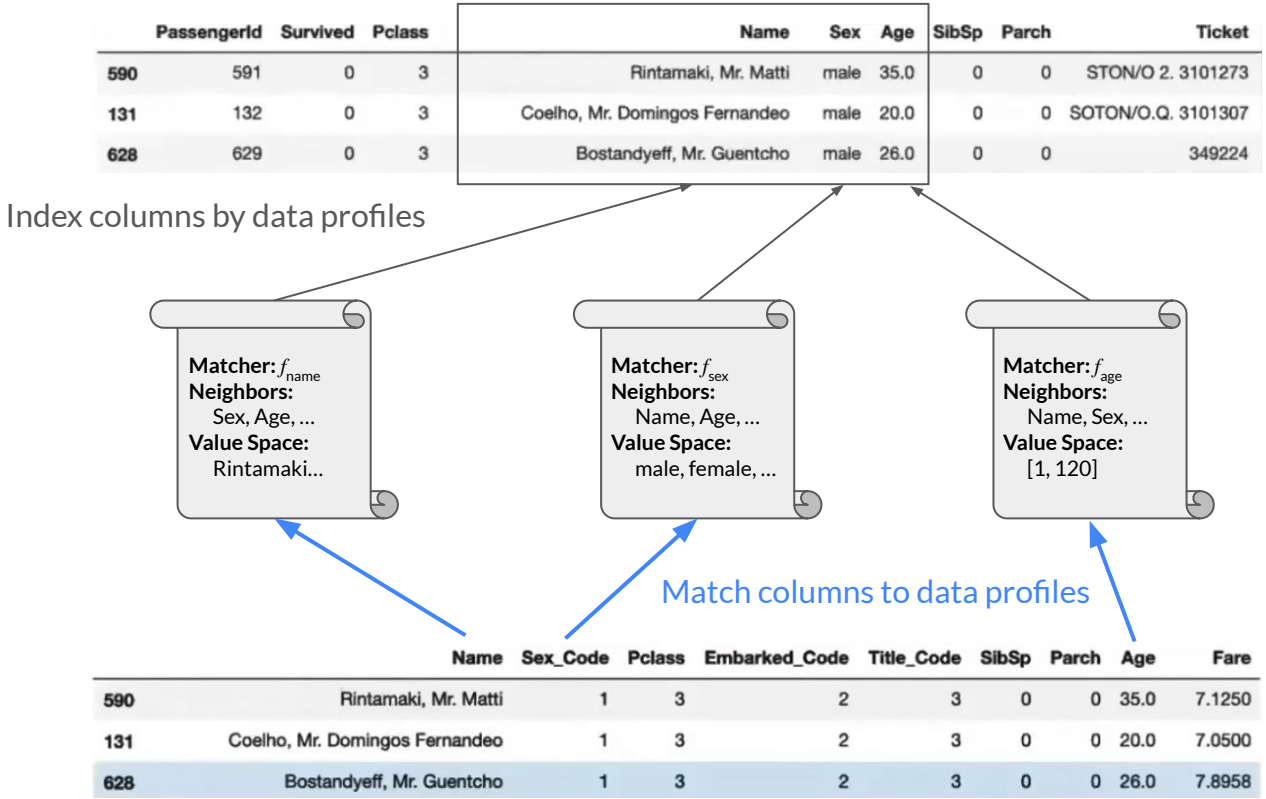
Index columns by data profiles



	Name	Sex_Code	Pclass	Embarked_Code	Title_Code	SibSp	Parch	Age	Fare
590	Rintamaki, Mr. Matti	1	3	2	3	0	0	35.0	7.1250
131	Coelho, Mr. Domingos Fernandeo	1	3	2	3	0	0	20.0	7.0500
628	Bostandyeff, Mr. Guentcho	1	3	2	3	0	0	26.0	7.8958

Speeding up relation mapping

1. Data Profiles
2. Compositional Profile
3. Workflow Graph

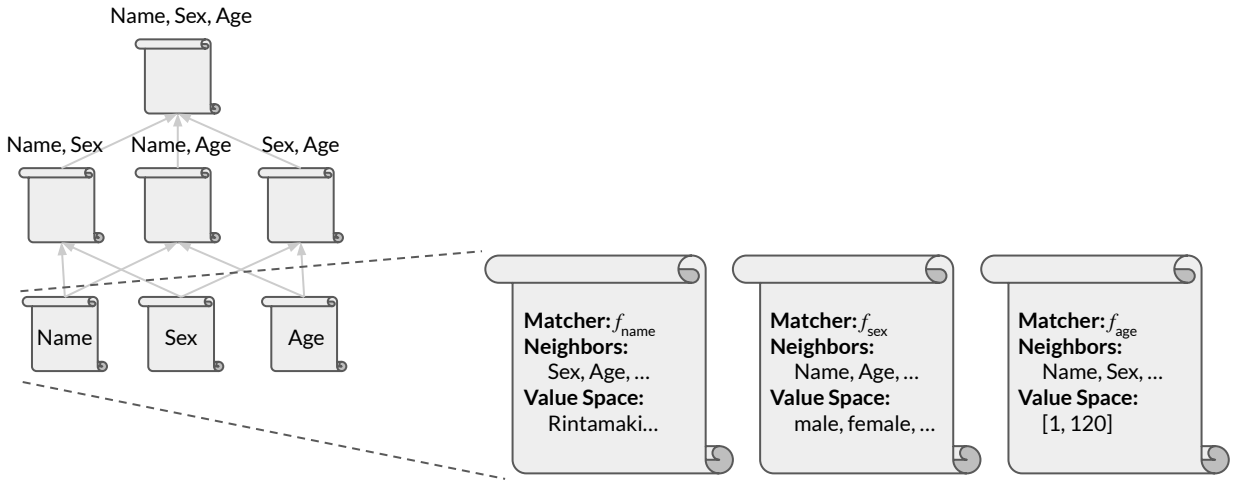


Speeding up relation mapping

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
590	591	0	3	Rintamaki, Mr. Matti	male	35.0	0	0	STON/O 2. 3101273
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224

1. Data Profiles
2. Compositional Profile
3. Workflow Graph

Data Profile Indices



Speeding up relation mapping

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	BirthDate
590	591	0	3	Rintamaki, Mr. Matti	male	35.0	0	0	STON/O 2. 3101273	**1877/09/01**
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307	**1892/10/23**
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224	**1886/03/15**

Index column sets based on common co-occurrences

Compositional Profile Index

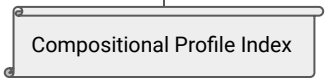
1. Data Profiles
2. Compositional Profile
3. Workflow Graph

	Name	Sex_Code	Pclass	Embarked_Code	Title_Code	SibSp	Parch	Age	Fare	BirthDate
590	Rintamaki, Mr. Matti	1	3	2	3	0	0	35.0	7.1250	**1877/09/01**
131	Coelho, Mr. Domingos Fernandeo	1	3	2	3	0	0	20.0	7.0500	**1892/10/23**
628	Bostandyeff, Mr. Guentcho	1	3	2	3	0	0	26.0	7.8958	**1886/03/15**

Speeding up relation mapping

1. Data Profiles
2. Compositional Profile
3. Workflow Graph

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	BirthDate
590	591	0	3	Rintamaki, Mr. Matti	male	35.0	0	0	STON/O 2. 3101273	**1877/09/01**
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307	**1892/10/23**
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224	**1886/03/15**



If a column pair matches (*Name*),

	Name	Sex_Code	Pclass	Embarked_Code	Title_Code	SibSp	Parch	Age	Fare	BirthDate
590	Rintamaki, Mr. Matti	1	3	2	3	0	0	35.0	7.1250	**1877/09/01**
131	Coelho, Mr. Domingos Fernandeo	1	3	2	3	0	0	20.0	7.0500	**1892/10/23**
628	Bostandyeff, Mr. Guentcho	1	3	2	3	0	0	26.0	7.8958	**1886/03/15**

Speeding up relation mapping

1. Data Profiles
2. Compositional Profile
3. Workflow Graph

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	BirthDate
590	591	0	3	Rintamaki, Mr. Matti	male	35.0	0	0	STON/O 2. 3101273	1877/09/01
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307	1892/10/23
628	629	0	3	Bostandyeff, Mr. Guentcho	male	26.0	0	0	349224	1886/03/15

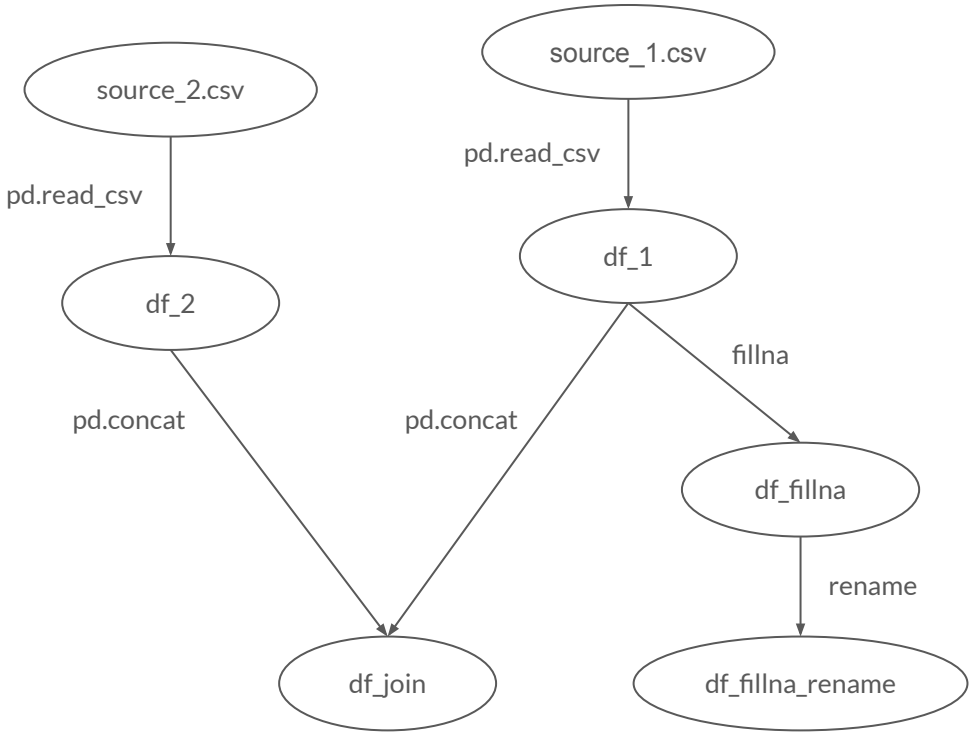
Compositional Profile Index

If a column pair matches (*Name*),
check if other columns (*BirthDate*)
in the compositional profile also match.

