

# Interactive Data Integration through Smart Copy and Paste

---

**Zachary G. Ives**      **Craig A. Knoblock**      Steven Minton  
Marie Jacob      Partha Pratim Talukdar      **Rattapoom Tuchinda**  
**Jose Luis Ambite**      **Maria Muslea**      Cenk Gazen

 Univ. Pennsylvania

 USC ISI

 Fetch Technologies

*Funded in part by NSF IIS-0477972, 0513778, 0415810,  
DARPA DIESEL seedling, DARPA contract FA8750-07-D-0815/0004*

CIDR 2009  
Jan 4, 2009

# Sometimes We Need to *Rapidly* and *Iteratively* Integrate Data

---

- Combining information on-site for a FEMA emergency response effort, e.g., hurricane or earthquake...  
How do we cobble together info about resources, contacts... **rapidly?**  
*(time critical)*
- Gathering data relating to a specific gene sequence...  
May **change our integration operations** as we see more data  
*(evolving understanding of data)*
- Assembling a list of features and prices for smartphones...  
As we see new phones and features, we **change our schema**  
*(evolving understanding of domain)*
- Data is spread across many heterogeneous sources –Web pages, Excel, Word – that we are seeing for the **first time!**
  - A particular kind of “dataspace” (see Franklin+ VLDB 08 tutorial)

# Standard Data Integration Is Too Loosely Coupled, Non-Interactive

---

First: data design

- Learn the domain space
- Create a global schema
- Find sources
- Define extractors/wrappers
- Define schema mappings between extracted tables and global schema

(Design-time)

*Consult experts*

Tool #1 (ER/UML, DDL)

Tool #2 (Word of mouth, Google)

Tool #3 (Wrapper induction)

Tool #4 (Mapping)

Then: can finally query the system! (Runtime)

Nontrivial to work under this model:

- Long development time (and learning curve!)
- Iterating from design → query → design is complex

May be faster to just **manually** copy & paste data into Excel...

# Can We Make this Process Easier and Faster?

---

Integration should be as easy as manual (copy & paste) integration – “spreadsheet of data integration”

Suppose our goal is to answer a single question (query)

- May not need a full-blown integrated schema

Everything needs to be interactive, iterative:

- Discover new sources & attributes as we're going
- Change our query as we **understand the data**

# A New Integration Metaphor: Smart Copy and Paste

---

- User sees spreadsheet-like workspace for assembling tables
- We use this as a seamless environment for design & runtime
- System watches what user pastes, proposes “**auto-completions**”
  - **Extracts** more data from a source
  - Determines potential **join** query explanations for rows
  - Suggests **new attributes**
- User sees immediate **results, explanations** for what was done
- User gives **feedback**:
  - Accepts/rejects/corrects auto-completions
  - Pastes more data
- System learns, adjusts auto-completions

# The Challenge: Realizing an Integrated Smart Copy and Paste System

---

Integration becomes “programming by demonstration,” requires **learning** about data sources, integration ops

- Build upon established learning techniques used in different data integration sub-components (e.g., source extraction)
- Novelty: “integrated learning” to form a seamless cycle between **design, query answers**, and learning from **feedback**
  - User directly manipulates the *output data* to change the design
  - Data provenance is key to going from answers → sources
- Subtleties in user interaction: what is the meaning of feedback on a tuple, how do we allocate among learners?  
source data, selection conditions, join conditions, dirty data, ...

# Demonstration: The CopyCat System

---

- Scenario: hurricane relief effort in Florida, where our goal is to assemble a list of shelters and how to contact them
- Three sources:
  - Web source with shelter names (many are schools)
  - Another Web source with school contact info
  - Zip code resolution (simulated due to lack of connectivity)

