Magellan: Toward a System-Building Agenda for Semantic Matching

AnHai Doan
University of Wisconsin-Madison & Informatica

Joint work with many students & colleagues

Informatica

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

Name	City	State
Dave Smith	Madison	WI
Joe Wilson	San Jose	CA
Dan Smith	Middleton	WI

Table B

Name	City	State
David D. Smith	Madison	WI
Daniel W. Smith	Middleton	WI

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

Table A

Name	City	State
Dave Smith	Madison	WI
Joe Wilson	San Jose	CA
Dan Smith	Middleton	WI

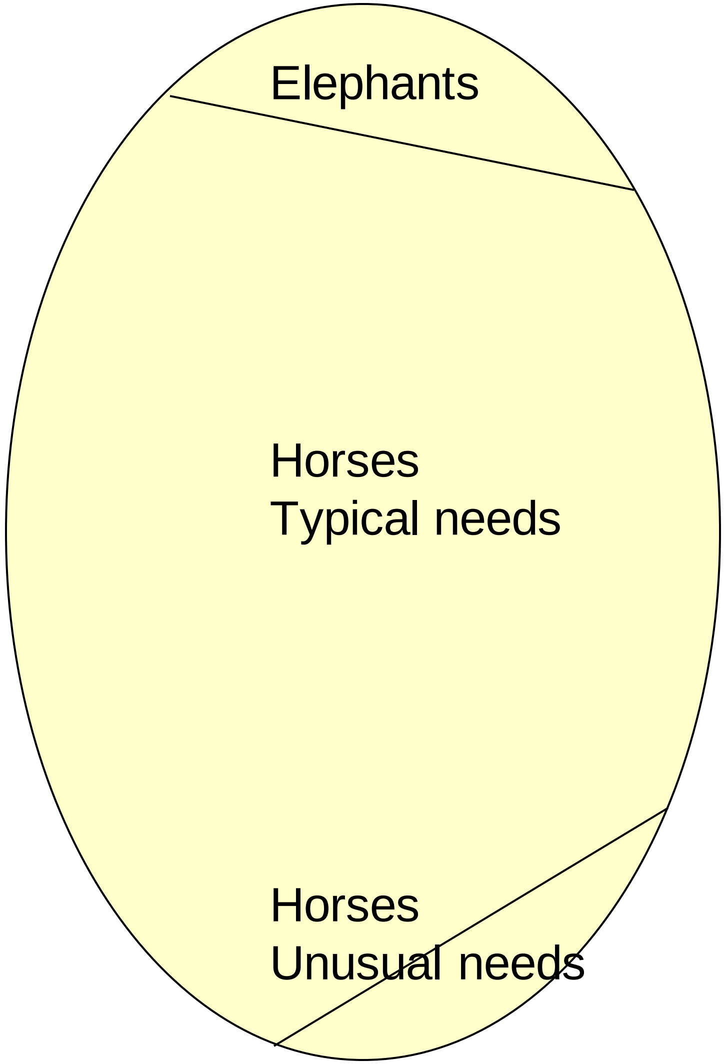
Table B

Name	City	State
David D. Smith	Madison	WI
Daniel W. Smith	Middleton	WI

- **Use supervised machine learning**

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 A

	Name	City	State
a_1	Dave Smith	Madison	WI
a_2	Joe Wilson	San Jose	CA
a_3	Dan Smith	Middleton	WI

Table B

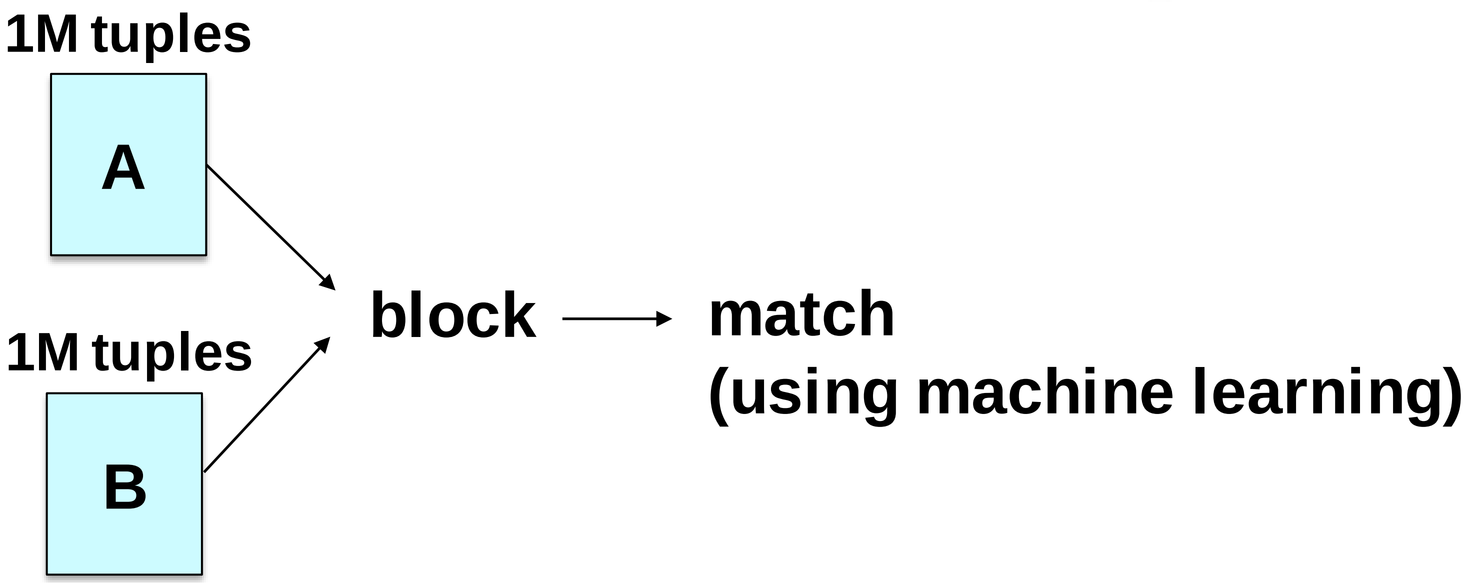
	Name	City	State
b_1	David D. Smith	Madison	WI
b_2	Daniel W. Smith	Middleton	WI

block on state = state → (a_1, b_1), (a_1, b_2), (a_3, b_1), (a_3, b_2) → **match** →