	Name	Sex_Code	Pclass	Embarked_Code	Title_Code	SibSp	Parch	Age	Fare	BirthDate
590	Rintamaki, Mr. Matti	1	3	2	3	0	0	35.0	7.1250	1877/09/01
131	Coelho, Mr. Domingos Fernandeo	1	3	2	3	0	0	20.0	7.0500	1892/10/23
628	Bostandyeff, Mr. Guentcho	1	3	2	3	0	0	26.0	7.8958	1886/03/15

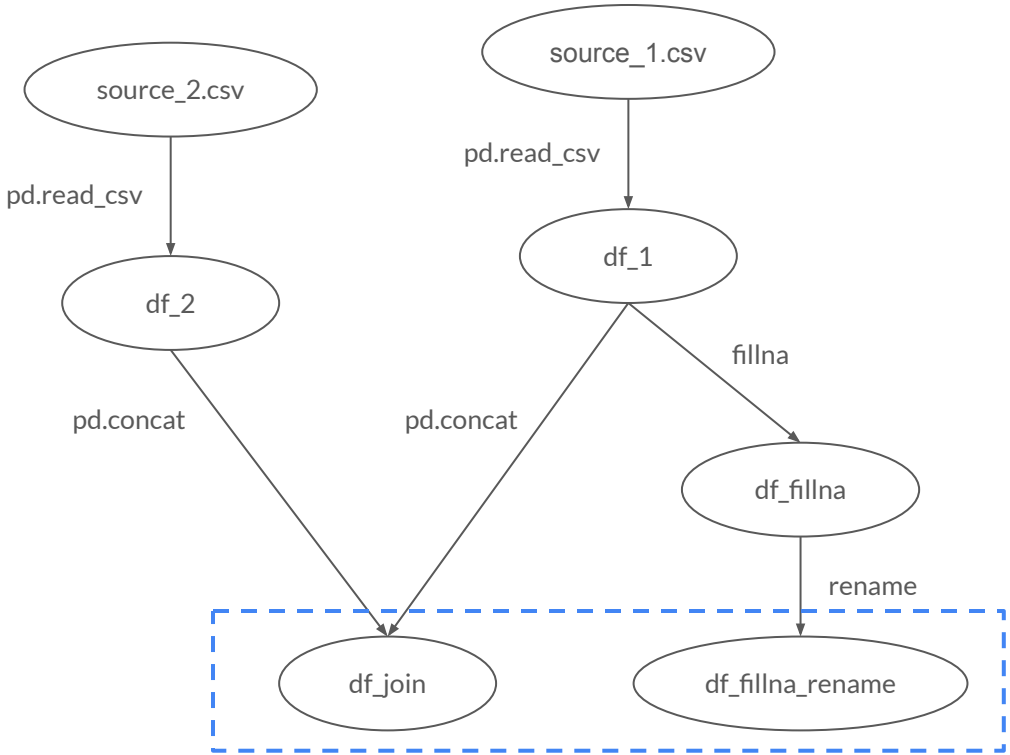
Speeding up relation mapping

1. Data Profiles
2. Compositional Profile
3. Workflow Graph



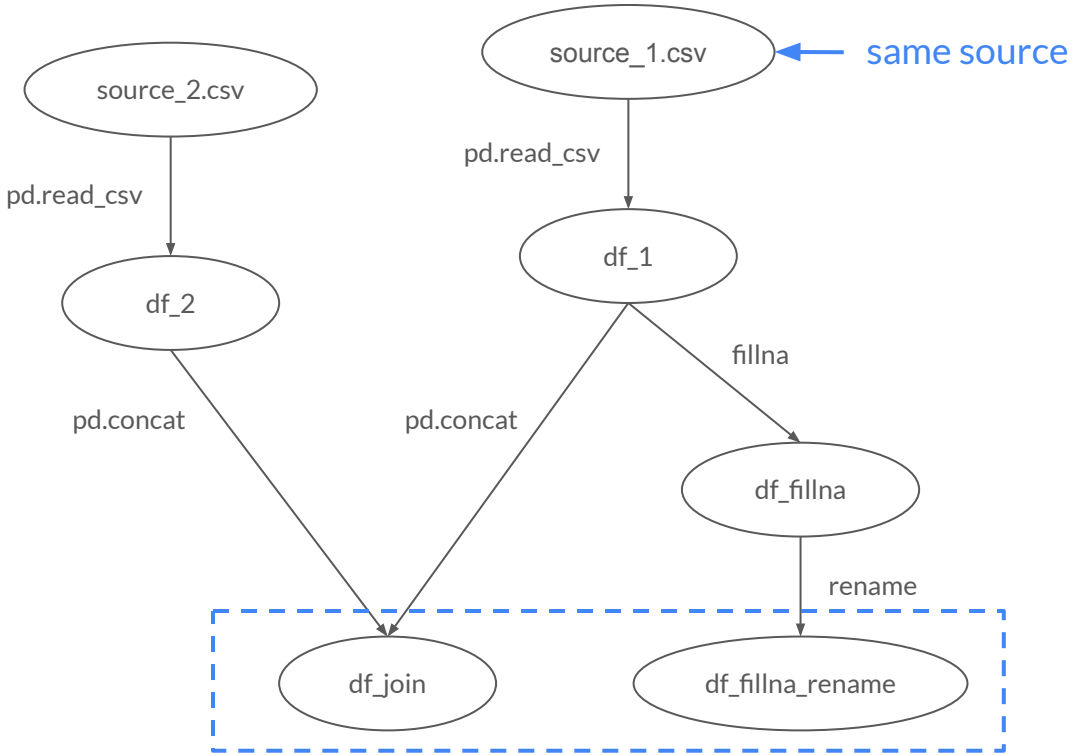
Speeding up relation mapping

1. Data Profiles
2. Compositional Profile
3. Workflow Graph

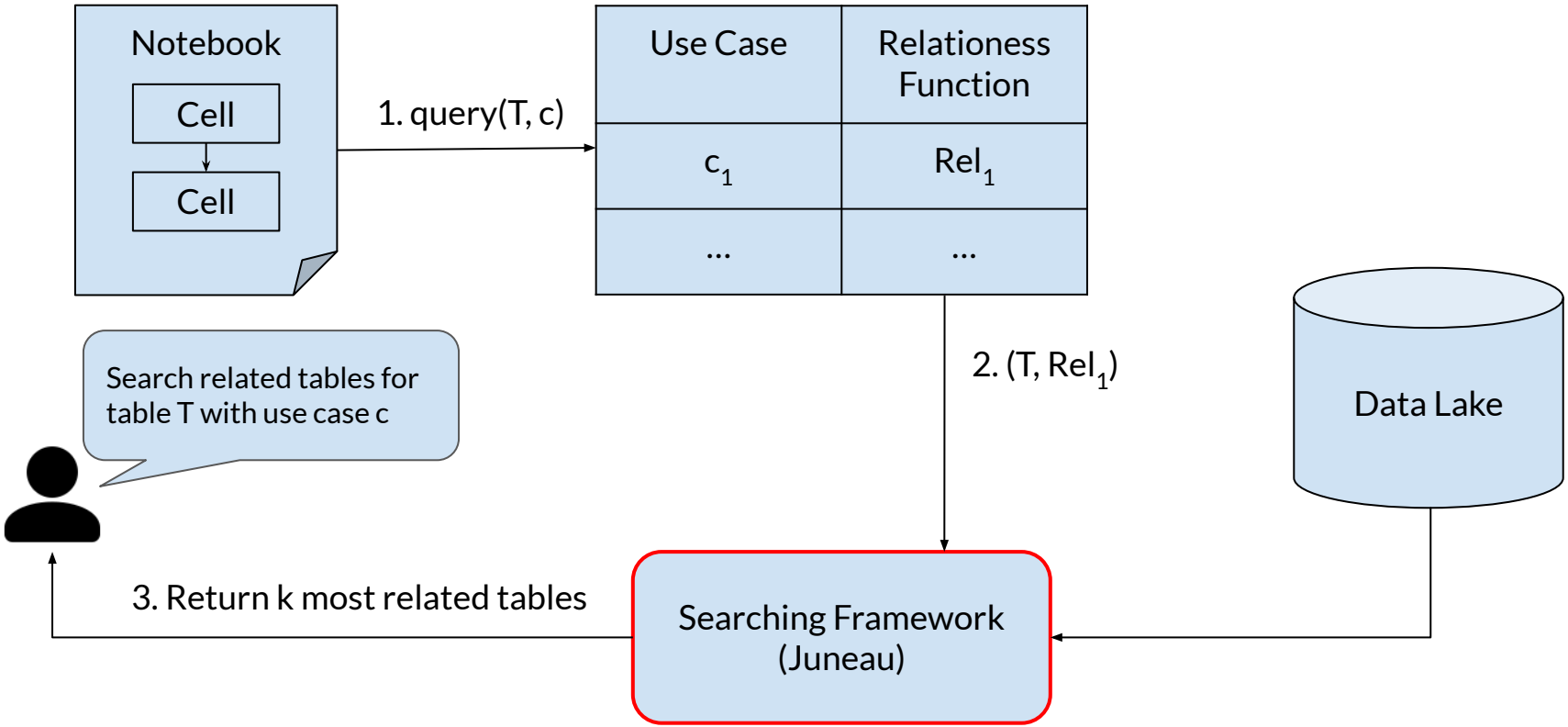


Speeding up relation mapping

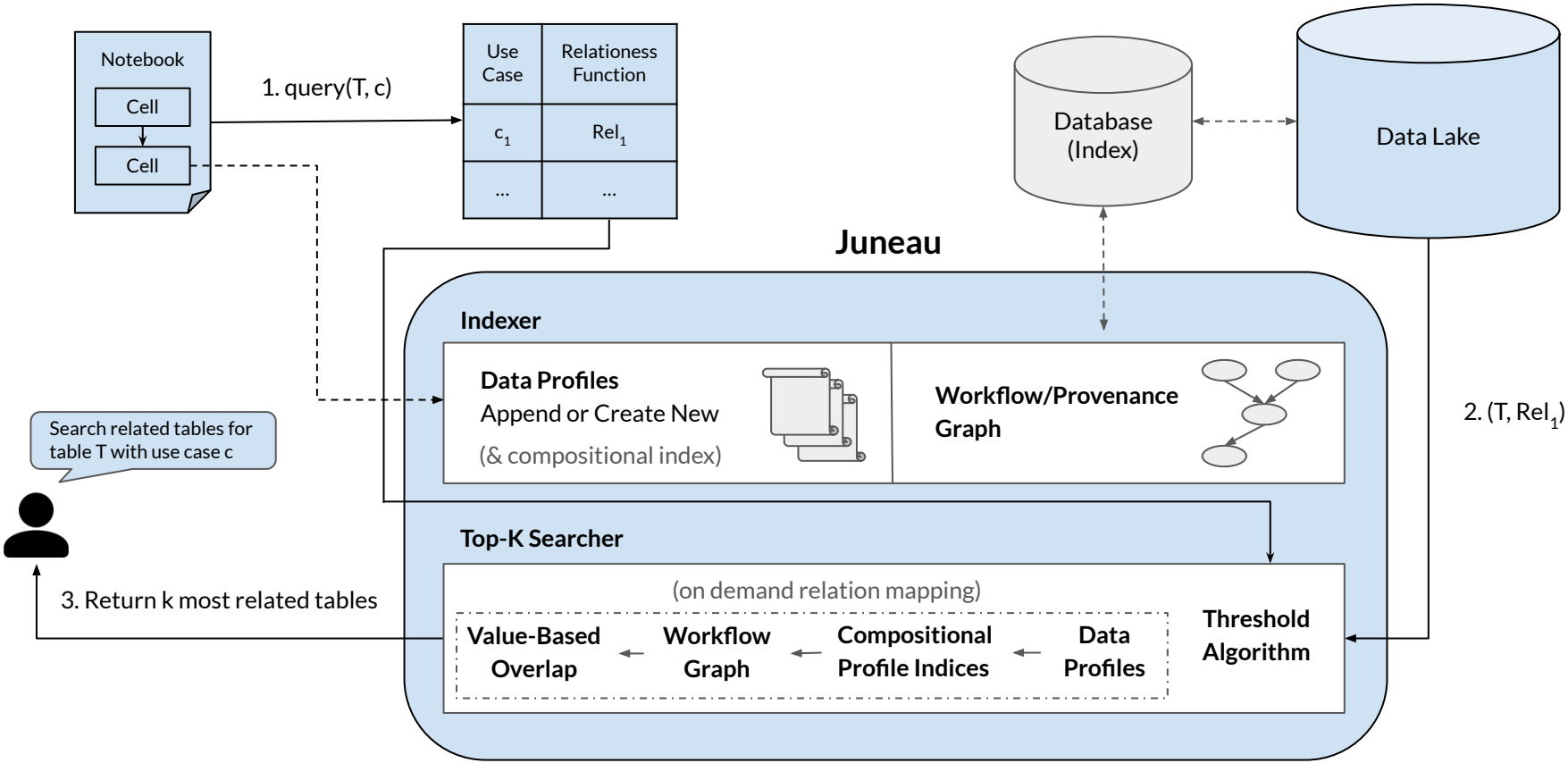
1. Data Profiles
2. Compositional Profile
3. Workflow Graph



System Implementation



System Implementation



Experiment Setup

- 5000+ indexed tables
- 102 notebooks from Kaggle, hand-labeled for use cases
- 14 different Kaggle tasks

Experimental Evaluation: Search Efficiency

Table 3: Average running time (seconds) of returning top−k tables when searching for related tables

top-k	Augmenting Training Data				Feature Extraction				Alternative Data Cleaning				Linking Data			
Q_4	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	919.4	1017	9.776	1.481	875.3	2036	8.288	2.332	947.7	1454	5.833	3.133	22.64	2252	14.67	3.033
5	919.4	1086	9.944	3.254	875.3	2149	8.624	2.336	947.7	1461	5.999	3.145	22.64	2252	15.077	3.261
10	919.4	1106	12.17	5.105	875.3	2228	9.744	3.302	947.7	1488	7.029	4.364	22.64	2252	15.510	4.169
Q_5	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	1755	NF	23.51	5.119	1691	NF	21.30	5.188	1871	NF	20.46	4.318	127.9	NF	22.44	4.246
5	1755	NF	23.86	5.290	1691	NF	21.87	5.417	1871	NF	20.87	4.517	127.9	NF	22.51	4.343
10	1755	NF	24.15	5.532	1691	NF	23.29	5.666	1871	NF	20.95	4.892	127.9	NF	22.73	4.864

L64 = LSH Ensemble

TA = (Ours) Threshold Algorithm (without Data Profiles and Indices)

TA+P = (Ours) Threshold Algorithm (with Data Profiles and Indices)

SJ = (Ours) Full Juneau

Experimental Evaluation: Search Efficiency