# Learning a Source (Details in Paper)

Source Document

*Row feedback*

Paste



Source App

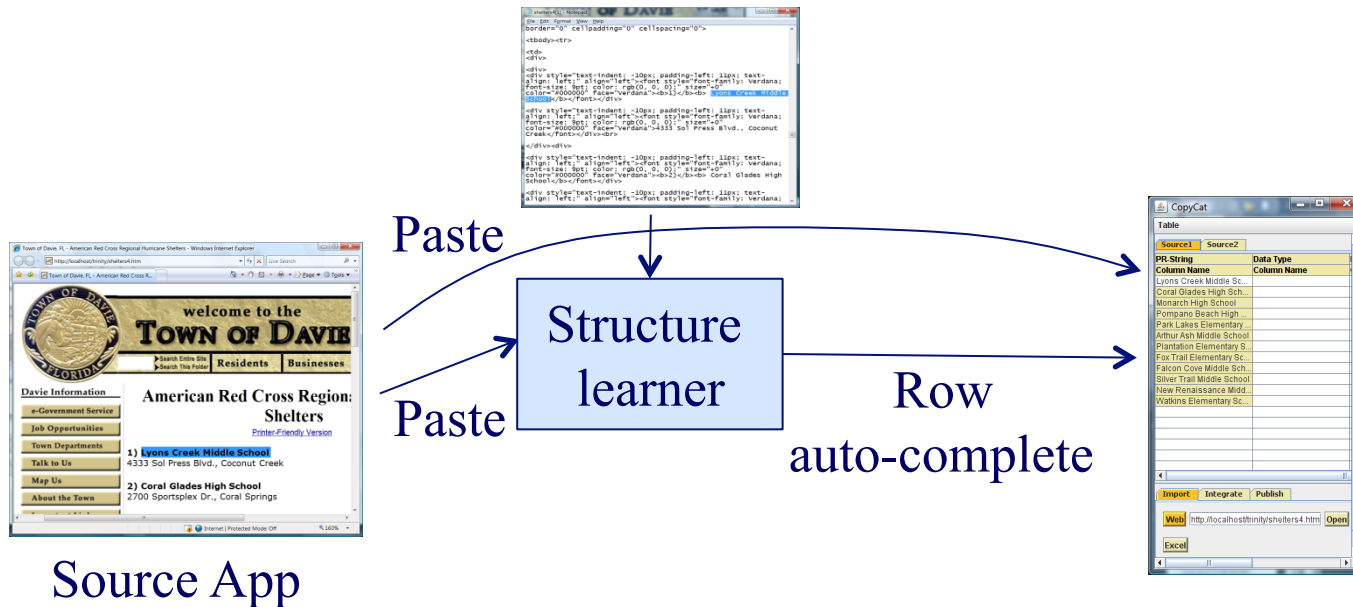
The screenshot shows the CopyCat application interface. It displays a table with two columns: "Source1" and "Source2". The "Source1" column contains a list of school names, and the "Source2" column is empty. The table is titled "Table".

Source1	Source2
PP-String	Data Type
Column Name	Column Name
Lyons Creek Middle Sch.	
Coral Glades High Sch.	
Monarch High School	
Pompano Beach High	
Park Lakes Elementary	
Arthur Ash Middle School	
Plantation Elementary S.	
Fox Trail Elementary Sch.	
Falcon Cove Middle Sch.	
Silver Trail Middle School	
New Renaissance Midd.	
Walkers Elementary Sch.	



