Developing Open Source Software in Academia for Data Integration: Experience and Lessons Learned

AnHai Doan
University of Wisconsin-Madison

Joint work with many others
Data Integration

- Combines disparate data into an unified database
- Must solve many problems
  - data acquisition, extraction, exploring, profiling, transforming, cleaning, schema matching/merging, entity matching/merging, etc.

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Apple</td>
<td>CA</td>
</tr>
<tr>
<td>x2</td>
<td>IBM</td>
<td>NY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>loc</th>
<th>rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Inc</td>
<td>CA</td>
<td>51</td>
</tr>
<tr>
<td>IBM Corp</td>
<td>NY</td>
<td>25</td>
</tr>
<tr>
<td>GE</td>
<td></td>
<td>35</td>
</tr>
</tbody>
</table>

id | cname     | address | rev |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>IBM Corp</td>
<td>CA</td>
<td>25</td>
</tr>
<tr>
<td>y2</td>
<td>Apple Inc</td>
<td>CA</td>
<td>51</td>
</tr>
<tr>
<td>y3</td>
<td>GE</td>
<td>NY</td>
<td>351</td>
</tr>
</tbody>
</table>
```
Long-Standing Challenge

- Numerous works in the past 30 years
- In many communities: DB, AI, KDD, Web, Semantic Web
Will Become Even More Critical in the Age of Data Science

- The data science pipeline

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>Apple</td>
<td>CA</td>
</tr>
<tr>
<td>x_2</td>
<td>IBM</td>
<td>NY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>loc</th>
<th>rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Inc</td>
<td>CA</td>
<td>51</td>
</tr>
<tr>
<td>IBM Corp</td>
<td>NY</td>
<td>25</td>
</tr>
<tr>
<td>GE</td>
<td></td>
<td>35</td>
</tr>
</tbody>
</table>

is there any correlation between location and revenue?

Data integration, preparation, wrangling, …
Will Become Even More Critical in the Age of Data Science

- Where data scientists spend **80%** of their effort
- To ensure high quality data
  - otherwise “garbage in, garbage out”

Data scientists spend 50 - 80% of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.
Yet the Field Appears Stagnating

- Focuses mostly on algorithmic solutions
  - ever more complex, but are they useful in practice?

- Builds isolated system prototypes
  - hard to use and combine
  - too weak for real-world applications

- Not sure what to teach our students
  - teaching isolated system prototypes?
  - can’t use them after graduation

- Not sure how to help real users (e.g., domain scientists)
  - they need to solve end-to-end DI tasks
  - cannot tell them how to start, what algorithms to consider, what systems to use, what they need to manually do in each step
Our Proposal: Devote Far More Effort to Building DI Systems

- Most current works develop algorithmic solutions

- But DI is engineering by nature
  - Can’t just develop algorithms in vacuum
  - Like developing join algorithms without building rest of RDBMS
  - Must build real systems for real users,
    to evaluate algorithms, integrate R&D efforts, make practical impacts

- Can learn from the fields of RDBMSs and Big Data
  - Where pioneering systems significantly pushed the field forward
What Kinds of System to Build?

There is Already an Existing DI System Out There
The PyData Ecosystem of Data Science Tools

- 138,000+ interoperable open-source Python packages
  - from “only” 86K packages 2 years ago

- Many popular packages, lots of users
  - Anaconda: 536 packages, 4.5M users

- Strong focus on interoperability & ease of use

- Our proposal
  - do not build isolated monolithic DI systems
  - extend this PyData “system”
  - develop packages that can interoperate and be easily combined to solve DI problems for PyData users
Extending PyData Brings Many Benefits

- Can instantly leverage many capabilities of PyData
  - avoid building weak research prototypes
- Can better convince many users to work with our tools
  - many domain scientists, company teams already use PyData
- Can teach PyData and our tools seamlessly in classes
- Developing packages is much easier to do in academia
  - so our students have better chance of developing useful software
- Tools by different DI groups can interoperate
- PyData is currently very weak in DI aspects
  - so we can make real contributions
- If DI community focuses on a single system, we can better measure whether we are making good progress
  - just like in the RDBMS days
Our Agenda

- **Build tools for core DI problems**
  - entity matching, schema matching, extraction, cleaning, etc.