Table 3: Average running time (seconds) of returning top−k tables when searching for related tables

top-k	Augmenting Training Data				Feature Extraction				Alternative Data Cleaning				Linking Data			
Q_4	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	919.4	1017	9.776	1.481	875.3	2036	8.288	2.332	947.7	1454	5.833	3.133	22.64	2252	14.67	3.033
5	919.4	1086	9.944	3.254	875.3	2149	8.624	2.336	947.7	1461	5.999	3.145	22.64	2252	15.077	3.261
10	919.4	1106	12.17	5.105	875.3	2228	9.744	3.302	947.7	1488	7.029	4.364	22.64	2252	15.510	4.169
Q_5	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	1755	NF	23.51	5.119	1691	NF	21.30	5.188	1871	NF	20.46	4.318	127.9	NF	22.44	4.246
5	1755	NF	23.86	5.290	1691	NF	21.87	5.417	1871	NF	20.87	4.517	127.9	NF	22.51	4.343
10	1755	NF	24.15	5.532	1691	NF	23.29	5.666	1871	NF	20.95	4.892	127.9	NF	22.73	4.864

> 2300 seconds

L64 = LSH Ensemble

TA = (Ours) Threshold Algorithm (without Data Profiles and Indices)

TA+P = (Ours) Threshold Algorithm (with Data Profiles and Indices)

SJ = (Ours) Full Juneau

Experimental Evaluation: Search Efficiency

Table 3: Average running time (seconds) of returning top–k tables when searching for related tables

top-k	Augmenting Training Data				Feature Extraction				Alternative Data Cleaning				Linking Data			
Q_4	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	919.4	1017	9.776	1.481	875.3	2036	8.288	2.332	947.7	1454	5.833	3.133	22.64	2252	14.67	3.033
5	919.4	1086	9.944	3.254	875.3	2149	8.624	2.336	947.7	1461	5.999	3.145	22.64	2252	15.077	3.261
10	919.4	1106	12.17	5.105	875.3	2228	9.744	3.302	947.7	1488	7.029	4.364	22.64	2252	15.510	4.169
Q_5	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	1755	NF	23.51	5.119	1691	NF	21.30	5.188	1871	NF	20.46	4.318	127.9	NF	22.44	4.246
5	1755	NF	23.86	5.290	1691	NF	21.87	5.417	1871	NF	20.87	4.517	127.9	NF	22.51	4.343
10	1755	NF	24.15	5.532	1691	NF	23.29	5.666	1871	NF	20.95	4.892	127.9	NF	22.73	4.864

Adding data profiles and indices speeds up the search by 100x.