# Learning a Source (Details in Paper)

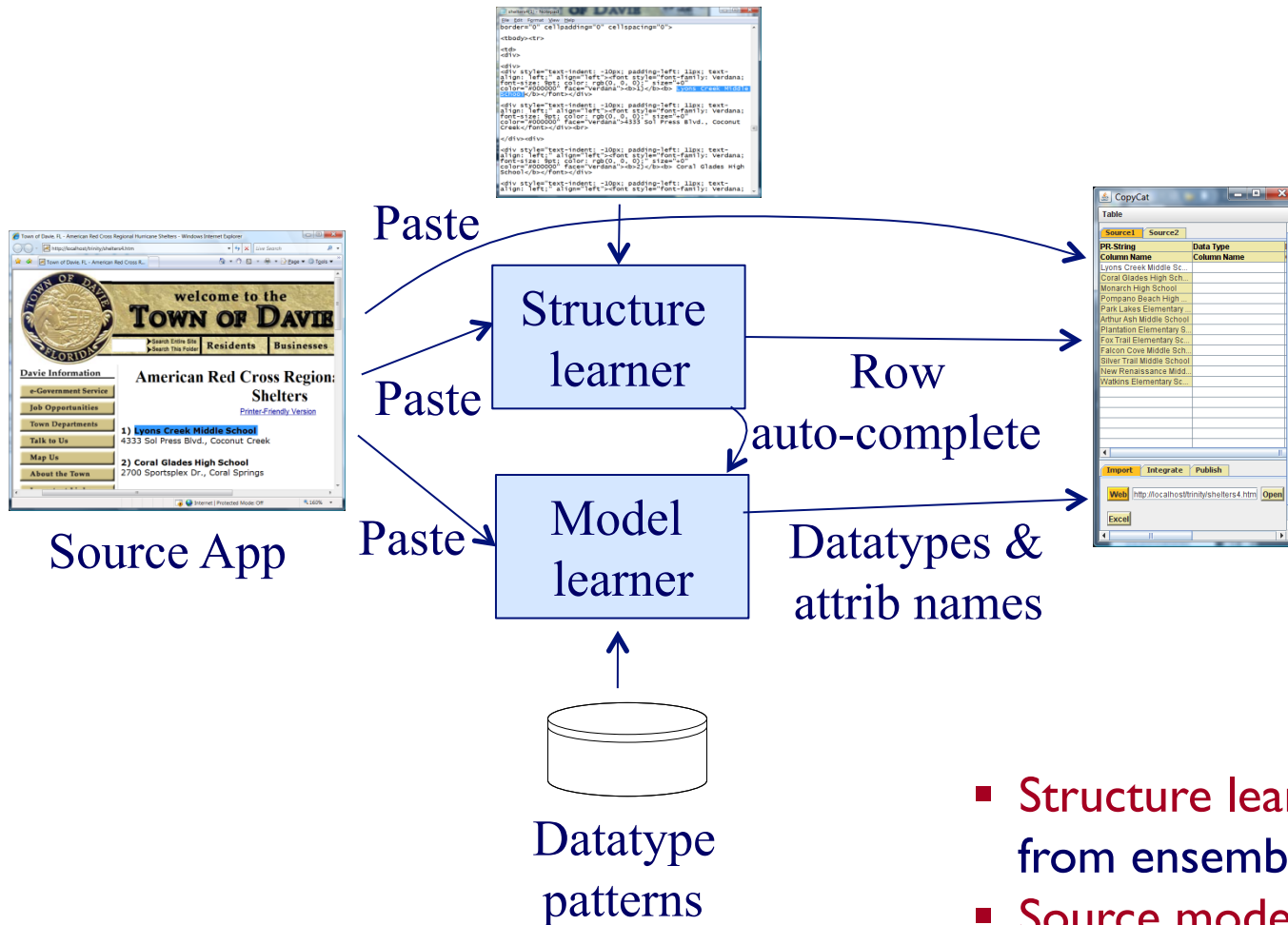
## Source Document



- **Structure learner** combines results from ensemble of sub-learners

# Learning a Source (Details in Paper)

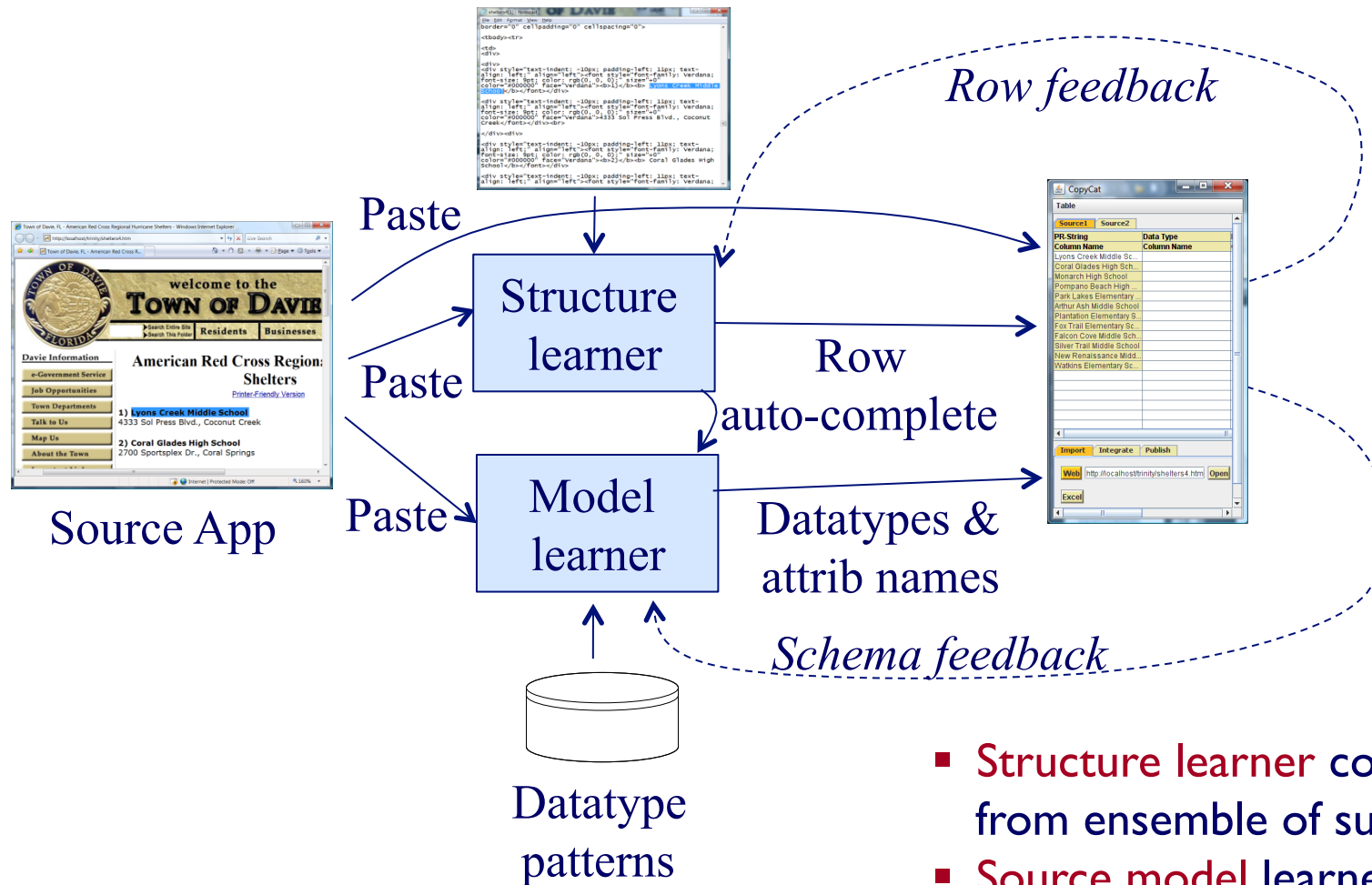
## Source Document



- **Structure learner** combines results from ensemble of sub-learners
- **Source model learner** uses logistic regression to classify datatypes

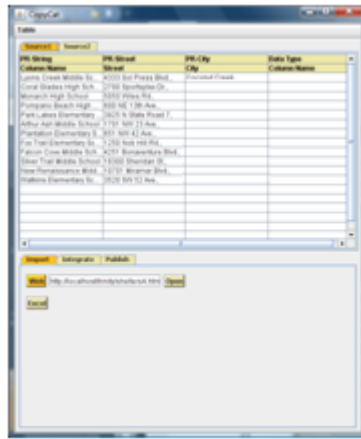
# Learning a Source (Details in Paper)

## Source Document



- **Structure learner** combines results from ensemble of sub-learners
- **Source model learner** uses logistic regression to classify datatypes

# Learning / Suggesting a Query (Details in Paper)



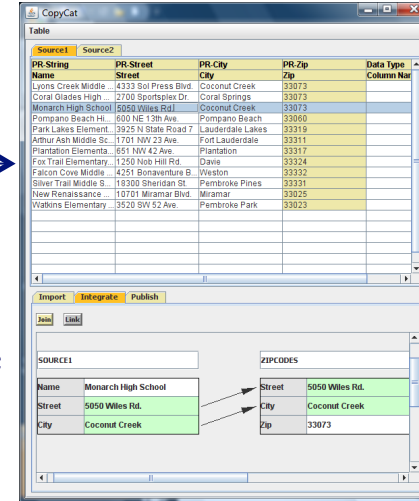
A screenshot of a spreadsheet application showing a table with columns from different sources. The columns are labeled 'PR-Street', 'PR-City', and 'Data Type'. The rows contain various school names and addresses.

Columns pasted  
from different  
sources

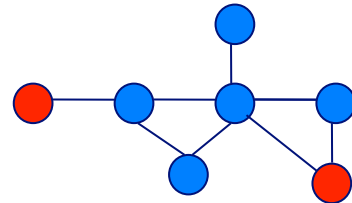
Paste

Top-k  
generator

Column  
(join query)  
auto-complete



A screenshot of a 'CopyCat' application. The top part shows a table with columns 'PR-String', 'PR-Street', 'PR-City', 'PR-Zip', and 'Data Type'. The bottom part shows a join query interface with fields for 'SOURCE1' and 'ZIPCODES'. The 'SOURCE1' field is populated with 'Monarch High School' and '5050 Willes Rd'. The 'ZIPCODES' field is populated with 'Street: 5050 Willes Rd', 'City: Coconut Creek', and 'Zip: 33073'.