  - focus on helping power users

- **Extend to the lay user/collaborative/cloud settings**

- **Foster PyDI, an ecosystem of interoperable DI tools**
  - as part of PyData

Started the Magellan project in 2015
Focused on entity matching
## Entity Matching

### Table A

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dave Smith</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>Joe Wilson</td>
<td>San Jose</td>
<td>CA</td>
</tr>
<tr>
<td>Dan Smith</td>
<td>Middleton</td>
<td>WI</td>
</tr>
</tbody>
</table>

### Table B

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>David D. Smith</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>Daniel W. Smith</td>
<td>Middleton</td>
<td>WI</td>
</tr>
</tbody>
</table>

![Diagram of entity matching]
Blocking and Matching

Table A

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dave Smith</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>Joe Wilson</td>
<td>San Jose</td>
<td>CA</td>
</tr>
<tr>
<td>Dan Smith</td>
<td>Middleton</td>
<td>WI</td>
</tr>
</tbody>
</table>

Table B

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>David D. Smith</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>Daniel W. Smith</td>
<td>Middleton</td>
<td>WI</td>
</tr>
</tbody>
</table>

Block on state = state → match

\[(a_1, b_1) + (a_1, b_2) - (a_3, b_1) - (a_3, b_2) +\]
Build Tools for Core DI Problems

- **Identify a concrete DI problem**
  - entity matching, schema matching, etc.
  - solve it end-to-end: from raw data to a desired outcome for end user
    - e.g., EM with at least 95% accuracy
  - solve it for **power users**: those who can code

- **Develop a how-to guide**
  - most end-to-end DI problems cannot be fully automated, so user must be in the loop ➔ what should he/she do?
  - how-to guide tells user how to solve the problem step by step
  - it’s a detailed algorithm for the human user
  - if user can code, he/she can follow the guide to solve the problem

- **Identify pain points in the guide**

- **Develop tools for pain points**
  - tools are packages in PyData ecosystem
Example Problem & How-To Guide

- **Development stage**
  - finds an *accurate* workflow, using *data samples*

- **Production stage**
  - executes workflow on *entirety of data*
  - focuses on *scalability*

1M tuples

A

1M tuples

B

block → match
(using machine learning)
select a good blocker: blocker X, blocker Y

select a good matcher: matcher U, matcher V
Guide for the Production Stage

Scaling, quality monitoring, exception handling, crash recovery, …
Example Pain Points & Tools

**Select a good blocker**

- **A’**
  - **A’** → **blocker** → **C_x**
  - **B’**
  - **B’** → **blocker** → **C_y**

**How to debug a blocker?**

**How to sample and label?**

**Select a good matcher**

- **C_x**
  - (-,-)
  - (-,-)
  - (-,-)
  - (-,-)
  - sample

**Cross-validate matcher U**

- 0.89 F_1
- 0.93 F_1

**Cross-validate matcher V**

- (-,-) +
- (-,-) -
- (-,-) +

**How to debug a matcher?**

- **A’**
  - **A’** → **blocker** → **C_x**
  - **B’**
  - **B’** → **blocker** → **C_y**

**Sample**

- **A**
  - **A’**
  - **B’** → **B’**

**How to sample and label?**

- **0.89 F_1**
- **0.93 F_1**

**Quality check**

- **yes**
  - **no**

**Label**

- **S**
  - (-,-)
  - (-,-)

**Cross-validate matcher V**

- (-,-) +
- (-,-) -
- (-,-) +
Example Pain Points & Tools

- Tool to highlight possible matching categories
- Tool to debug labels
- Tool to help revise labels

([Laura’s, 23 Farewell Str], [Laura, 23 Farewell]) +
([Palmyra, 46 Main St], [Palmyra, 15 Broadway]) -
([KFC, 24 Main St], [KFC, 41 Johnson Ave]) +
Proposed System Architecture

Development stage

An EM scenario, e.g., matching Tables A, B

Problem definition
Data profile
Tool profile

How-to guide
Power user

Step 1 → Step 2 → ... → Step n

EM requirements (accuracy, runtime, etc.)