(a_1, b_1) +
(a_1, b_2) -
(a_3, b_1) -
(a_3, b_2) +

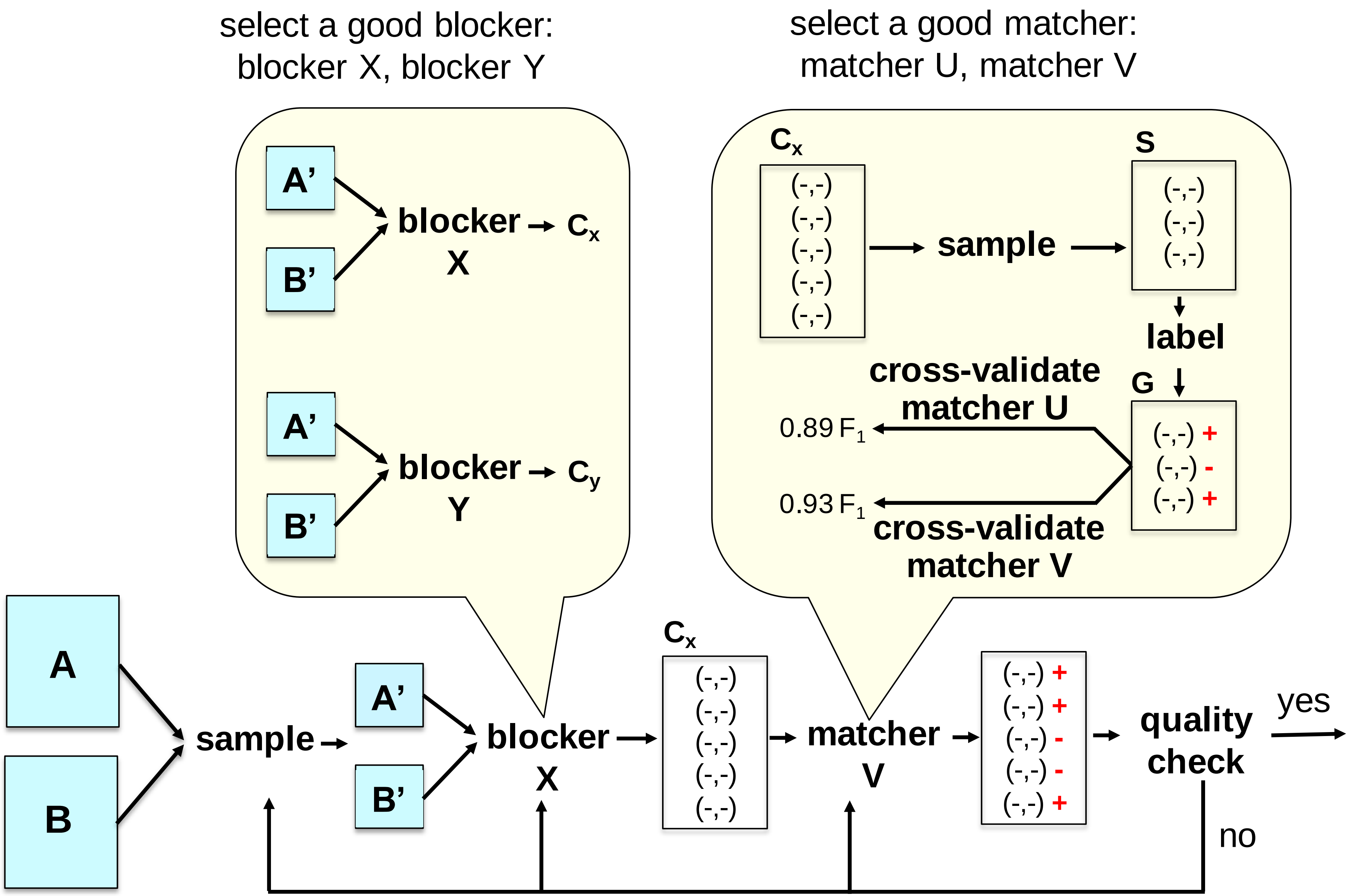
- **Focuses on developing blockers and matchers**

