Magellan: Toward a System-Building Agenda for Semantic Matching

AnHai Doan
University of Wisconsin-Madison & Informatica

Joint work with many students & colleagues
Motivation

- **Worked in academia from 2000-2010**
  - developed many algorithmic solutions for schema/ontology matching

- **Worked in industry 2010-2014**
  - realized that many of these solutions were not applicable
  - no open-source code that could be immediately used
  - impact of academic work was very limited

- **Back in academia in 2015**
  - decided to focus on building systems that real users can immediately use
  - hoped that if such systems were built, academic work would follow, I can make more impacts

- **Decided to focus on entity matching**
  - was easier to get data
  - but eventually want to consider other semantic matching tasks too
## Entity Matching (EM)

**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>
The Magellan Project @ UW-Madison

- Started in 2015
- Develop a general-purpose EM platform
- Inspired by
  - PostgreSQL for relational data management
  - Scikit-learn for machine learning
  - Hadoop/Spark for big data processing
Significant Progress in Past Five Years

- Deployed at 12 companies and domain science groups
  - 8 companies: Walmart, Recruit Holdings, Johnson Control, AF Insurance, Informatica, etc.
  - 4 domain sciences: Economics, Limnology, Biomedicine, Land Use
  - Pushed into production in 8 cases

- Contributed to several high-profile projects
  - saving Amazon forest, managing water quality in the Greater Lake region of the US

- Used by 500+ students in 6 data science courses at UW-Madison

- Commercialized by GreenBay Technologies
  - Acquired by Informatica in Aug 2020
  - Pushed into an EM platform to be used by thousands of customers
  - Influencing solutions for schema matching and knowledge graph construction

- Multiple research papers, SIGMOD/ACM Research Highlight Awards
The R&D Template of Magellan

1. **Identify the problem and user populations**
2. **Understand how a user typically does EM**
3. **Identify pain points and develop tools/guidance**
   - Goal is to improve productivity of the user
4. **Build tools into three data science ecosystems**
   - On-prem, cloud, mobile
   - Make tools atomic and easy to combine
   - Combine tools to build easy-to-use EM systems for users
5. **Work with real users, learn, and repeat**

- Radically different from prior system building efforts
- Can be applied to other problems: IE, schema/ontology matching, etc.
1. Identify Problems & User Populations
   - Focus on simple but common problems
   - Focus on user populations we can easily work with
Identify Problems

- Use supervised machine learning

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>
A “Very Boring” Problem

- **Received very little attention**
  - judged trivial, hard to develop novel technical solutions, hard to publish

- **Most academic works focus on more complex problems**
  - e.g., how to exploit a knowledge graph to improve the accuracy of EM
  - easier to develop novel technical solutions

- **We selected the above problem because many users need to solve it**
  - especially the “horses”
“Horse” Populations That We Target

- **Domain scientists**
  - Biomedicine, land use, limnology, economics, etc.
  - They are within walking distance
  - Domain experts, some coding skills (e.g., Python, R, SQL)

- **Students, educators, researchers in data integration, data science**
  - Students form teams to do class project, we asked each team to solve an EM problem

- **Data scientists at companies**
  - Often work in a way similar to domain scientists

- **Lay users, data enthusiasts**
  - Journalists, citizen data scientists; domain experts, but often no coding skills

- **We do not target enterprise customers**
  - They often want “hardcore” stuff: proprietary code, big/complex processes, lot of support
  - But we ended up working with a few
2. Understand How a User Typically Does EM
   - Observe how real users do it
   - Observe how students do it in class projects
Existing Work Has a Relatively Simple View of How User Does EM

<table>
<thead>
<tr>
<th>Table A</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>City</strong></td>
<td><strong>State</strong></td>
</tr>
<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>
<thead>
<tr>
<th>Table B</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>City</strong></td>
<td><strong>State</strong></td>
</tr>
<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>

- Focuses on developing blockers and matchers
We Observe That Real-World EM Processes Are Far More Complex

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

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

1M tuples

A

block → match
(using machine learning)

1M tuples

B
select a good blocker:
blocker X, blocker Y

select a good matcher:
matcher U, matcher V

Cross-validate matcher U

Cross-validate matcher V

quality check

yes

no
Production Stage

Scaling, quality monitoring, exception handling, crash recovery, …
3. Identify Pain Points and Develop Tools/Guidance
Example Pain Points

- How to select a good blocker?