Production stage

Tables A, B

EM workflow (procedural)

Executor

Matches

Optimizer

EM plans ← EM operators

EM workflow (declarative)

Tools as packages in the Python data analysis stack

Tools as packages in the Python big data stack
Many Technical/Practical Challenges

- Exploring match definitions
- Exploring data
- Visualization
- Helping users build blockers and matchers
- Labeling data
- Quickly building a basic EM workflow
- Debugging blockers and matchers
- Data cleaning in an EM-centric way
- Provenance
- Explaining
- …

We solve the challenges, refine the guide, build tools
Agenda

- Build tools for core DI problems
  - entity matching, schema matching, extraction, cleaning, etc.
  - focus on helping power users
- Extend to the lay user/collaborative/cloud settings
- Foster PyDI, an ecosystem of DI tools
  - as part of PyData
String Matching for Lay Users

- A special case of EM

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>Michael J. Williams</td>
<td>(a₁, b₁)</td>
</tr>
<tr>
<td>a₂</td>
<td>Michael J. Smith</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>Chen Y. Li</td>
<td>(a₃, b₂)</td>
</tr>
<tr>
<td>b₁</td>
<td>Williams, Michael</td>
<td></td>
</tr>
<tr>
<td>b₂</td>
<td>Li, Chen</td>
<td></td>
</tr>
</tbody>
</table>

- Common operation
  in matching, cleaning, exploration, profiling, etc.
Current String Matching Solutions

- Define match condition using string similarity measures
  - such as Jaccard, edit distance

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>Williams, Michael</td>
</tr>
<tr>
<td>a₂</td>
<td>Li, Chen</td>
</tr>
<tr>
<td>a₃</td>
<td>Chen Y. Li</td>
</tr>
<tr>
<td>Michael J. Williams</td>
<td>b₁</td>
</tr>
<tr>
<td>Michael J. Smith</td>
<td>b₂</td>
</tr>
<tr>
<td>Chen Y. Li</td>
<td></td>
</tr>
</tbody>
</table>

\[ J_{\text{Jaccard}}(\text{word}(a), \text{word}(b)) \geq 0.7 \]

\[ X = \{“Chen”, “Y”, “Li”\} \quad Y = \{“Li”, “Chen”\} \]

\[ J_{\text{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{2}{3} \]

- Efficiently execute using index-based filters

<table>
<thead>
<tr>
<th>word</th>
<th>ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael</td>
<td>a₁, a₂</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Chen</td>
<td>a₃</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

String from B

- b₂: Li, Chen
- Potential matching strings
- a₃: Chen Y. Li
### Very Difficult for Lay Users

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>Michael J. Williams</td>
</tr>
<tr>
<td>a₂</td>
<td>Michael J. Smith</td>
</tr>
<tr>
<td>a₃</td>
<td>Chen Y. Li</td>
</tr>
<tr>
<td>b₁</td>
<td>Williams, Michael</td>
</tr>
<tr>
<td>b₂</td>
<td>Li, Chen</td>
</tr>
</tbody>
</table>

\[
\text{Jaccard(word(a), word(b))} \geq 0.7
\]

- Jaccard, Dice, edit distance, ...
- 0.3, 0.5, 0.7, 0.9, ...
- word, 3gram, 2gram, ...