We Observe That Real-World EM Processes Are Far More Complex

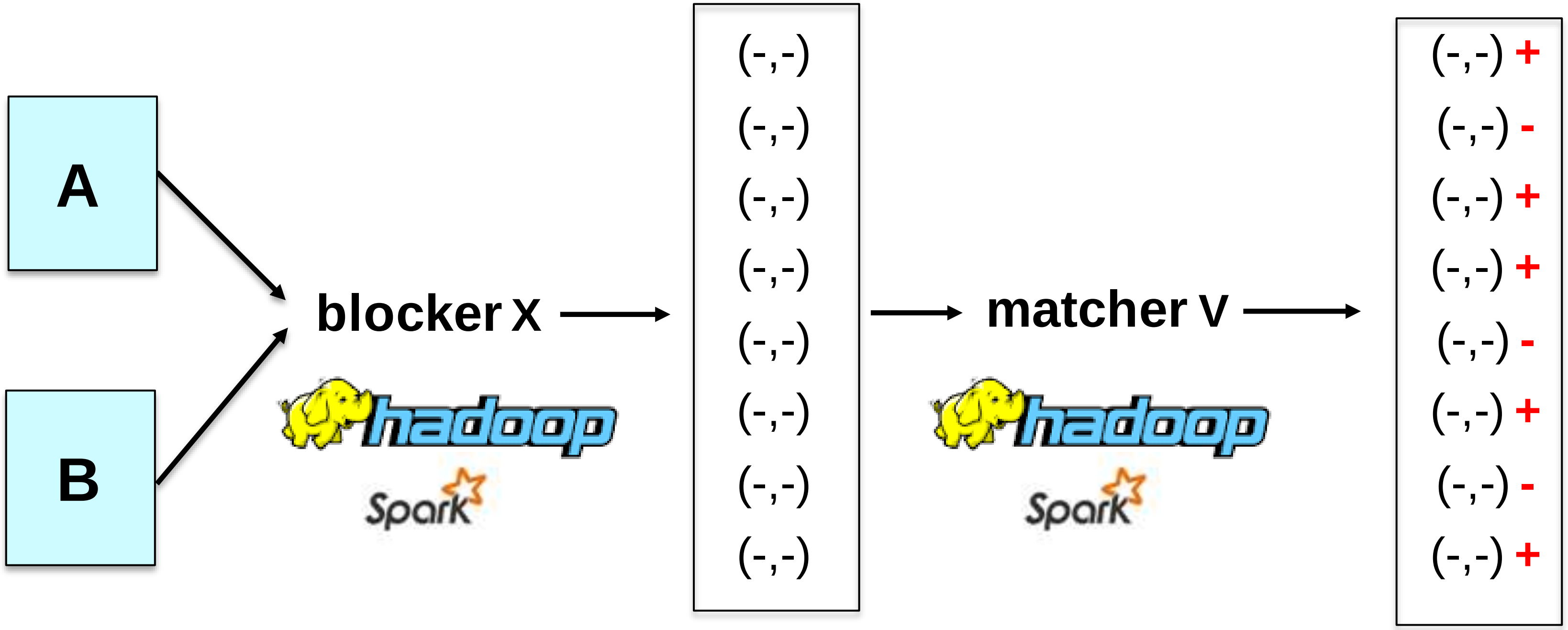


- **Development stage**
  - finds an accurate workflow, using data samples
- **Production stage**
  - executes workflow on entirety of data
  - focuses on scalability

Development Stage



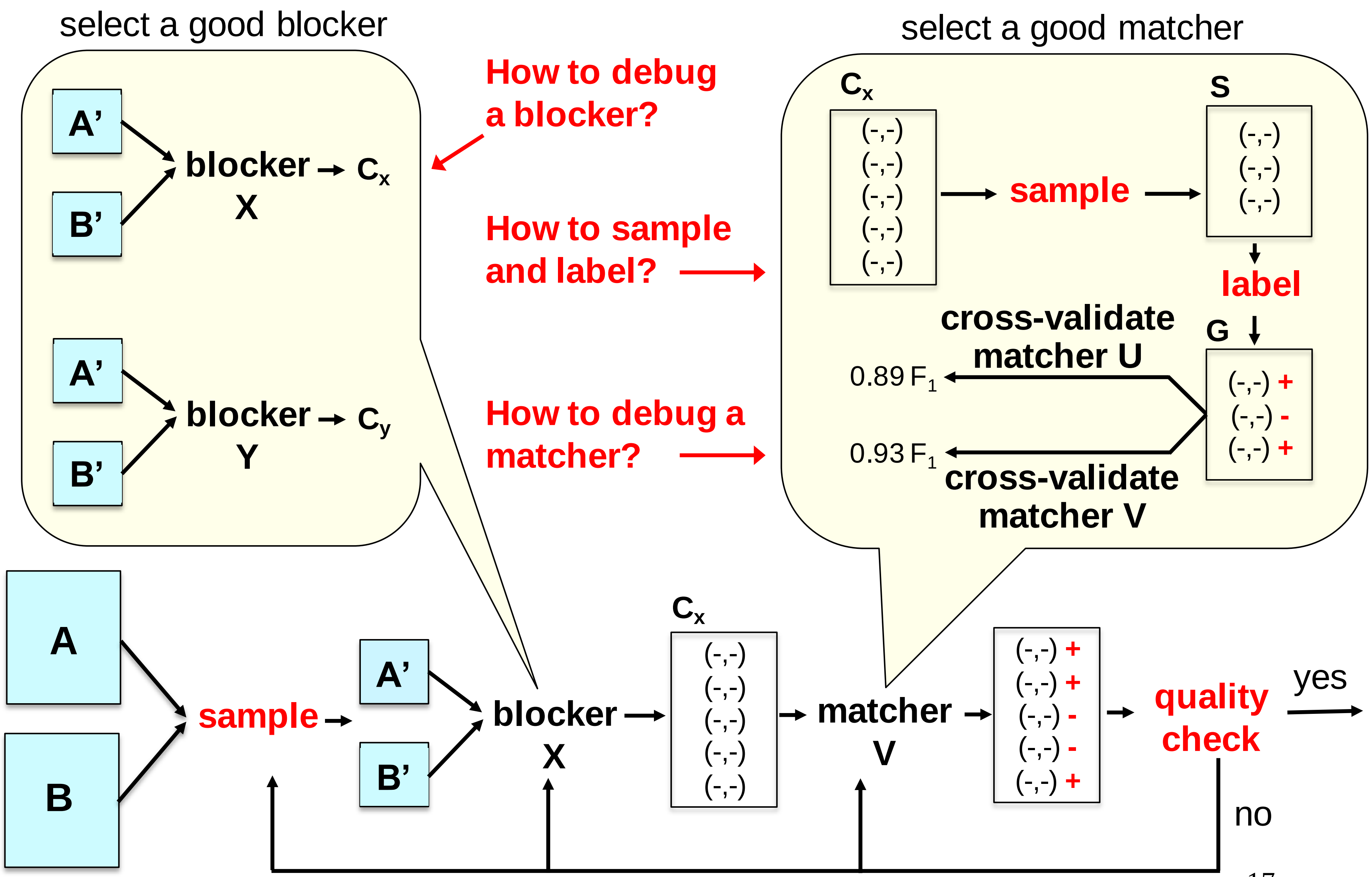
Production Stage



Scaling, quality monitoring, exception handling, crash recovery, …

3. Identify Pain Points and Develop Tools/Guidance

Example Pain Points



Debugging a Blocker

Table A

	Name	City	Age
a_1	Dave Smith	Altanta	18
a_2	Daniel Smith	LA	18
a_3	Joe Welson	New York	25
a_4	Charles Williams	Chicago	45
a_5	Charlie William	Atlanta	28