- How to debug a blocker?

- How to sample and label?

- How to debug a matcher?

- How to sample and label?

- How to debug a matcher?

- Cross-validate matcher U

- Cross-validate matcher V

- Select a good matcher

- Sample

- Cross-validate matcher G

- Label

- Quality check

- Yes

- No
Debugging a Blocker

- Does blocker Q kill off too many matches?
- What are the killed-off matches?
- Why are they killed off by Q?

Table A

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>Dave Smith</td>
<td>Altanta</td>
</tr>
<tr>
<td>a₂</td>
<td>Daniel Smith</td>
<td>LA</td>
</tr>
<tr>
<td>a₃</td>
<td>Joe Wilson</td>
<td>New York</td>
</tr>
<tr>
<td>a₄</td>
<td>Charles Williams</td>
<td>Chicago</td>
</tr>
<tr>
<td>a₅</td>
<td>Charlie William</td>
<td>Atlanta</td>
</tr>
</tbody>
</table>

Table B

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>David Smith</td>
<td>Atlanta</td>
</tr>
<tr>
<td>b₂</td>
<td>Joe Wilson</td>
<td>NY</td>
</tr>
<tr>
<td>b₃</td>
<td>Daniel W. Smith</td>
<td>LA</td>
</tr>
<tr>
<td>b₄</td>
<td>Charles Williams</td>
<td>Chicago</td>
</tr>
</tbody>
</table>

blocker Q

\[ a₃ . \text{City} = b₃ . \text{City} \]

\[ \begin{align*}
(a₂, b₃) \\
(a₄, b₄) \\
(a₅, b₁)
\end{align*} \]

C
Debugging a Blocker

- Debugger quickly finds matches killed-off by the blocker
- User examines these matches and improves the blocker

\[ A \times B \]

\[ D = A \times B \setminus C \]

\[ C = Q(A, B) \]

Tables A, B → Config Generator → Set of configs → Top-k SSJs → Set E of match candidates → Match Verifier → Matches in E Explanations

Output C of blocker Q → Rank aggregator → Active/online learning → Top-n pairs → User feedback
Learning a Blocker

- Take sample $S$ from $A \times B$ (without materializing $A \times B$)
- Train a random forest $F$ on $S$ (to match tuple pairs)
  - using active learning, where user labels pairs

```
● Sample $S$ from $|A \times B|$
  ● Train a random forest $F$
    ● Stopping criterion satisfied?
      - $Y$
      - $N$
      - Select $q$ “most informative” unlabeled examples
      - Label the $q$ selected examples
```

User
## Learning a Blocker

### 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>213-467-1108</td>
<td>california</td>
<td>16</td>
</tr>
<tr>
<td>236</td>
<td>patina</td>
<td>'5955 melrose ave.'</td>
<td>los angeles</td>
<td>213-467-1108</td>
<td>california</td>
<td>16</td>
</tr>
<tr>
<td>555</td>
<td>valentino</td>
<td>'3115 pico blvd.'</td>
<td>santa monica</td>
<td>310-699-4313</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>310-699-4313</td>
<td>italian</td>
<td>21</td>
</tr>
<tr>
<td>875</td>
<td>sammy's romanian steak house</td>
<td>'157 chryslle 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>212-867-4655</td>
<td>steakhouses</td>
<td>641</td>
</tr>
<tr>
<td>982</td>
<td>buren'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>'9650 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>'2937 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>'2937 peachtree rd. ne'</td>
<td>atlanta</td>
<td>404-231-1368</td>
<td>italian</td>
<td>85</td>
</tr>
</tbody>
</table>
Learning a Blocker

- Extract candidate blocking rules from random forest F

Example random forest F for matching books

Extracted candidate blocking rules

- \((isbn\_match = N) \rightarrow \text{No}\)
- \((isbn\_match = Y) \text{ and } (#pages\_match = N) \rightarrow \text{No}\)
- \((title\_match = N) \rightarrow \text{No}\)
- \((title\_match = Y) \text{ and } (publisher\_match = N) \rightarrow \text{No}\)
- \((title\_match = Y) \text{ and } (publisher\_match = Y) \text{ and } (year\_match = N) \rightarrow \text{No}\)
Collaborative Labeling