L64 = LSH Ensemble

TA = (Ours) Threshold Algorithm (without Data Profiles and Indices)

TA+P = (Ours) Threshold Algorithm (with Data Profiles and Indices)

SJ = (Ours) Full Juneau

Experimental Evaluation: Search Efficiency

Table 3: Average running time (seconds) of returning top−k tables when searching for related tables

top-k	Augmenting Training Data				Feature Extraction				Alternative Data Cleaning				Linking Data			
Q_4	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	919.4	1017	9.776	1.481	875.3	2036	8.288	2.332	947.7	1454	5.833	3.133	22.64	2252	14.67	3.033
5	919.4	1086	9.944	3.254	875.3	2149	8.624	2.336	947.7	1461	5.999	3.145	22.64	2252	15.077	3.261
10	919.4	1106	12.17	5.105	875.3	2228	9.744	3.302	947.7	1488	7.029	4.364	22.64	2252	15.510	4.169
Q_5	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	1755	NF	23.51	5.119	1691	NF	21.30	5.188	1871	NF	20.46	4.318	127.9	NF	22.44	4.246
5	1755	NF	23.86	5.290	1691	NF	21.87	5.417	1871	NF	20.87	4.517	127.9	NF	22.51	4.343
10	1755	NF	24.15	5.532	1691	NF	23.29	5.666	1871	NF	20.95	4.892	127.9	NF	22.73	4.864

Including workflow indices further decreases the compute time by 3-5x.

L64 = LSH Ensemble

TA = (Ours) Threshold Algorithm (without Data Profiles and Indices)

TA+P = (Ours) Threshold Algorithm (with Data Profiles and Indices)

SJ = (Ours) Full Juneau

Experimental Evaluation: Search Efficiency

Table 3: Average running time (seconds) of returning top−k tables when searching for related tables

top-k	Augmenting Training Data				Feature Extraction				Alternative Data Cleaning				Linking Data			
Q_4	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	919.4	1017	9.776	1.481	875.3	2036	8.288	2.332	947.7	1454	5.833	3.133	22.64	2252	14.67	3.033
5	919.4	1086	9.944	3.254	875.3	2149	8.624	2.336	947.7	1461	5.999	3.145	22.64	2252	15.077	3.261
10	919.4	1106	12.17	5.105	875.3	2228	9.744	3.302	947.7	1488	7.029	4.364	22.64	2252	15.510	4.169
Q_5	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ	L64	TA	TA+P	SJ
1	1755	NF	23.51	5.119	1691	NF	21.30	5.188	1871	NF	20.46	4.318	127.9	NF	22.44	4.246
5	1755	NF	23.86	5.290	1691	NF	21.87	5.417	1871	NF	20.87	4.517	127.9	NF	22.51	4.343
10	1755	NF	24.15	5.532	1691	NF	23.29	5.666	1871	NF	20.95	4.892	127.9	NF	22.73	4.864

Juneau is multiple orders of magnitude faster than the baseline LSH ensemble.

L64 = LSH Ensemble

TA = (Ours) Threshold Algorithm (without Data Profiles and Indices)

TA+P = (Ours) Threshold Algorithm (with Data Profiles and Indices)

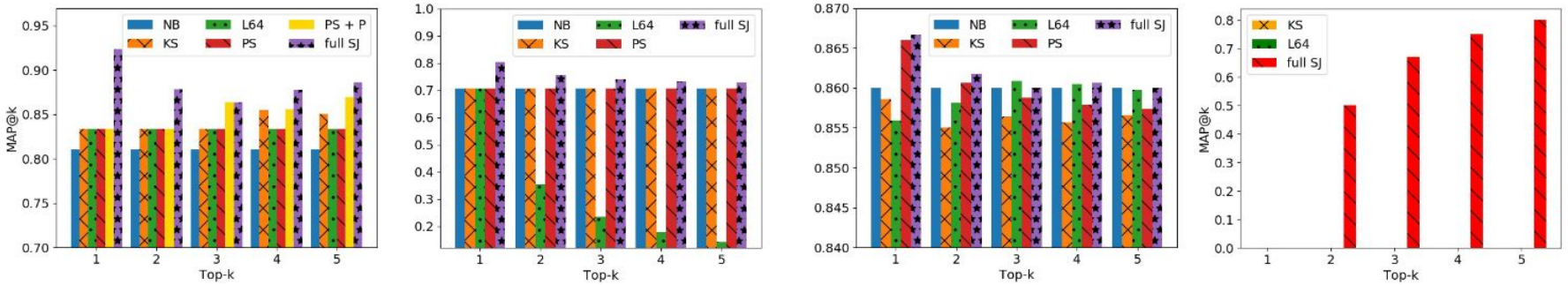
SJ = (Ours) Full Juneau

Experimental Evaluation: Search Result Quality

How do we measure the result quality?

- **Data Augmentation**
  - Precision of classifier trained on augmented data
- **Feature Extraction**
  - Precision of classifier trained on extracted features
- **Data Cleaning**
  - Precision of classifier trained on cleaned data
- **Linkable Data**
  - Check if the returned table R is joined with T
    (T is some table joined with the query table S in the original notebook workflow)

Experimental Evaluation: Search Result Quality



(a) Augmenting Training Data (b) Feature Extraction (c) Data Cleaning (d) Linkable Data

Figure 3: MAP@K of tables returned by different search classes

NB = Original performance in the notebook

KS = Keyword Search

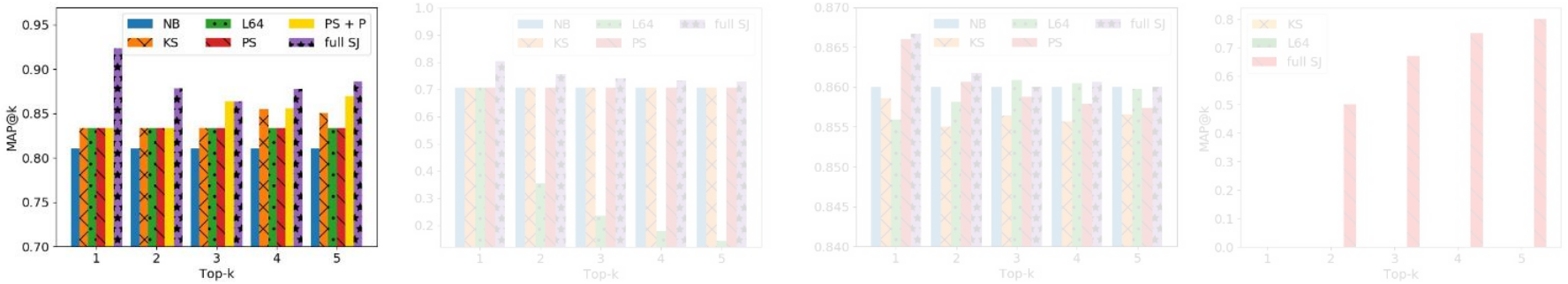
L64 = LSH Ensemble

PS = (Ours) Row/Column Similarity Only

PS + J = (Ours) R/C Sim. + Provenance Sim.

full SJ = (Ours) Full Juneau

Experimental Evaluation: Search Result Quality



(a) Augmenting Training Data (b) Feature Extraction (c) Data Cleaning (d) Linkable Data

Figure 3: MAP@K of tables returned by different search classes

1. **Data Augmentation**: Significantly better than the baseline and other strategies

NB = Original performance in the notebook

KS = Keyword Search

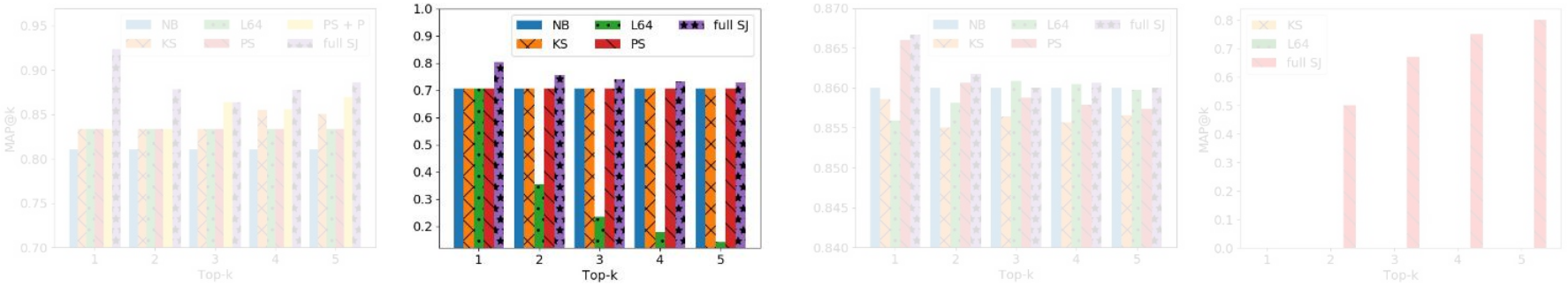
L64 = LSH Ensemble

PS = (Ours) Row/Column Similarity Only

PS + J = (Ours) R/C Sim. + Provenance Sim.

full SJ = (Ours) Full Juneau

Experimental Evaluation: Search Result Quality



(a) Augmenting Training Data (b) Feature Extraction (c) Data Cleaning (d) Linkable Data

Figure 3: MAP@K of tables returned by different search classes

1. **Data Augmentation**: Significantly better than the baseline and other strategies
2. **Feature Extraction**: Slightly improved results

NB = Original performance in the notebook

KS = Keyword Search

L64 = LSH Ensemble

PS = (Ours) Row/Column Similarity Only

PS + J = (Ours) R/C Sim. + Provenance Sim.