Our Solution: Smurf (Self-Service)

- Lay users only need to label string pairs
  - took less than 30 mins in our experiments

- Consider random forest as matching condition
  - achieves much higher accuracy
  - 65.03 – 98.43 $F_1$ vs 54.58 – 88.01 $F_1$
Background: Random Forest

- Ensemble classifier containing a set of decision trees

Return the label output by the majority of trees
Learning the Matching Condition

A → Sampling → Sample S → Extract features → Feature vectors S' → Active learning → Random forest F

Feature library

can stop? → yes → RF

no → select “informative” unlabeled pairs

label selected pairs

A → b c d e → (a, e) → (0.3, …, 0.6) → train RF

A → a b c → (b, d) → (0.0, …, 0.4) → train RF

B → d e → (b, e) → (0.4, …, 0.8)
Executing the Matching Condition

Random forest

Learn matching condition

Execute matching condition

Predicted matches
Executing the Matching Condition

\[ \text{execute } t_1 \quad \text{execute } t_2 \quad \text{execute } t_3 \]

- \( \text{edit\_distance} < 3 \)
- \( \text{dice\_3gram} > 0.8 \) N \quad N \quad N
- \( \text{dice\_3gram} > 0.6 \) Y \quad Y \quad Y
- \( \text{non-match} \) \quad \text{match} \) \quad \text{non-match} \) \quad \text{match} \) \quad \text{match} \)

\[ (C_1 \cap C_2) \cup (C_1 \cap C_3) \cup (C_2 \cap C_3) \]

\( t_1 \quad t_2 \quad t_3 \)
CloudMatcher: A Similar Solution for EM

Users: Analyst, Underwriter, Agent, etc.

Table A, B

- Trained models
- Sampled data
- Predicted matches
- ....
Example: Matching Tables A & B

**Table A**

Sample rows

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>addr</th>
<th>city</th>
<th>phone</th>
<th>type</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>534</td>
<td>'amie morton's of chicago'</td>
<td>'435 s. la clanege blv.'</td>
<td>'los angeles'</td>
<td>310/246-1501</td>
<td>american</td>
<td>0</td>
</tr>
<tr>
<td>535</td>
<td>'art's delicatessen'</td>
<td>'12224 ventura blvd.'</td>
<td>'studio city'</td>
<td>818/762-1221</td>
<td>american</td>
<td>1</td>
</tr>
<tr>
<td>536</td>
<td>'hotel bel-air'</td>
<td>'701 stone canyon rd.'</td>
<td>'bel air'</td>
<td>310/472-1211</td>
<td>californian</td>
<td>2</td>
</tr>
<tr>
<td>537</td>
<td>'cafe bizco'</td>
<td>'14015 ventura blvd.'</td>
<td>'sherman oaks'</td>
<td>818/788-3536</td>
<td>french</td>
<td>3</td>
</tr>
<tr>
<td>536</td>
<td>campenile</td>
<td>'624 s. la brea ave.'</td>
<td>'los angeles'</td>
<td>213/936-1447</td>
<td>american</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table B**

Sample rows

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>addr</th>
<th>city</th>
<th>phone</th>
<th>type</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'apple pan the'</td>
<td>'10601 w. pico blvd.'</td>
<td>'west la'</td>
<td>310-475-3585</td>
<td>american</td>
<td>534</td>
</tr>
<tr>
<td>2</td>
<td>'asahi ramen'</td>
<td>'2027 sawtelle blvd.'</td>
<td>'west la'</td>
<td>310-479-2231</td>
<td>'noodle shops'</td>
<td>535</td>
</tr>
<tr>
<td>3</td>
<td>'baja fresh'</td>
<td>'3345 kimber dr.'</td>
<td>'westlake village'</td>
<td>805-498-4049</td>
<td>mexican</td>
<td>536</td>
</tr>
<tr>
<td>4</td>
<td>'belvedere the'</td>
<td>'9882 little santa monica blvd.'</td>
<td>'beverly hills'</td>
<td>310-788-2305</td>
<td>'pacific new wave'</td>
<td>537</td>
</tr>
<tr>
<td>5</td>
<td>'bental's frites'</td>
<td>'1433 third st. promenade'</td>
<td>'santa monica'</td>
<td>310-458-2889</td>
<td>'fast food'</td>
<td>538</td>
</tr>
</tbody>
</table>
## Labeling for Active Learning

### Train Model

Do these pairs refer to the same real world entity?