Tool to highlight possible matching definitions

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]) +
Tool to Highlight Possible Match Definitions

- We do not have a tool yet, but we do have guidance for user
  1. Take a small sample $S$ of tuple pairs (say 50)
  2. All labelers must label $S$
  3. Compare their outputs, highlight discrepancies, discuss
  4. Repeat Steps 1-3 until no more discrepancies
  5. During above steps, document all possible match definitions that come up

- In addition to above guidance, currently we also recommend the following
  - Use blocking debugger to return pairs that are likely to be matches, collectively discuss them
  - Use an active learner to identify controversial pairs, collectively discuss them
What We Learn When Working with Users

- **They need to understand (and agree on) the match definition**
  - KFC on Univ Ave = KFC on Farewell Ave?
  - iPhone 6s black = iPhone 6s white?
  - The Amazon rain forest group has worked on EM for three years, and yet still have problems

- **They need to understand the data (tables A and B)**
  - How dirty? Lot of missing values? Any portion of data is unreliable?

- **They need to understand the limitations of tools**
  - Can random forests match textual data accurately?

- **Need to develop tools and guidance to help them gain this understanding**
  - As a part of the EM process
Summary

● A user typically does EM in a multi-step iterative complex process
  – Far more complicated than we thought, we do not fully understand it today
  – Need more work to completely specify this EM process

● Cannot be completely automated, aims instead to improve user productivity
  – Keep the same process, but make it easier for user to execute (far less ambitious goal)

● Identify pain point steps in the process, for each such step
  – Develop (semi-)automatic tools to help user if possible
  – If not, develop guidance telling user how to do the step
4. Build Tools into Three Data Science Ecosystems

- On-prem, cloud, mobile
- Make tools atomic & easy to combine
- Combine tools to build easy-to-use EM systems
Our First Observation

- **Tools need to exploit a wide variety of techniques**
  - Relational data processing, ML, statistics, visualization, cleaning, etc.
- **Very time consuming to implement so many techniques from scratch**
  - Best to exploit existing data science ecosystems
  - A natural starting point is PyData, ecosystem of DS tools in Python
- **We also don’t want complex “monolithic” tools**
  - Difficult to build & maintain them in academia
  - Difficult to reuse
  - Difficult to combine them in unexpected ways, which users often do
- **So we build tools that are atomic and easy to combine**
- **Build them as commands in Python packages, as part of PyData**
- **Then combine them to build more complex tools**
# Current PyData Tools

<table>
<thead>
<tr>
<th>Step of the How-to Guide (A)</th>
<th>Use Existing Packages (B)</th>
<th>Write Our Own Code (C)</th>
<th>Develop Tools for Pain Points (D)</th>
<th>Number of Commands (E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read/Write Data</td>
<td>pandas</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Down Sample</td>
<td></td>
<td></td>
<td>Down sampler</td>
<td>1</td>
</tr>
<tr>
<td>Data Exploration</td>
<td>pandas</td>
<td>pandas-profiling</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pandas-table</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OpenRefine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocking</td>
<td>Dask</td>
<td>Multiple blockers</td>
<td>Blocking debugger</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>joblib</td>
<td>py_stringmatching</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>py_stringsimjoin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sampling</td>
<td>pandas</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Labeling</td>
<td>PyQt5</td>
<td></td>
<td>GUI labeler</td>
<td>2</td>
</tr>
<tr>
<td>Creating Feature Vectors</td>
<td>joblib</td>
<td>py_stringmatching</td>
<td>Automatic feature creation</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Manual feature creation</td>
<td></td>
</tr>
<tr>
<td>Matching</td>
<td>scikit-learn</td>
<td></td>
<td>Matching debuggers</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>PyTorch</td>
<td></td>
<td>Deep learning-based matcher</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XGBoost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computing Accuracy</td>
<td>pandas</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Adding Rules</td>
<td></td>
<td></td>
<td>Rule specification and execution</td>
<td>9</td>
</tr>
<tr>
<td>Managing Metadata</td>
<td></td>
<td>Catalog management</td>
<td></td>
<td>22</td>
</tr>
</tbody>
</table>

**Main Packages:** py_stringmatching, py_stringsimjoin, py_entitmaching, py_labeler, DeepMatcher
An EM scenario, e.g., matching Tables A, B

Pain point

Pain point

Power user

EM requirements (accuracy, runtime, etc.)