full SJ = (Ours) Full Juneau

Experimental Evaluation: Search Result Quality

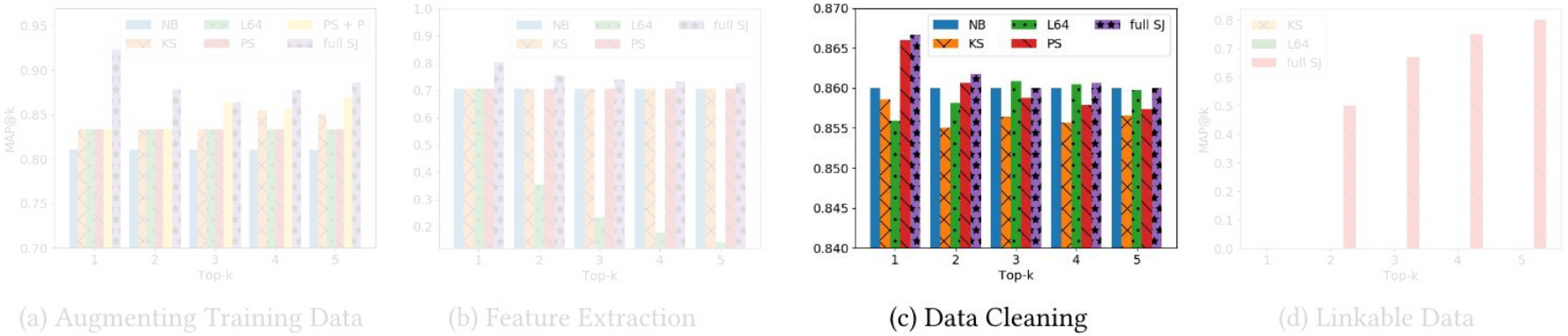


Figure 3: MAP@K of tables returned by different search classes

1. **Data Augmentation**: Significantly better than the baseline and other strategies
2. **Feature Extraction**: Slightly improved results
3. **Data Cleaning**: Measurable but minor impacts

NB = Original performance in the notebook

KS = Keyword Search

L64 = LSH Ensemble

PS = (Ours) Row/Column Similarity Only

PS + J = (Ours) R/C Sim. + Provenance Sim.

full SJ = (Ours) Full Juneau

Experimental Evaluation: Search Result Quality

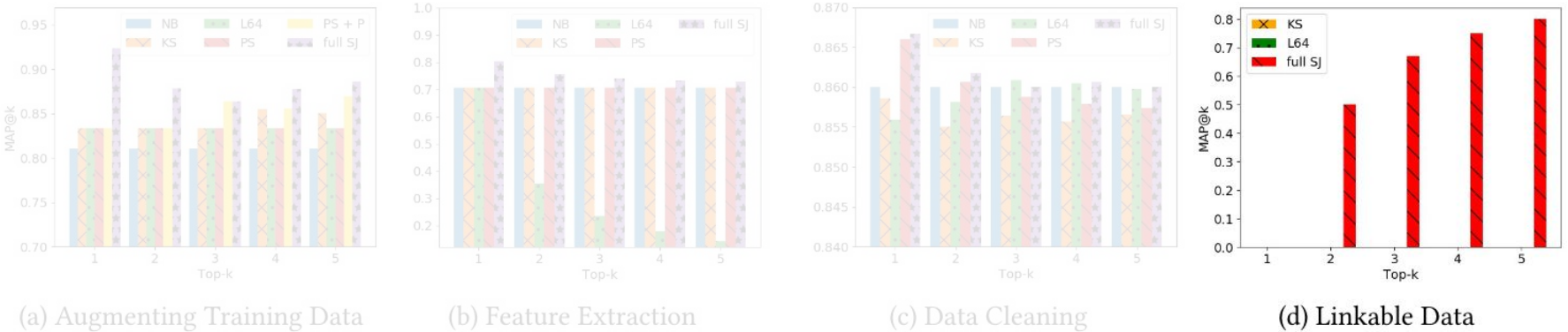


Figure 3: MAP@K of tables returned by different search classes

1. **Data Augmentation**: Significantly better than the baseline and other strategies
2. **Feature Extraction**: Slightly improved results
3. **Data Cleaning**: Measurable but minor impacts
4. **Linkable Data**: Provides meaningful tables

NB = Original performance in the notebook

KS = Keyword Search

L64 = LSH Ensemble

PS = (Ours) Row/Column Similarity Only

PS + J = (Ours) R/C Sim. + Provenance Sim.

full SJ = (Ours) Full Juneau

Conclusions & Future Work

We contributed to the reusability and modularity for data processing through

1. Building a query-by-table framework for data lakes
2. Developing basic measures for table relatenesses and specialized compositional measures for 4 main use cases.
3. Developing algorithmic and indexing strategies for efficient top-k search
4. Showing promising scalability and quality for the search

Next Steps: scaling to distributed settings + heterogeneous data types

Demo

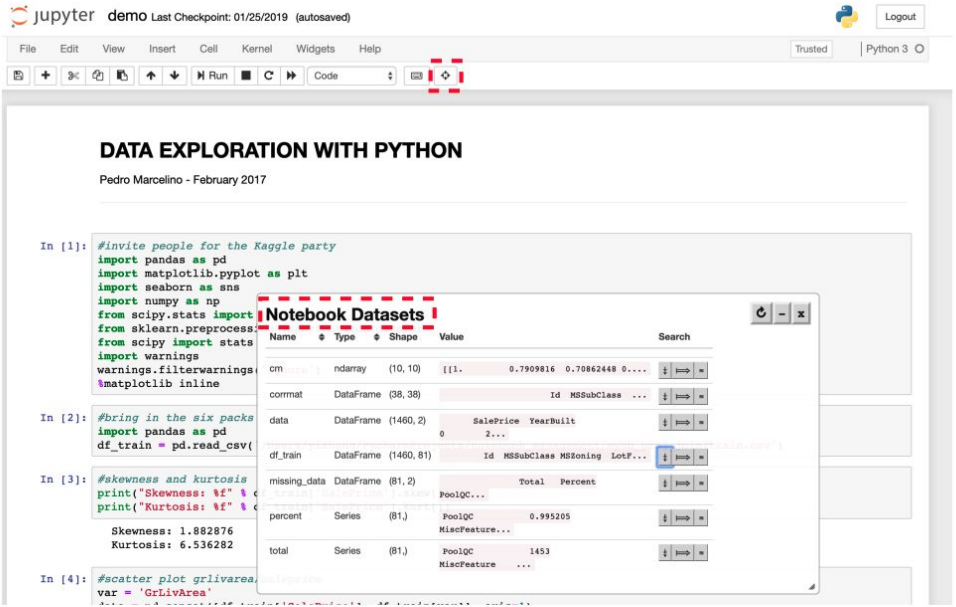


Figure 2: Demo Step 1. The user clicks on the Show Notebook Datasets button in the tool bar, and JUNEAU lists all datasets (tables) in the current notebook.

Demo

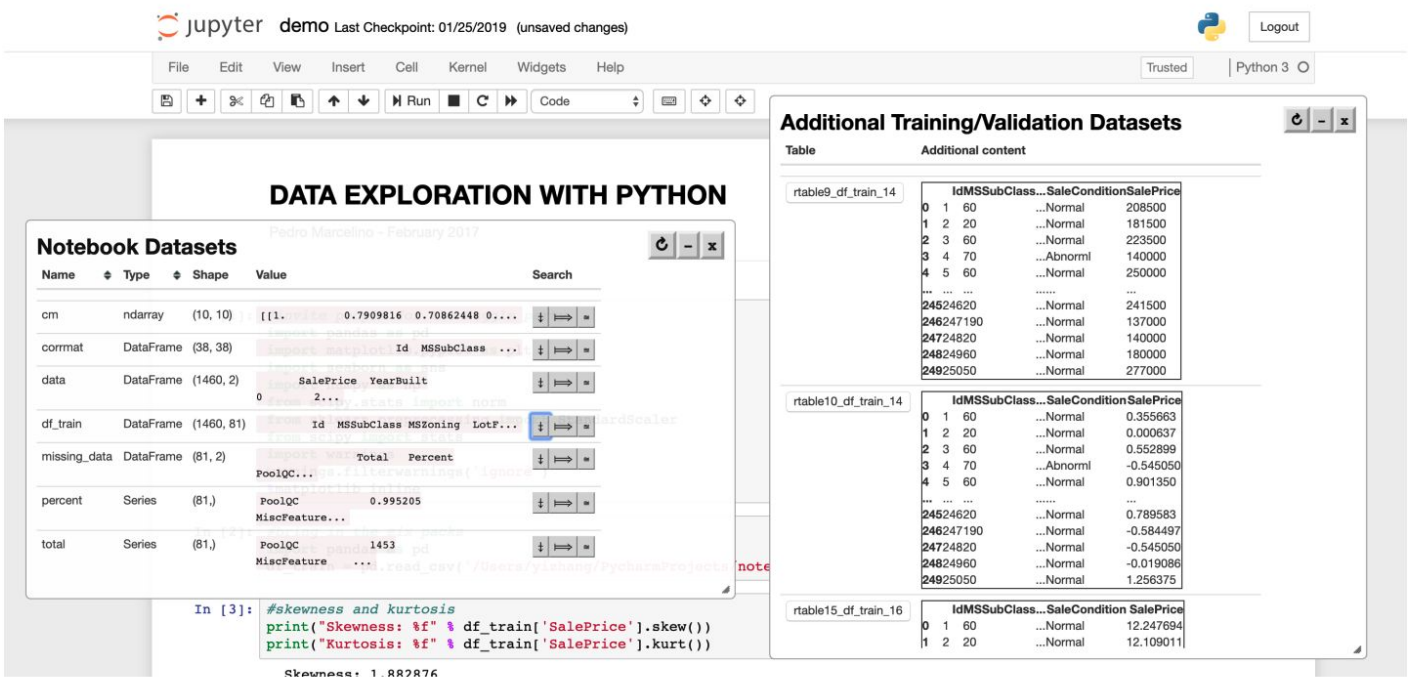


Figure 3: Demo Step 2. The user selects a table with a search mode, and the system will rapidly return a ranked list of tables that are related. In this example, the user selects "D", which means the user is looking for augmenting training or validation data.