Table B

	Name	City	Age
b_1	David Smith	Atlanta	18
b_2	Joe Wilson	NY	25
b_3	Daniel W. Smith	LA	30
b_4	Charles Williams	Chicago	45

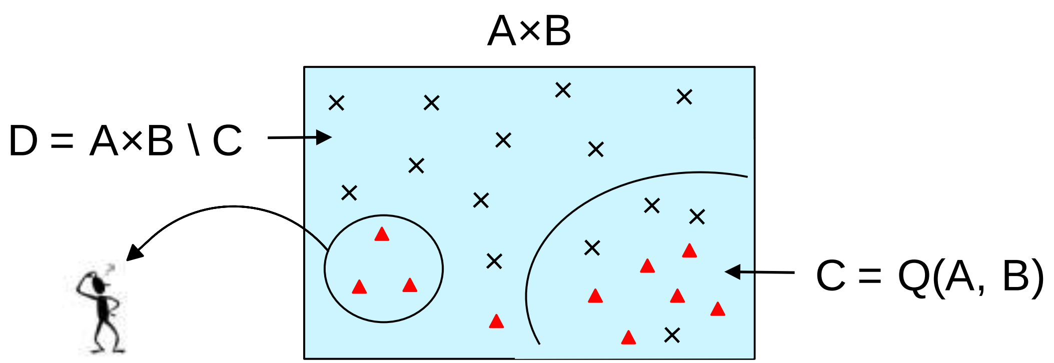
blocker Q
a.City = b.City →

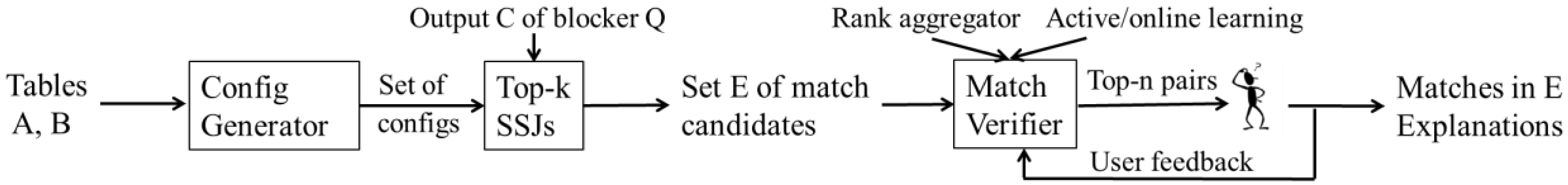
C
(a_2, b_3)
(a_4, b_4)
(a_5, b_1)

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

Debugging a Blocker

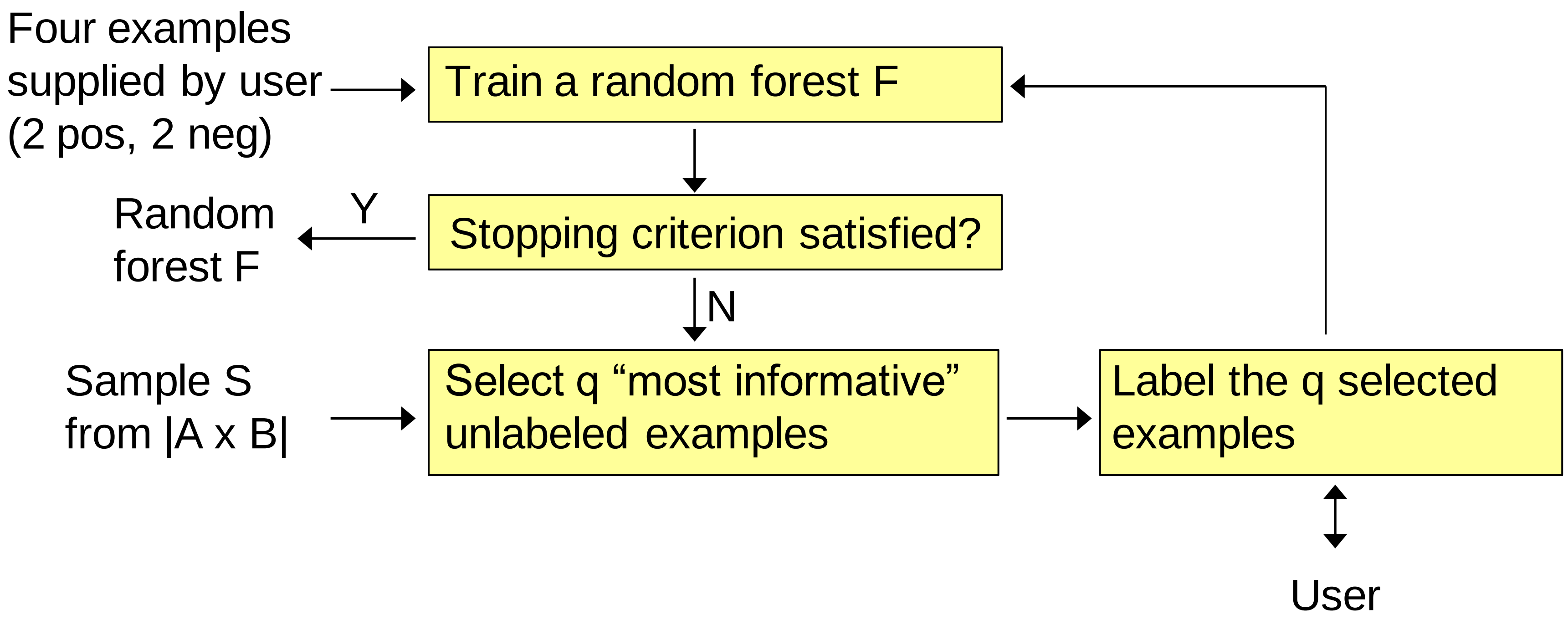
- Debugger quickly finds matches killed-off by the blocker
- User examines these matches and improves the blocker





Learning a Blocker

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



Learning a Blocker

CloudMatch Home Job Statistics Job Progress Task Monitoring Logout

Train Model

Do these pairs refer to the same real world entity?