EM workflow

How-to guide

Step 1

Step 2

... 

Step n

Problem definition

Data profile

Tool profile

EM workflow (procedural)

Tables A, B

Executor

Matches

Optimizer

EM plans ← EM operators

EM workflow (declarative)

Tools as commands packages in the Python data analysis stack

Tools as commands packages in the Python big data stack
Our Second Observation

- **Tools in PyData ecosystem can be used mostly on-prem**
- **When doing EM, users often want to move among three exec environments**
  - On-prem, cloud, mobile
So We Build into All Three Execution Environments

**PyData**
- py_stringmatching
- py_stringsimjoin
- py_entitymatching
- deepmatcher

**AWS**
- RuleExecutor
- CloudLabeler
- CloudProfiler
- ActiveLearner

- MobileLabeler
- Cymphony
Combine Tools to Create Easy-to-Use EM Systems

PyData

- py_stringmatching
- py_stringsimjoin
- py_entitymatching
- deepmatcher

AWS

- RuleExecutor
- CloudLabeler
- CloudProfiler
- ActiveLearner
- CloudMatcher

---

### Diagram

1. **Sampling** (A, B) → **Active Learning** (S)
2. **Random Forest** F
3. **Extract & Evaluate Blocking Rules**
4. **Execute Blocking Rules**
5. **Canidate Set** C
6. **Apply G to C**

(a) **Blocking**

(b) **Matching**
# Real-World Deployment of CloudMatcher

<table>
<thead>
<tr>
<th>Problem Owner</th>
<th>Problem Type</th>
<th>Table A</th>
<th>Table B</th>
<th>Precision (in %)</th>
<th>Recall (in %)</th>
<th>Cost</th>
<th>Time</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>American Family Insurance</strong></td>
<td>Phoenix customers</td>
<td>300</td>
<td>300</td>
<td>96.4</td>
<td>99.03</td>
<td>160</td>
<td>-</td>
<td>$2.33</td>
</tr>
<tr>
<td></td>
<td>Commercial insurance policy holders</td>
<td>1,049</td>
<td>17,572</td>
<td>96.15</td>
<td>97.22</td>
<td>321</td>
<td>-</td>
<td>$2.33</td>
</tr>
<tr>
<td></td>
<td>Commercial farm/ranch policy members</td>
<td>109,974</td>
<td>4,922,505</td>
<td>99.5</td>
<td>95</td>
<td>780</td>
<td>-</td>
<td>$13.96</td>
</tr>
<tr>
<td></td>
<td>Vehicles</td>
<td>18,938</td>
<td>72,898</td>
<td>66.02 – 80.02</td>
<td>81.65 – 93.15</td>
<td>851</td>
<td>-</td>
<td>$7.00</td>
</tr>
<tr>
<td></td>
<td>Drivers</td>
<td>790</td>
<td>634</td>
<td>99.86</td>
<td>94.89</td>
<td>250</td>
<td>-</td>
<td>$2.33</td>
</tr>
<tr>
<td><strong>Johnson Controls International</strong></td>
<td>Addresses</td>
<td>90,673</td>
<td>231,081</td>
<td>93.22 – 95.72</td>
<td>76.93 – 81.01</td>
<td>1200</td>
<td>-</td>
<td>$72</td>
</tr>
<tr>
<td></td>
<td>Vendors</td>
<td>50,295</td>
<td>50,292</td>
<td>29.95 – 38.04</td>
<td>91.89 – 98.10</td>
<td>1160</td>
<td>$69.60</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Vendors (no Brazil)</td>
<td>28,152</td>
<td>28,149</td>
<td>95.44 – 97.75</td>
<td>88.82 – 92.41</td>
<td>1200</td>
<td>$72</td>
<td>-</td>
</tr>
<tr>
<td><strong>UW Health</strong></td>
<td>Doctors &amp; staff</td>
<td>1,786</td>
<td>1,786</td>
<td>99.66</td>
<td>98.18</td>
<td>1200</td>
<td>-</td>
<td>$4.66</td>
</tr>
<tr>
<td><strong>Informatica</strong></td>
<td>Persons</td>
<td>48,119</td>
<td>48,119</td>
<td>100 – 100</td>
<td>98.42 – 100</td>
<td>462</td>
<td>-</td>
<td>$7.00</td>
</tr>
<tr>
<td><strong>Marshfield Clinic</strong></td>
<td>Drugs</td>
<td>446,048</td>
<td>440,048</td>
<td>99.14 – 99.63</td>
<td>98.45 – 99.14</td>
<td>1162</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Non-profit Org</strong></td>
<td>Elected officials</td>
<td>9,751</td>
<td>706,878</td>
<td>93.75 – 96.32</td>
<td>95.50 – 97.76</td>
<td>960</td>
<td>$57.60</td>
<td>-</td>
</tr>
<tr>
<td><strong>Domain Science</strong></td>
<td>UMetrics economics</td>
<td>2,616</td>
<td>21,530</td>
<td>94.5 – 96.5</td>
<td>98.12 – 99.21</td>
<td>680</td>
<td>$61.20</td>
<td>-</td>
</tr>
</tbody>
</table>