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>addr</th>
<th>city</th>
<th>phone</th>
<th>type</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>550</td>
<td>patina</td>
<td>'5955 melrose ave.'</td>
<td>'los angeles'</td>
<td>213/467-1106</td>
<td>Californian</td>
<td>16</td>
</tr>
<tr>
<td>235</td>
<td>patina</td>
<td>'5955 melrose ave.'</td>
<td>'los angeles'</td>
<td>213-467-1106</td>
<td>Californian</td>
<td>16</td>
</tr>
<tr>
<td>555</td>
<td>valentino</td>
<td>'3115 pico blvd.'</td>
<td>'santa monica'</td>
<td>310/829-4313</td>
<td>Italian</td>
<td>21</td>
</tr>
<tr>
<td>240</td>
<td>valentino</td>
<td>'3115 pico blvd.'</td>
<td>'santa monica'</td>
<td>310-829-4313</td>
<td>Italian</td>
<td>21</td>
</tr>
<tr>
<td>875</td>
<td>'sammy's romanian steak house'</td>
<td>'157 chrystie st. at delancey st.'</td>
<td>'new york'</td>
<td>212/673-0330</td>
<td>'east european'</td>
<td>341</td>
</tr>
<tr>
<td>108</td>
<td>'sparks steak house'</td>
<td>'210 e. 40th st.'</td>
<td>'new york city'</td>
<td>212-607-4655</td>
<td>Steakhouses</td>
<td>041</td>
</tr>
<tr>
<td>962</td>
<td>'binion's coffee shop'</td>
<td>'128 fremont st.'</td>
<td>'las vegas'</td>
<td>702/382-1600</td>
<td>'coffee shops/diners'</td>
<td>428</td>
</tr>
<tr>
<td>9</td>
<td>'brighton coffee shop'</td>
<td>'9500 brighton way'</td>
<td>'beverly hills'</td>
<td>310-276-7732</td>
<td>'coffee shops'</td>
<td>542</td>
</tr>
<tr>
<td>619</td>
<td>'la grotta'</td>
<td>'2837 peachtree rd. peachtree house condominium'</td>
<td>atlanta</td>
<td>404/231-1368</td>
<td>Italian</td>
<td>85</td>
</tr>
<tr>
<td>304</td>
<td>'la grotta'</td>
<td>'2537 peachtree rd. ne'</td>
<td>atlanta</td>
<td>404-231-1368</td>
<td>Italian</td>
<td>85</td>
</tr>
</tbody>
</table>
Returning Matches Between Two Tables

Sample matching pairs

<table>
<thead>
<tr>
<th>Table A id</th>
<th>Table B id</th>
</tr>
</thead>
<tbody>
<tr>
<td>534</td>
<td>219</td>
</tr>
<tr>
<td>535</td>
<td>220</td>
</tr>
<tr>
<td>536</td>
<td>221</td>
</tr>
<tr>
<td>537</td>
<td>222</td>
</tr>
<tr>
<td>538</td>
<td>223</td>
</tr>
<tr>
<td>539</td>
<td>224</td>
</tr>
</tbody>
</table>
Agenda

• **Build tools for core DI problems**
  - entity matching, schema matching, extraction, cleaning, etc.
  - focus on helping power users

• **Extend to the lay user/collaborative/cloud settings**

• **Foster PyDI, an ecosystem of DI tools**
  - as part of PyData
What I Believe Will Happen in Next Ten Years

- Ecosystem of tools + how-to guides will be the “system of choice”

- So it is critical to study how to build such ecosystems
  - E.g., how to build PyDI as a part of PyData
How to Build PyDI as Part of PyData

- **Study PyData and similar ecosystems (e.g., R)**
  - What do they do?