✔ Yes 2
✖ No 2

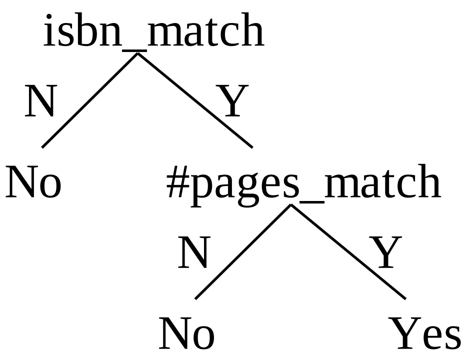
id	name	addr	city	phone	type	class
550	patina	'5955 melrose ave.'	'los angeles'	213/467-1108	californian	16
235	patina	'5955 melrose ave.'	'los angeles'	213-467-1108	californian	16
						✔Yes ✖No Unsure
555	valentino	'3115 pico blvd.'	'santa monica'	310/829-4313	italian	21
240	valentino	'3115 pico blvd.'	'santa monica'	310-829-4313	italian	21
						✔Yes ✖No Unsure
875	'sammy\'s roumanian steak house'	'157 chrystie st. at delancey st.'	'new york'	212/673-0330	'east european'	341
108	'sparks steak house'	'210 e. 46th st.'	'new york city'	212-687-4855	steakhouses	641
						✔Yes ✖No Unsure
962	'binion\'s coffee shop'	'128 fremont st.'	'las vegas'	702/382-1600	'coffee shops/diners'	428
9	'brighton coffee shop'	'9600 brighton way'	'beverly hills'	310-276-7732	'coffee shops'	542
						✔Yes ✖No Unsure
619	'la grotta'	'2637 peachtree rd. peachtree house condominium'	atlanta	404/231-1368	italian	85
304	'la grotta'	'2637 peachtree rd. ne'	atlanta	404-231-1368	italian	85
						✔Yes ✖No Unsure

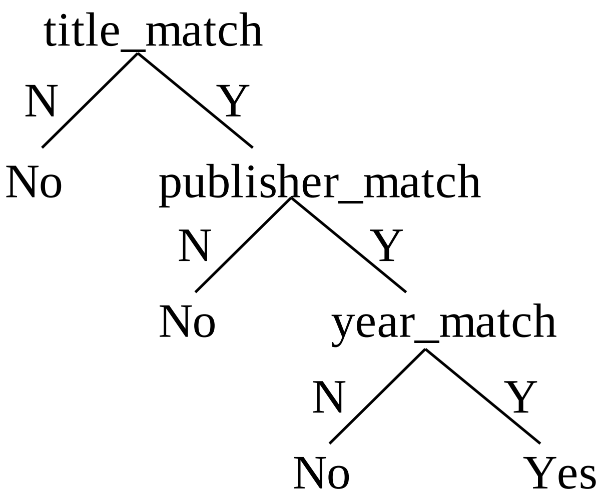
Home Submit

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) $\longrightarrow$ No

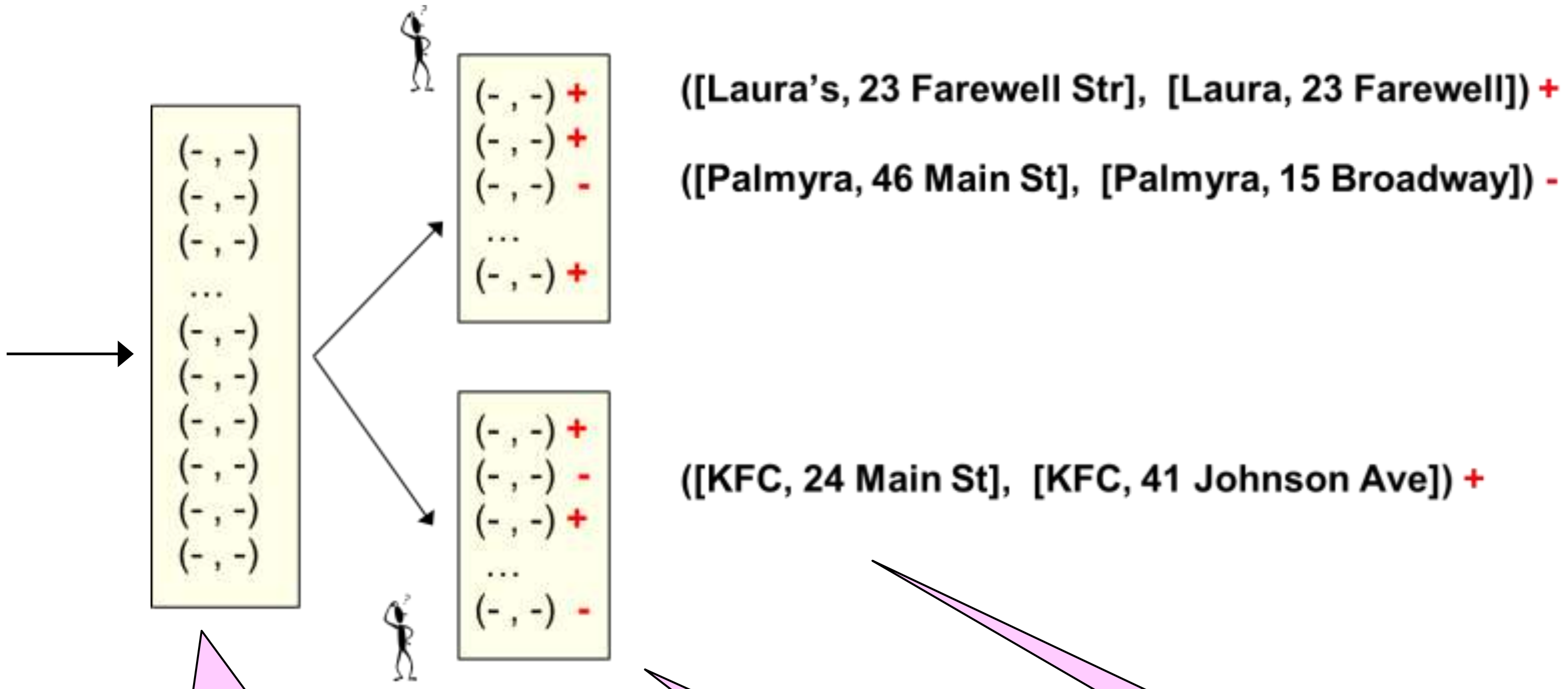
(isbn_match = Y) and (#pages_match = N) $\longrightarrow$ No