- Outperformed three commercial systems
Discussion & Lessons Learned

- **Building systems then using them to do research**
  - a great way to make impacts

- **It is possible to build such systems in academia with a small team**
  - we have never had a full-time programmer, just a few graduate students
  - system has many small independent tools, each student works on a tool

- **Do not overlook “boring trivial problems” for the “horses”**
  - often turn out to be very technically challenging

- **Our system-building template is very promising**
  - validated by what we have seen at Informatica
Discussion & Lessons Learned

- For the entity matching community
  - need a lot more data sets
    - that are diverse, otherwise hard to know if a technique is robust
    - that are big (10-50M tuples), many things break at scale
    - that have different levels of noise, as noisy data can really impact accuracy & runtime
    - really need gold for these data sets, but hard to create
  - benchmarks and competitions must focus on a lot more pain points
    - so far mostly focus on the matching step
    - need to consider more pain points
      - blocking, data cleaning in a pre-processing step, debugging, labeling, etc.
Discussion & Lessons Learned

- **For the ontology matching community**
  - would be great if can help with two major problems faced by thousands of companies & domain scientists
  - schema matching for data lakes
    - given a data lake (say having 100K tables), find all column pairs that match
  - business glossary matching for data lakes
    - given a set of business terms and a data lake, find all pairs <term, column> that match
    - “Mfg Location Capacity” matches column “MLCap”
    - “House’s Listed Price” matches column “HPrice”

- **These are not ontology matching, but very closely related**
  - major problems in industry & domain sciences
Discussion & Lessons Learned

● **Can apply the Magellan template to these two problems**
  – identify the end-to-end process that a real user follows to solve them
  – identify pain points, develop tools & guidance

● **There are numerous pain points**
  – cleaning column names
    ▪ “MPCap” => Manufacturing Location Capacity
    ▪ “HPrice” => House Price
  – finding synonyms in the lake
    ▪ Manufacturing = Factory, Location = Area, A/C = Cooling
  – scaling up blocking
    ▪ need to scale for lakes of up to 10M columns

● **But first must create data sets with gold**
  – a critical but difficult problem
  – how to create gold for a data lake with 100,000 columns? **Solving this makes big impacts**
Conclusions

- **Magellan seeks to build a general platform for entity matching**
  - generalized later to other semantic matching tasks

- **Departing radically from existing work**
  - observes that the EM process is often very complex, driven by user, can’t be automated, so focus on improving the productivity of user
    - identify the complex EM process
    - identify pain points, develop tool/guidance
    - build tools into three data science ecosystems

- **Provide a promising R&D template for other semantic matching problems**
  - schema/ontology matching, business glossary matching, etc.
Reference Papers

- **Entity Matching Meets Data Science: A Progress Report from the Magellan Project**, Y. Govind, P. Konda, and others. *SIGMOD-19*