Demo

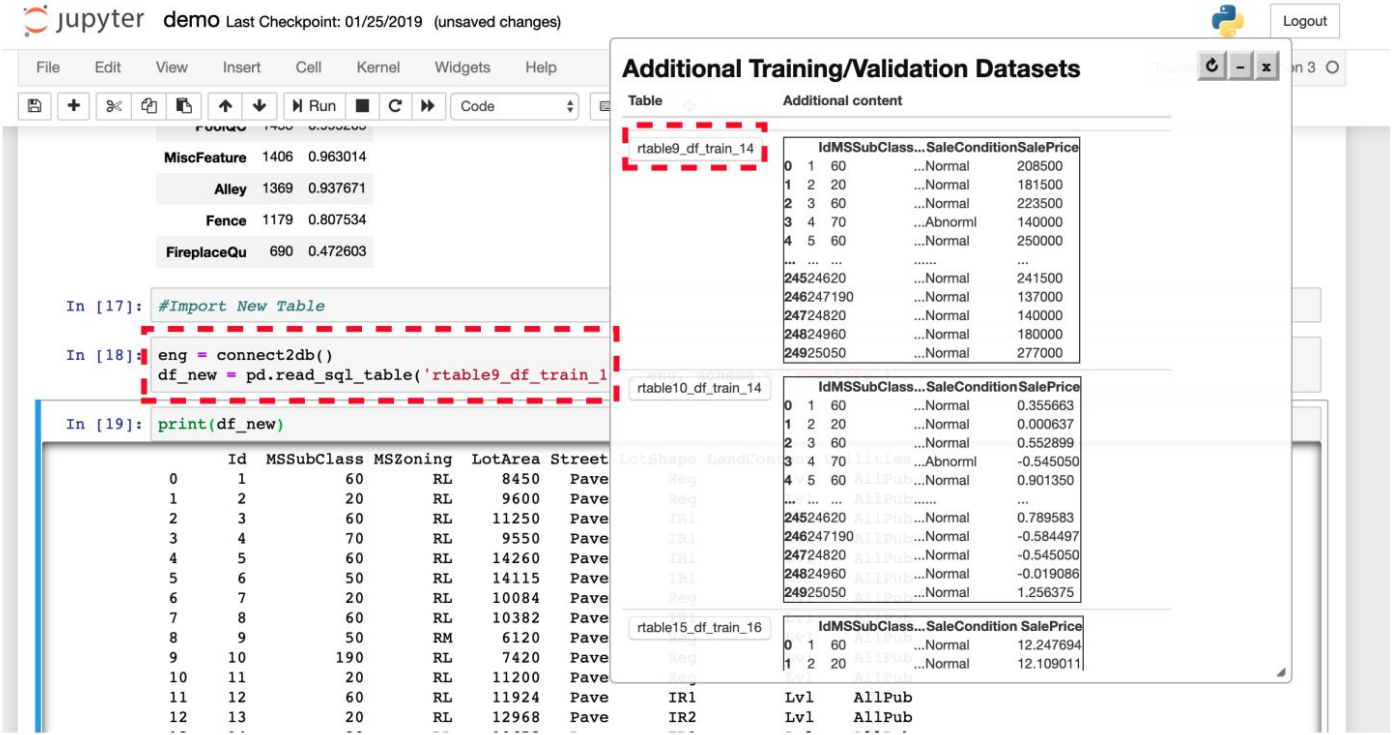


Figure 5: Demo Step 3. The user clicks on a table of interest, and JUNEAU adds a new cell to the notebook that loads the table.

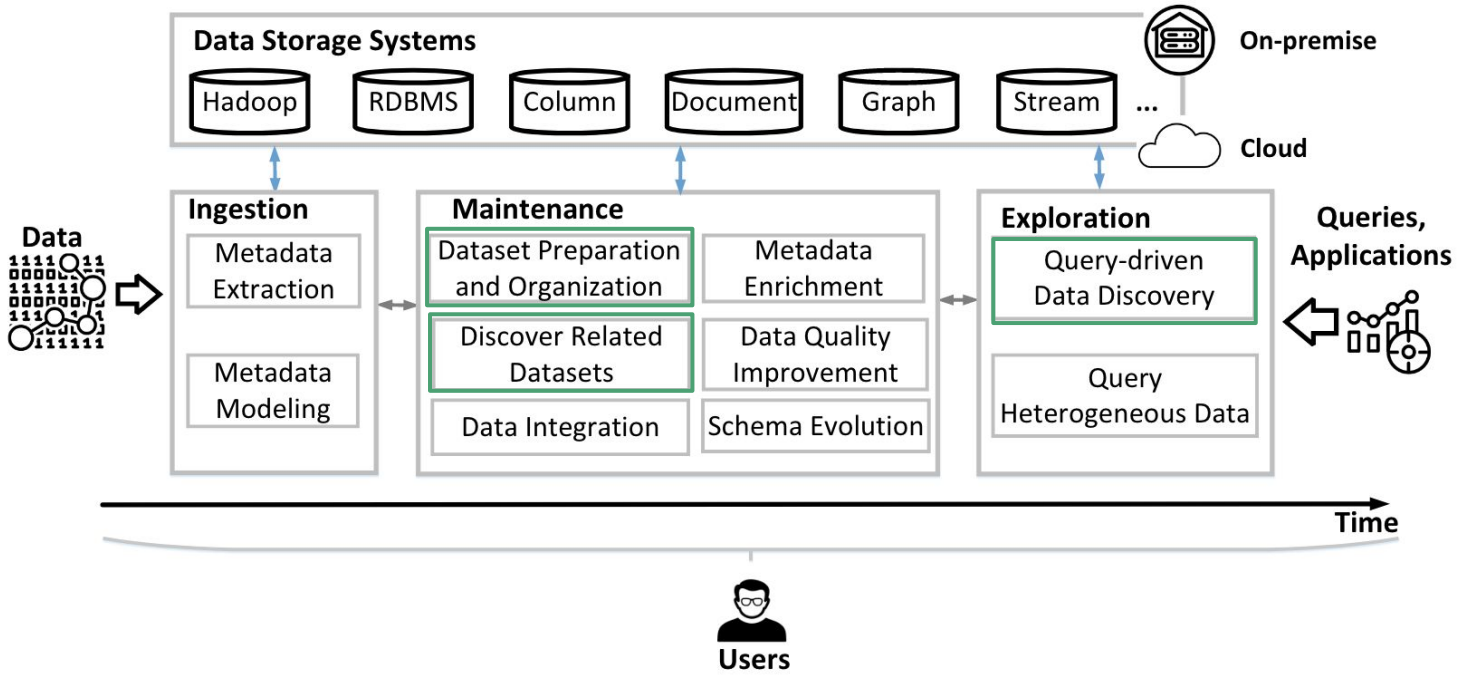
Finding Related Tables in Data Lakes for Interactive Data Science