(title_match = N) $\longrightarrow$ No

(title_match = Y) and (publisher_match = N) $\longrightarrow$ No

(title_match = Y) and (publisher_match = Y) and (year_match = N) $\longrightarrow$ No

Collaborative Labeing



Tool to highlight possible matching definitions

Tool to debug labels

Tool to help revise labels

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

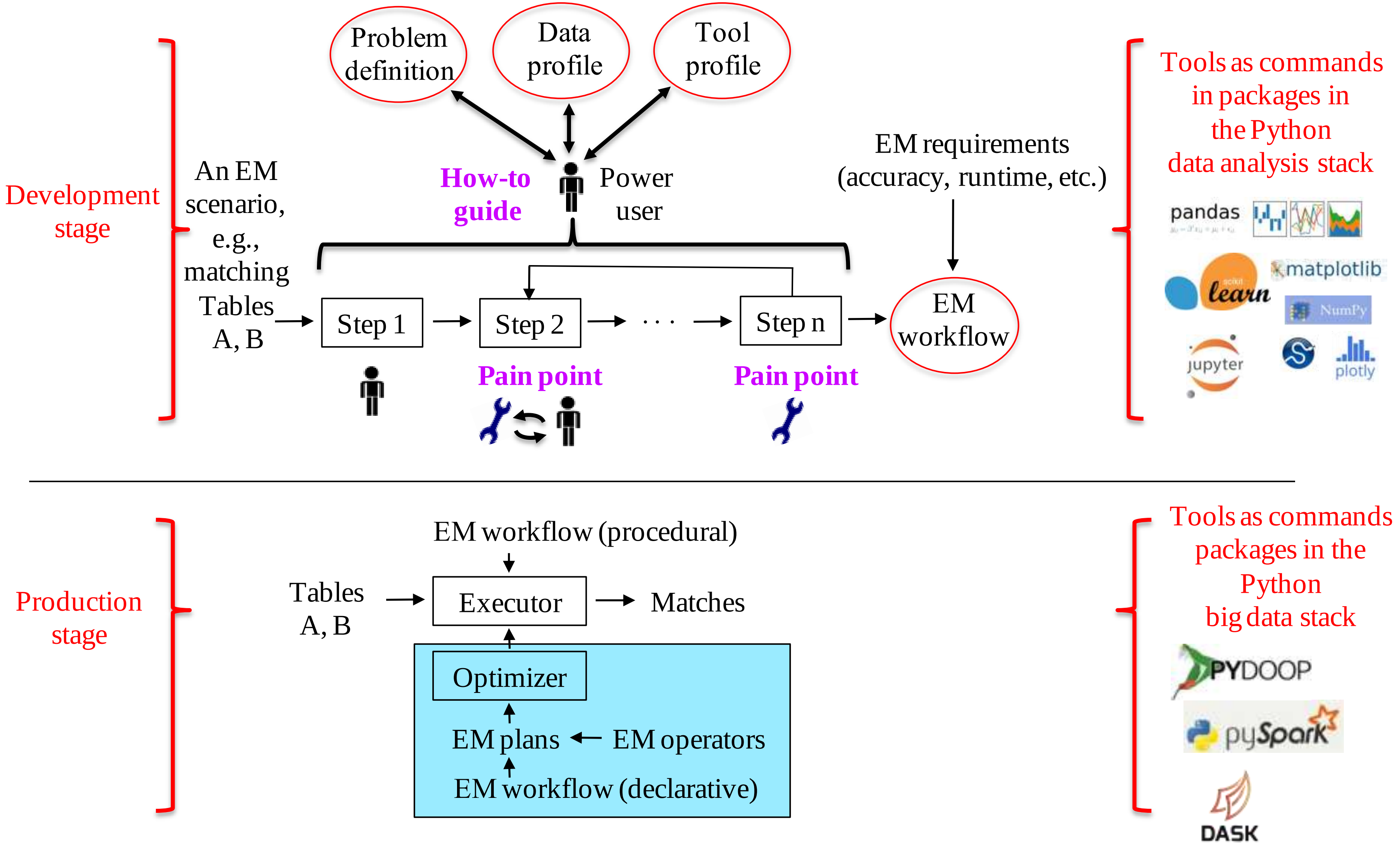
- **Tools need to exploit a wide variety of techniques**
  - Relational data processing, ML, statistics, visualization, cleaning, etc.
- **Very time consuming to implement so many techniques from scratch**
  - Best to exploit existing data science ecosystems
  - A natural starting point is PyData, ecosystem 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