  - Why are they so successful?
    - because they address the pain points!
    - tools are designed to interoperate
    - tools are open source and free
    - extensive infrastructure & support
  - What are their challenges and solutions?
  - Any research on how to build such ecosystems?

- **Apply the lessons learned to foster PyDI**
Data Integration and Data Preparation in Python

“Data scientists, according to interviews and expert estimates, spend from 50-80% of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.”

— Steve Lohr, Aug 17, 2014, New York Times (For Big Data Scientists, ‘Janitor Work’ Is Key Hurdle to Insights)

BigGorilla is an open-source data integration and data preparation ecosystem (powered by Python) to enable data scientists to perform integration and analysis of data. BigGorilla consolidates and documents the different steps that are typically taken by data scientists to bring data from different sources into a single database to perform data analysis. For each of these steps, we document existing technologies and also point to desired technologies that could be developed.

- OPEN-SOURCE COMPONENTS FOR DATA INTEGRATION AND DATA PREPARATION
- CAN BE COMBINED AND REUSED IN DIFFERENT CONTEXT AND DIFFERENT WAYS
- TUTORIALS, DATASETS, AND EXAMPLES
Challenges for Building Ecosystems

- Version incompatibility among packages
- Dependency hell
  - someone upgrades NumPy and break our packages
- Managing missing values (MV), MV policy per package
  - how to infer MV policy of a new package?
- How to select data structures to facilitate sharing?
- How to build, manage, use indexes?
- How to manage keys?
- How to manage metadata?
- If metadata is uncertain, this raises query processing over uncertain data and semantic issues
Metadata Challenges

Closed-World Systems

RDBMS

- SQL queries
- commands
- data
  - A
  - B
- metadata
  - A.ssn is a key

Open-World Systems

- Package X
  - command \(x_1\)
  - command \(x_2\)
  - ...
- Package Y
  - command \(y_1\)
  - command \(y_2\)
  - ...
- Magellan
  - command 1
  - command 2
  - ...
- data
  - A
  - B
- metadata
  - A.ssn is a key

PyData ecosystem
## The Magellan Project (Since 2015)

<table>
<thead>
<tr>
<th>System Types</th>
<th>Problem Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>String matching</td>
</tr>
</tbody>
</table>
| On-premise packages| ```py_stringmatching```
|                    | ```py_stringsimjoin```
|                    | ```py_entitymatching```
|                    | ```deepmatcher```
| Micro-cloud services| ```smurf (self-service)```
|                    | ```cloudmatcher (self-service)```
| Macro-cloud services|                         |
|                    |                         |

- `py_stringmatching`
- `py_stringsimjoin`
- `py_entitymatching`
- `deepmatcher`
- `smurf`
Example: Py_EntityMatching

- Has been in development since 2015
  - ~36 months
  - 13 developers
- Includes 6 Python packages
- Built on top of 16 different packages from PyData ecosystem
  - E.g., pandas, scikit-learn, etc.
- Exposes 231 commands to users
- Code base includes 130 Python files and 37K lines of code
- Packages are comprehensively tested
  - 2316 unit test cases
  - 135 performance test cases
- Code base is extensively documented
  - ~15K lines of comments
- Most advanced and comprehensive open-source EM system for power users
Real-World Applications: Companies

- **Walmart Labs**
  - Helped improve a system already in production
  - Increased recall by 34%, while reducing precision by only 0.65%

- **Johnson Controls**
  - Matched hundreds of thousands of suppliers for JCI
  - Precision above 95%, recall above 92% (across many data sets)

- **Marshfield Clinic**
  - Matched 18M pairs of drugs, precision: 99.18% recall: 95.29%
  - Paper and talk by the chief scientist

- **Recruit Holdings**
  - Matched stores, companies and properties
  - Helped to reduce costs and improve accuracy by 4%, press release

- **American Family Insurance**
  - Many data sets, press release by the VP of data science

- **UW Health**
  - Matched doctors across health care surveys, improved recall by 31%
CloudMatcher Deployed at AmFam
Real-World Applications: Domain Sciences

- **Match grant descriptions** *(economics)*
  - Improved recall by 35% compared to a rule-based solution
  - Paper under preparation

- **Match two tables of 1851 and 13.5M organization names** *(economics)*

- **Match attribute names in community data repository** *(environmental science)*

- **Match cattle ranches in Amazon** *(land use science)*
  - Improved recall by 30%, being pushed into production

- **Match attribute names in community data repository** *(biomedicine)*
  - Paper published
Publications in the Database Field

- **Human-in-the-Loop Data Analysis: A Personal Perspective**, A. Doan. *HILDA Workshop @ SIGMOD-18*.