From the archaeologist’s perspective

Yanhao Wang

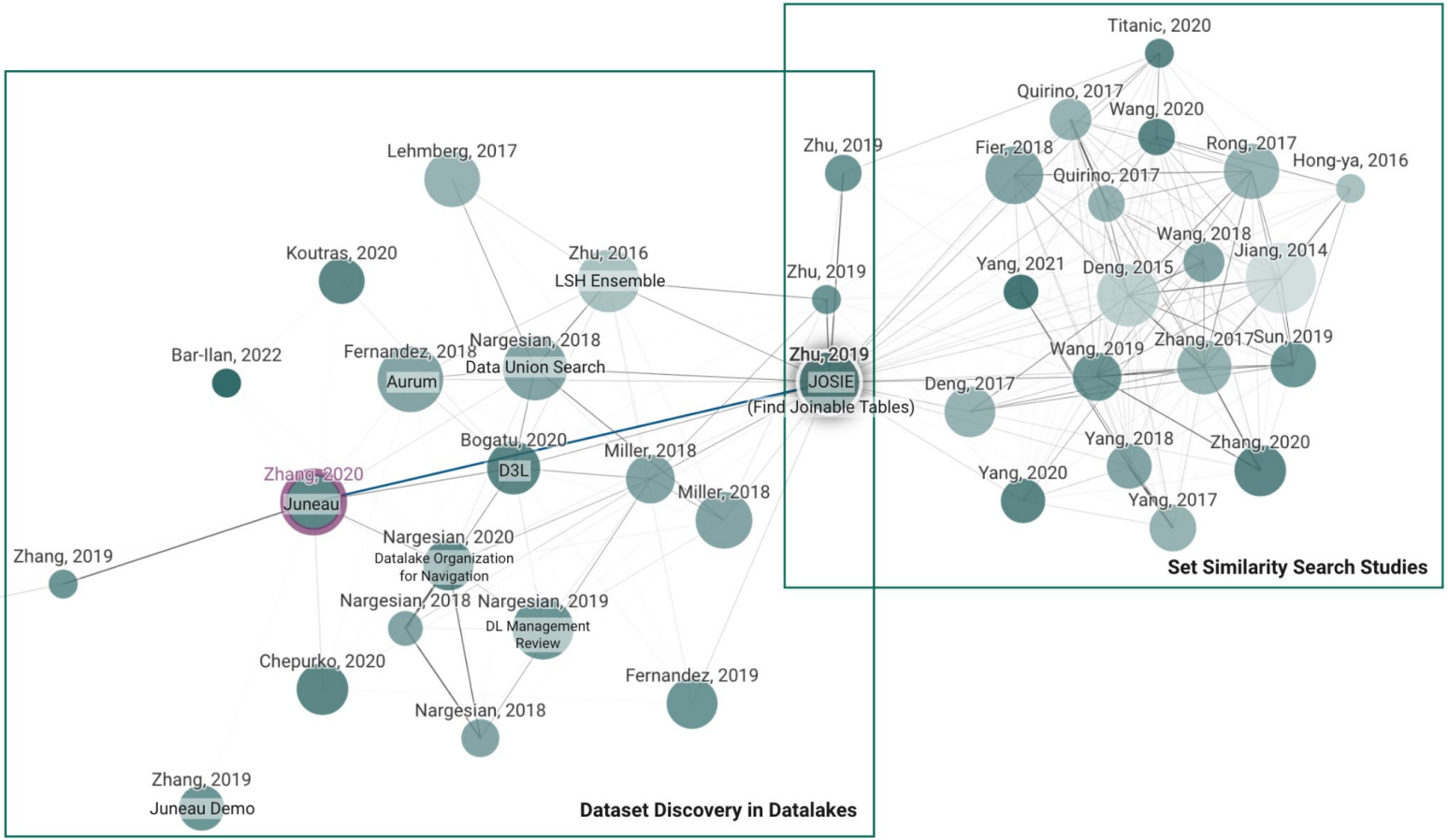
The problem space: Data Lake Systems

schema-on-use



Hai, Rihan, Christoph Quix, and Matthias Jarke. "Data lake concept and systems: a survey." arXiv preprint arXiv:2106.09592 (2021).

The problem space: Related Dataset Discovery

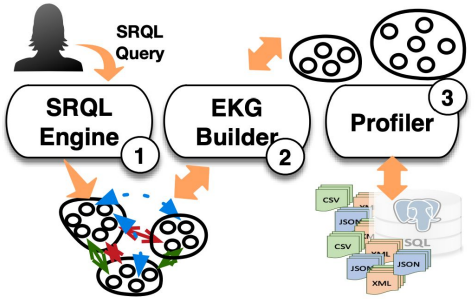


JOSIE [Zhu, 2019]

- Problem: Joinable Table Discovery
  - Given (T, C), return top-K tables that can be joined with T on C
- Focus on data-value overlap
- New perspective: Consider the table columns as sets, and same tuple values as the set intersection, and formalize the problem as ***overlap set similarity*** **problem.**
- Apply existing overlap set similarity search techniques in data lakes, resolving the unique problems in this scenario:
  - Large number of tables in a massive data lake
  - Hard to decide the threshold for the intersection size threshold

Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In Proceedings of the 2019 International Conference on Management of Data. ACM, 847–864.

Aurum [Fernandez, 2018]



- Problem: Joinable Table Discovery
- Indexing using data profiles and sketches
- Use **hypergraphs** to find similar datasets, where nodes are columns
  - Profiles each column with a signature f(card., data distr., min-hash, ...)
  - Index signatures in *Locality-Sensitive Hashing (LSH)*

    hash functions with high collision probability for similar inputs
    E.g. MinHash, random projections, etc.
  - Assign a weighted edge between two column (nodes) if their index fall into the same bucket, with the similarity value as the edge weight
  - Construct hyperedges based on these weighted edges
- Fast and robust against data value changes

Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A data discovery system. In 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE, 1001–1012.

D3L [Bogatu, 2020]

- Problem: Related Table Discovery
  - Given T, return top-K tables that contain relevant attributes for populating T
- Extended types of similarity computed at schema- and instance-level
  - Representation patterns of instance values
  - Semantics of textual attributes
  - Data distribution for numerical values
- Map these features (after LSH) to a 5-d Euclidean space for calculating relatedness

A. Bogatu, A. A. A. Fernandes, N. W. Paton, and N. Konstantinou. Dataset Discovery in Data Lakes. In ICDE, pages 709–720. IEEE, 2020

Juneau [Zhang, 2020]

- Problem: (Task-specific) Related Table Discovery
  - Given (T, τ), return top-K tables that are most relevant to T in terms of search type τ
- Additionally consider the **context** (table provenance) and **intent** (task) of the user
- **Further extended the notion of relatedness**, on top of the often-used value overlap and attribute overlap:
  - New attribute/instance rate
  - Provenance similarity
  - Description similarity
- Introduced workflow graph & variable dependency graph as auxiliary structures to support provenance similarity computation
- Proposed a practical framework to **integrate a subset of relatedness metrics** to address different data science tasks

Zhang, Yi, and Zachary G. Ives. "Finding related tables in data lakes for interactive data science." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

Query-driven Data Discovery

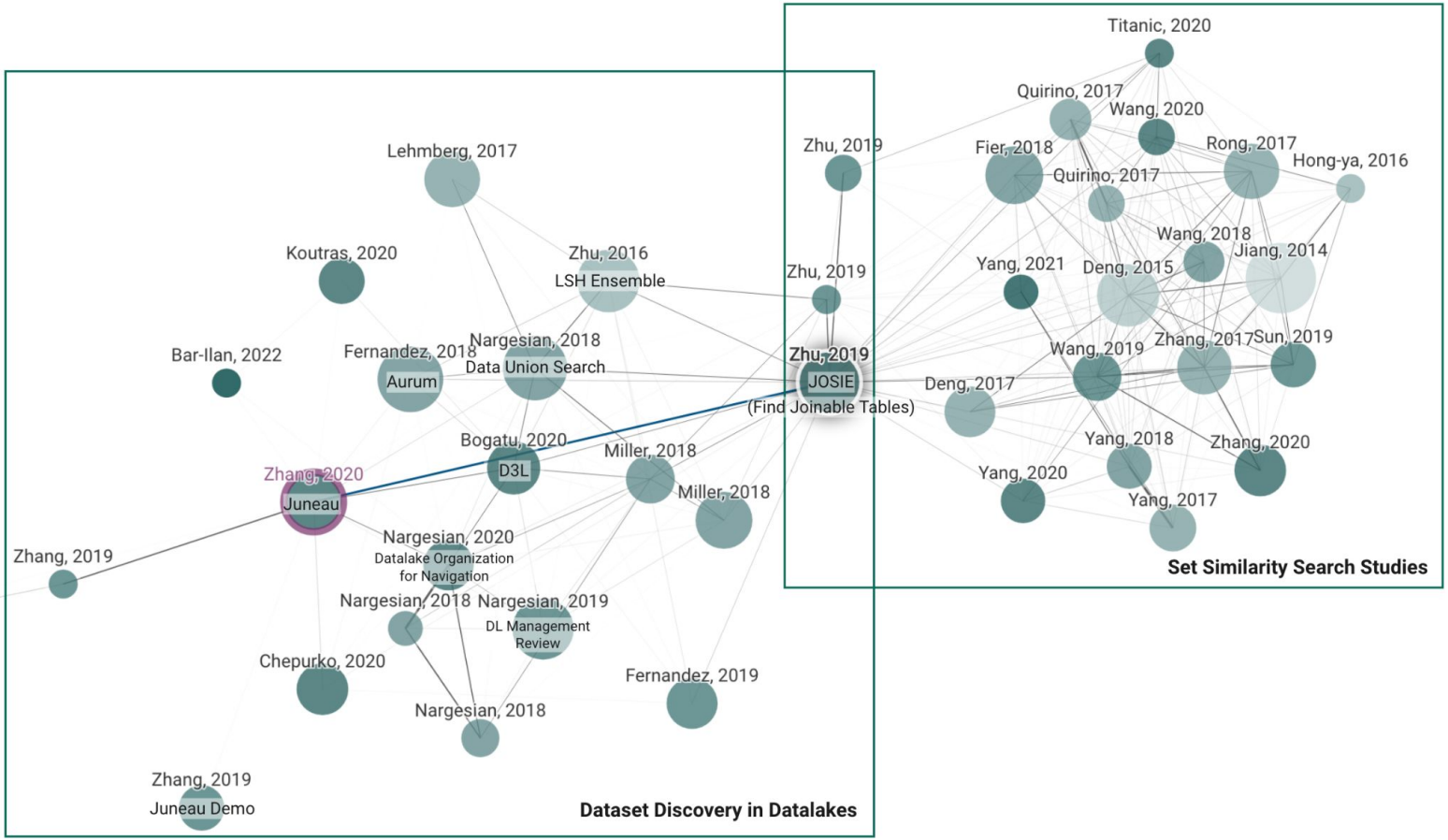
Discover Related Datasets

Dataset Preparation and Organization

Systems	Target Problem	Relatedness Criteria	Similarity metrics	Auxiliary structure
Aurum	Joinable Table Discovery	Instance value overlap Attribute name Primary-key/foreign-key candidate	Jaccard similarity (MinHash) Cosine similarity (TF-IDF)	Hypergraph
JOSIE	Joinable Table Discovery	Instance value overlap	Intersection size of sets	Inverted Index
D3L	Related Table Discovery	Instance value overlap Attribute name Semantics Data value representation pattern (Numerical) data distribution	Jaccard similarity (MinHash) Cosine similarity (Random projections)	5–dimensional Euclidean space
Juneau	(Task-specific) Related Table Discovery	Instance value overlap Domain overlap Attribute name Key constraint New attributes rate New instance rate Variable dependency Descriptive metadata Null Values	Jaccard similarity	Workflow graph Variable dependency graph

Hai, Rihan, Christoph Quix, and Matthias Jarke. "Data lake concept and systems: a survey." arXiv preprint arXiv:2106.09592 (2021).