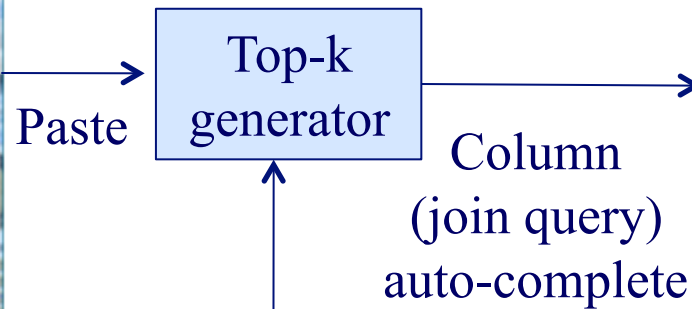


Graph of potential  
joins & costs

# Learning / Suggesting a Query (Details in Paper)

PR-String	PR-Street	PR-City	PR-Zip	Data Type	Column Name
Lyons Creek Middle	4133 Sol Press Blvd.	Coconut Creek	33073		
Coral Glades High	2700 Sportsplex Dr.	Coral Springs	33073		
Monarch High School	5050 Willes Rd.	Coconut Creek	33073		
Pompano Beach Hi.	601 NE 12th Ave.	Pompano Beach	33060		
Park Lakes Element.	3925 N State Road 7	Lauderdale Lakes	33319		
Arthur Ash Middle Sc.	1701 NW 23 Ave.	Fort Lauderdale	33311		
Plantation Elements	651 NW 42 Ave.	Plantation	33317		
Fox Trail Elementary	1250 Nob Hill Rd.	Davie	33324		
Falcon Cove Middle	4251 Bonaventure B.	Weston	33332		
Glazer Trail Middle S.	19200 Sheridan St.	Pembroke Pines	33331		
New Renaissance	10701 Miramar Blvd.	Miramar	33025		
Watkins Elementary	3520 SW 52 Ave.	Pembroke Park	33023		

Columns pasted from different sources



Source1	Source2	PR-String	PR-Street	PR-City	PR-Zip	Data Type	Column Name
		Lyons Creek Middle	4133 Sol Press Blvd.	Coconut Creek	33073		
		Coral Glades High	2700 Sportsplex Dr.	Coral Springs	33073		
		Monarch High School	5050 Willes Rd.	Coconut Creek	33073		
		Pompano Beach Hi.	601 NE 12th Ave.	Pompano Beach	33060		
		Park Lakes Element.	3925 N State Road 7	Lauderdale Lakes	33319		
		Arthur Ash Middle Sc.	1701 NW 23 Ave.	Fort Lauderdale	33311		
		Plantation Elements	651 NW 42 Ave.	Plantation	33317		
		Fox Trail Elementary	1250 Nob Hill Rd.	Davie	33324		
		Falcon Cove Middle	4251 Bonaventure B.	Weston	33332		
		Glazer Trail Middle S.	19200 Sheridan St.	Pembroke Pines	33331		
		New Renaissance	10701 Miramar Blvd.	Miramar	33025		
		Watkins Elementary	3520 SW 52 Ave.	Pembroke Park	33023		

Import Integrate Publish

Join Link

SOURCE1

Name Monarch High School

Street 5050 Willes Rd.

City Coconut Creek

City Coconut Creek

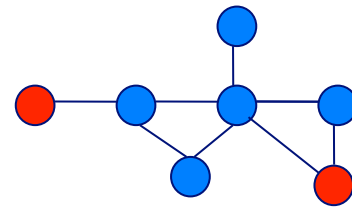
ZIPCODES

Street 5050 Willes Rd.

City Coconut Creek

Zip 33073

Feedback based on tuple provenance



Adjusted weights

MIRA-based cost learner

# Related Work

---

Programming by demonstration [Cypher+93], [Lau 01]

- esp. Karma [Tucinda+07]

Dataspaces, best-effort integration

- see Franklin, Halevy, Maier VLDB 08 survey

User-driven data integration

- Potluck [Huynh+07], Q [Talukdar+08]

Wrapper induction (source extraction)

- Lixto, [Ashish+97], [Kushmerick+97], [Muslea+01], [Gazen&Minton 06]

Provenance / lineage [Cui 01], [Buneman+01], [Green+07]

- for debugging [Chiticariu & Tan 06]

# Conclusions & Future Work

---

Smart copy and paste is a new way of thinking about task-driven data integration

- Lightweight, seamless combination of design-time and runtime components – “spreadsheet of integration”
- Learns source structure, model
- Suggests and learns the integration query through feedback
- Knits together data and queries/sources via provenance

CopyCat validates basic architecture, but still much to be done!

- Scale-up – how do the UI, feedback process scale to many alternatives?
- Complex functions – how to easily incorporate?
- Data cleaning
- Directly integrating visualization (cf. Jeff Heer’s keynote talk)