Publications in Domain Sciences and Press Releases


- Recruit's Artificial Intelligence Laboratory Releases BigGorilla: An Open-source Data Integration and Data Preparation Ecosystem (Press Release)

- AMERICAN FAMILY, UW–MADISON TEAM UP IN DATA SCIENCE (PRESS RELEASE)
  - Working with AnHai Doan, a professor of computer science, and his grad students, “we have built and deployed a machine-learning algorithm all across our businesses that has already saved us significant money.” Justin Cruz, VP Strategic Data & Analytics, American Family Insurance
Other Results

- Two packages installed on Kaggle
  - a DS platform of 0.5M+ users

- Used by 400+ ugrad/grad students in 5 DS courses

- SIGMOD Research Highlight Award

- Raised several millions in research funding
  - from UW, companies, NSF

- Startup seed and licensing
Experience & Lessons Learned

- **How-to guides are critical**
  - makes working with real users so much easier
  - can walk them through the problem end to end, step by step

- **Building into PyData is absolutely the right thing to do**
  - makes system building so much easier, can leverage a lot
  - systems get more powerful every day

- **Building packages in academia is highly promising**
  - packages are relatively small and modular
  - so much easier for students to build and maintain
  - I would not want to do anything more complicated
  - no more giant monolithic stand-alone systems
Experience & Lessons Learned

- **Do you need experience in system building to start?**
  - no, just need to believe system building is the right thing to do
  - but would be good if preparing to build systems for 5-10 years
  - this is because it’s very hard to build things right the first time

- **What kind of system to build?**
  - stand-alone, monolithic systems: very hard to do so in academia
  - packages are much easier to build and maintain

- **Our system building philosophy**
  - break into pieces as much as possible
  - forgo certain capabilities
  - try to build pieces that are small yet already useful
    (e.g., labeler, active learner, string matcher)
Experience & Lessons Learned

- **Integrating with education**
  - use our systems in data science courses, test drive with students

- **Fund raising**
  - building systems takes a lot of money, try multiple sources
  - companies: get slush money, get sponsored research money
  - other domain science grants (e.g., NIH)
  - our own NSF grants
  - UW data science initiatives: raised $700K, no overhead

- **Need to engage users**
  - domain scientists
  - companies
  - be willing to solve problems end to end for them
  - how to recruit them?
Experience & Lessons Learned

● Team organization
  – highly transient, how to obtain the most from such a team?
  – this is the biggest challenge I face

  – me at the top
  – several Ph.D. students
  – many M.S. students plus hourly students

  – Ph.D. students write some code as part of their thesis work (need to revise the Ph.D. graduation requirements)
  – M.S. students write code, work out prototypes

  – really need 1-2 programmers, perhaps 1-2 postdocs

● Tech transfer and startup
Conclusions

- Data integration & data science increasingly critical
- Must do more system work for real users
- Proposed a system building agenda
  - start with real-world problem scenario
  - try to develop how-to guide to solve it end-to-end for real users
  - identify pain points, do research, develop tools
  - build tools into PyData data science ecosystem
  - tightly integrate research, system building, education, and outreach
- Our experience suggests this agenda is highly promising and practical
  - suggests a new kind of data system: open-world ecosystem of tools
- See vision paper with same title on my homepage