The problem space: Related Dataset Discovery



Edmint [Amsterdamer, 2022]

- Problem: Given a dataset T, automatically select multiple datasets for extension
  - The value of each potential dataset is dependent on the choice of other datasets
- One important metric is still relatedness, and the author uses the relatedness metrics proposed by [Zhang, 2020] to integrate relatedness from different dimensions
- Not only consider the relatedness from the search table, but also balancing:
  - The integration gain: number of acquired table cells that are truthful and correctly integrated
  - The integration cost: decrease in quality, increase of incompleteness

Amsterdamer, Y., & Cohen, M. (2021). Automated Selection of Multiple Datasets for Extension by Integration. Proceedings of the 30th ACM International Conference on Information & Knowledge Management.

Finding Related Tables in Data Lakes for Interactive Data Science

Reviewer Role : Vishnu K Krishnan

Georgia Tech

Paper Summary

- What is the paper about? –
  - Developed framework Juneau to support search for semantically related tables.
  - Focused towards data science use-cases – cleaning, augmenting, etc.
- What does it solve? –
  - It works to aid data scientists through promotion of reuse of data/workflows, consistency in processing and reduce redundant and inefficient work.
- How It functions? -
  - Usage of top-k, pruning and approximation strategies to return the most relatable tables

Georgia Tech

Strong Points

- Identified the bottleneck while doing top-k search – relational mapping and pruned the candidate tables effectively to Improve the top-k performance.
- Threshold algorithm's on the fly calculation to save on precomputation costs.
- Novel method of profiling tables using various overlapping methods and definitions. Provided clear formal definitions.
- Proposed workaround for the NP-hard subset selection problem by limiting a variable to a small integer.(choosing key mapping to ensure maximum overlap – small key mapping size allows limit)
- Usage of Data profiles, compositional profiles and workflow graphs to speed up computation.
- Utilized Workflow graphs as directed bi-partite graphs for Juneau computational notebook software. Computational notebook platforms directly capture workflows and data exploration by interleaving source code cells with their outputs.

Opportunities of Improvement

- More representations, more data points and performance figures could have been provided.
- Can it be used in the industry? Privacy issues – for example, if this were to be utilized by industry practitioners where some data could be considered confidential, can the current solution be augmented to ensure privacy of the data?
- Trade-off precision for speed using approximation in the threshold algorithm. (early stop used)

Georgia Tech

Verdict?

Accept

Data Lake vs Data Warehouse

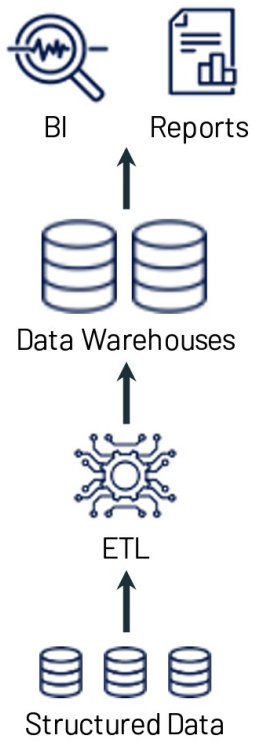
Data lake:

- raw data, can be unstructured
- low-cost storage, but no transactions, data quality checks
- data scientists and engineers

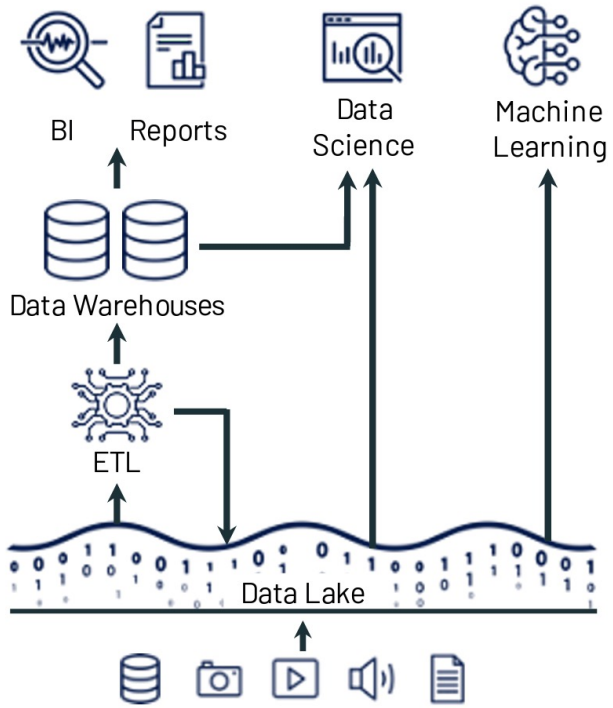
Data warehouse:

- structured data (schema-on-write)
- expensive for large data volumes
- managers and business analysts

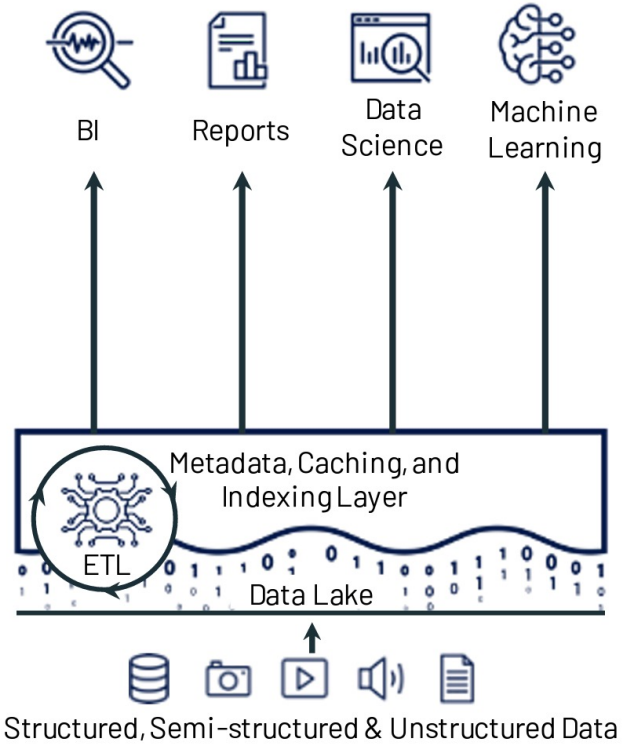
“Lakehouse”



(a) First-generation platforms.



(b) Current two-tier architectures.



(c) Lakehouse platforms.

Source: [Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics]

Contribution/Strengths

Novel problem formulation driven by use cases

- Query by table framework - they hypothesized that interactive DS users search on data lake not only by keyword but also by table.
- More prior work on table search focus on the web
- Despite the difference, the searching for all four tasks can be captured under a general framework and solved by Juneau. This application-driven reformulation of the table searching problem is novel and inspiring.
- The authors exploited the characteristics of different use cases (e.g., data augmentation vs feature extraction) to establish measures that capture the relatedness of tables.

Contribution/Strengths

Implementation and evaluation

- Integrated with Jupyter Notebook (among others) - makes adoption easy
- It leverages multiple indices to perform staged relation mapping detection to speed up the topk search: data profile index on columns, compositional profile index on co-occurring sets of columns, and workflow graph index.
- Juneau evaluates the results on objective measures, for example, the precision of the classifier on the original table vs the precision on the table returned by Juneau.
- Evaluated using real data science workflows from kaggle.com.

Limitation/Weaknesses

Lack of user perception

- Given that targeted usage scenarios are the highlight of the table relatedness measure (thus usefulness of the result), testing out if users actually define related tables in the same way as the relatedness measure is crucial
- No qualitative measures with data scientists - How do actual data scientists interpret the search results? Do they feel comfortable integrating search results into their data pipelines? Are there times that they like the results but they don’t perform well, and vice versa?

Limitation/Weaknesses

Unclear generalization

- All the metrics and searching methods are specifically designed for these mappings, it remains unclear how easy the proposed system generalizes to new settings.
- So, the query is based on a table and a specific task at hand, what if I wish to perform task 1 followed by task 3 maybe the query can combine two tasks and return the best-suited table for this.
- Assumption that cells in a notebook are in sequence is too optimistic. Notebooks are designed for random and unstructured runs, so assuming the order is a big assumption.

Limitation/Weaknesses

Weak baseline

- In the experiments, only a brute-force baseline is compared for efficiency. There is no state-of-the-art solution for finding similar joinable or unionable tables
- The experiments section does not sufficiently compare against other works - the authors mention several related approaches like Google Dataset search, q system, etc.
- The experiments do not conclusively show that Juneau is superior to keywork search. Although Juneau has slightly higher accuracy, its query times are much longer than keyword search.
- The corpus used in the evaluation consists of data ~5GB. This is pretty small in comparison to the sizes of data stored on data lakes in the real world, which might make the results biased.

Limitation/Weaknesses

Data quality issues

- Sometimes people leave trails of abandoned methods that are just not correct, outdated, or contain bugs. Storing and indexing these seems wasteful, and can also lead to poor/misleading search results.
- I wonder how they have handled different versions of the data - are all available or just the latest one?
- Data cleaning and augmentation can be quite task specific, but the search doesn't explicitly encode the task, and incorrect methods can slip in.

Next class

[Auto-Suggest: Learning-to-Recommend Data Preparation Steps Using Data Science Notebooks]

Author: Bojun, Siddhi

Reviewer: Shubham, Shen En

Archaeologist: Aniruddha

Practioner: Jingfan

Researcher: Ting