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

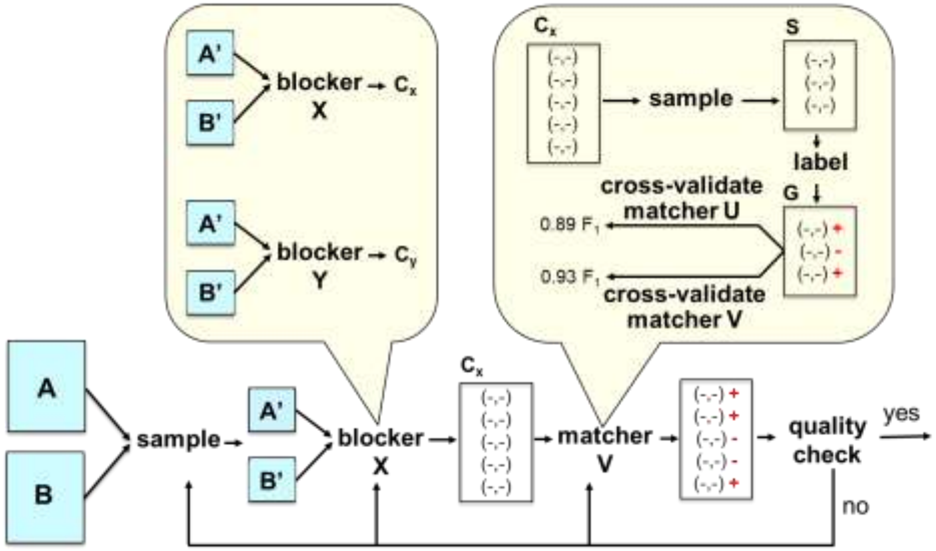
Main Packages: py_stringmatching, py_stringsimjoin, py_entitymatching, py_labeler, DeepMatcher

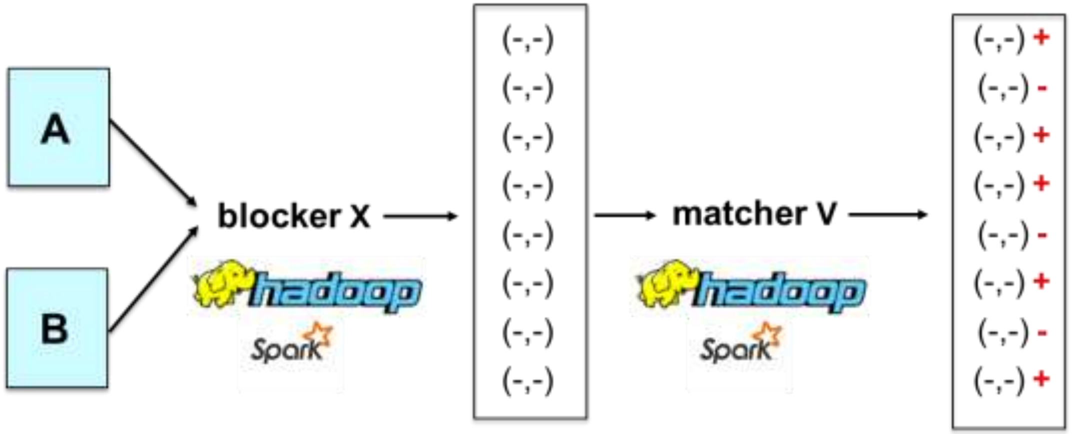
Our First System Architecture



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

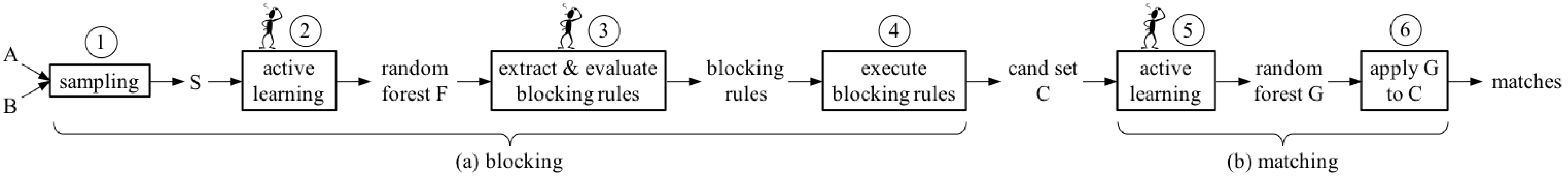
Combine Tools to Create Easy-to-Use EM Systems

PyData

- py_stringmatching
- py_stringsimjoin
- py_entitymatching
- deepmatcher

AWS

- RuleExecutor
- CloudLabeler
- CloudProfiler
- ActiveLearner
- **CloudMatcher**



Real-World Deployment of CloudMatcher

Problem Owner	Problem Type	Table A	Table B	Precision (in %)	Recall (in %)	Cost			Time		
						Questions	Crowd	Compute	User/Crowd	Machine	Total
American Family Insurance	Phoenix customers	300	300	96.4	99.03	160	-	$2.33	9m	5m	14m
	Commercial insurance policy holders	1,049	17,572	96.15	97.22	321	-	$2.33	18m	25m	43m
	Commercial farm/ranch policy members	109,974	4,922,505	99.5	95	780	-	$13.96	50m	4h 58m	5h 48m
	Vehicles	18,938	72,898	66.02 – 80.02	81.65 – 93.15	851	-	$7.00	2h	46m	2h 46m
	Drivers	790	634	99.86	94.89	250	-	$2.33	10m	8m	18m
Johnson Controls International	Addresses	90,673	231,081	93.22 – 95.72	76.93 – 81.01	1200	$72	-	36h 48m	38m	37h 26m
	Vendors	50,295	50,292	29.95 – 38.04	91.89 – 98.10	1160	$69.60	-	30h 31m	58m	31h 29m
	Vendors (no Brazil)	28,152	28,149	95.44 – 97.75	88.82 – 92.41	1200	$72	-	22h 19m	22m	22h 41m
UW Health	Doctors & staff	1,786	1,786	99.66	98.18	1200	-	$4.66	50m	15m	1h 5m
Informatica	Persons	48,119	48,119	100 – 100	98.42 – 100	462	-	$7.00	36m	1h 35m	2h 11m
Marshfield Clinic	Drugs	446,048	440,048	99.14 – 99.63	98.45 – 99.14	1162	-	-	1h 10m	8h 40m	9h 50m
Non-profit Org	Elected officials	9,751	706,878	93.75 – 96.32	95.50 – 97.76	960	$57.60	-	23h 14m	23m	23h 37m
Domain Science	UMetrics economics	2,616	21,530	94.5 – 96.5	98.12 – 99.21	680	$61.20	-	23h 12m	12m	23h 24m

- Outperformed three commercial systems

Discussion & Lessons Learned

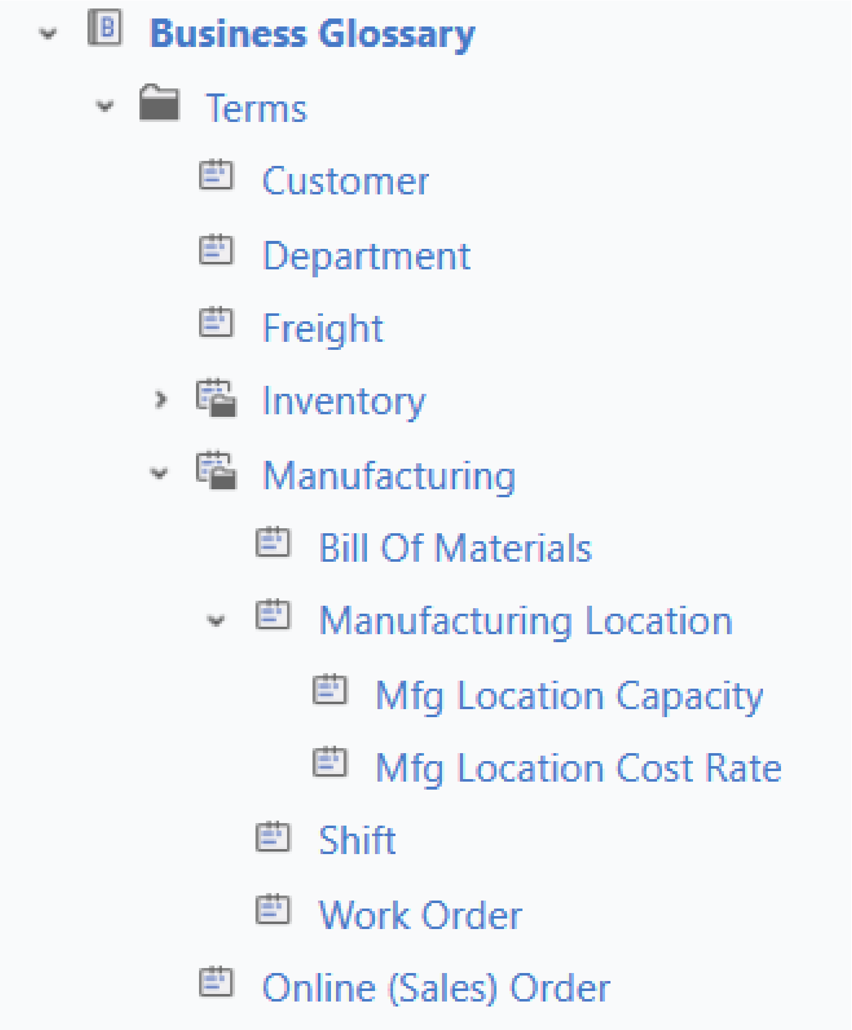
- **Building systems then using them to do research**
  - a great way to make impacts
- **It is possible to build such systems in academia with a small team**
  - we have never had a full-time programmer, just a few graduate students
  - system has many small independent tools, each student works on 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

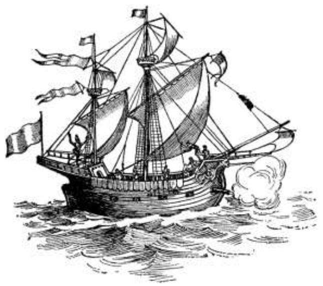
- **For the ontology matching community**
  - would be great if can help with two major problems faced by thousands of companies & domain scientists
  - schema matching for data lakes
    - given a data lake (say having 100K tables), find all column pairs that match
  - business glossary matching for 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

- Magellan: Toward Building Ecosystems of Entity Matching Solutions, AnHai Doan, Pradap Konda, Paul Suganthan G. C., Yash Govind, Derek Paulsen, Kaushik Chandrasekhar, Philip Martinkus, Matthew Christie, *Communications of the ACM*, 2020
- Entity Matching Meets Data Science: A Progress Report from the Magellan Project, Y. Govind, P. Konda, and others. *SIGMOD-19*
- Toward a System Building Agenda for Data Integration (and Data Science), A. Doan, P. Konda, P. Suganthan G.C., A. Ardalan, J. Ballard, S. Das, Y. Govind, H. Li, P. Martinkus, S. Mudgal, E. Paulson, H. Zhang. *IEEE Data Engineering Bulletin, Special Issue on Large-Scale Data Integration, 2018*
- Magellan: Toward Building Entity Matching Management Systems, P. Konda, S. Das, P. Suganthan G.C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, S. Prasad, G. Krishnan, R. Deep, V. Raghavendra. *VLDB-16*
- CloudMatcher: A Cloud/Crowd Service for Entity Matching, Y. Govind, E. Paulson, M. Ashok, P. Suganthan G.C., A. Hitawala, A. Doan, Y. Park, P. Peissig, E. LaRose, J. Badger. *BIGDAS Workshop @ KDD-17*