CS520: KNOWLEDGE GRAPHS

Data Models, Knowledge Acquisition, Inference, Applications

Lectures and Invited Guests

Spring 2021, Tu/Thu 4:30-5:50, cs520.Stanford.edu

**Learn about the basic concepts,
latest research & applications**

Knowledge Graphs Seminar

- What is a Knowledge Graph?
- How to Create a Knowledge Graph?
- How to Reason with and Access Knowledge Graphs?
- Applications

Knowledge Graphs Seminar

- What is a Knowledge Graph?
- How to Create a Knowledge Graph?
    - How to design the schema?
    - Creating a KG from data
    - Create a KG from text and images
- How to Reason with and Access Knowledge Graphs?
- Applications

Knowledge Graphs

How to Create a Knowledge Graph from Structured Data?

Outline

- Overview
- Schema Mapping
- Record Linkage
- Summary

Overview

- Large organizations have lot of internal data
  - Customer profiles
  - Product offerings
  - Transactions
- They also consume external data from third party providers
  - News reports
  - Funding decisions
  - Supplier relationships

Overview

- 360-degree view of a customer

Financial News

Who supplies to who?

FACTSET

Who are my customers?

INTERNAL
COMPANY
DATA

Acma Inc filed for bankruptcy
Suppliers to Acma are facing financial distress
Stress propagates recursively in the supply chain
Credit officers must be alerted
Risk analysis must take this into account

Overview

- Knowledge graph by integrating external and internal data
  - Schema design
    - Relating the schema of sources to the knowledge graph schema
  - Record linkage
    - Recognizing if two instances refer to the same object in the real-world

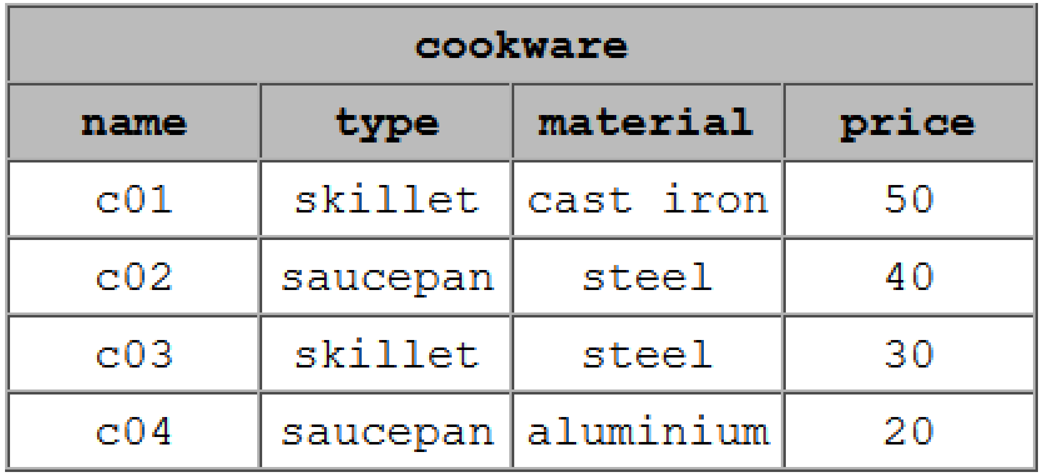
Schema Mapping

- Practical challenges
- Example of schema mapping
- Specifying schema mapping
- Bootstrapping schema mapping

Practical Challenges

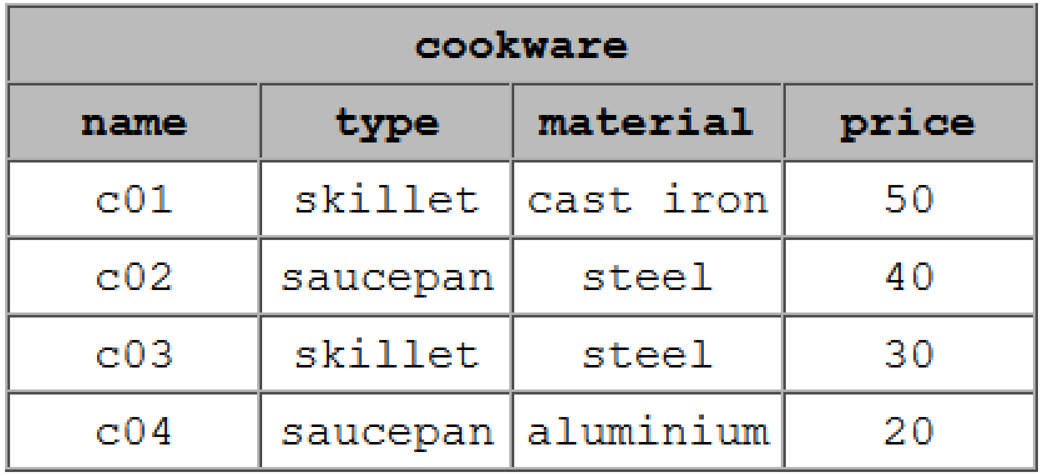
- Difficult to understand schema
    - Large tables, unhelpful names (e.g., segment1, segment2, etc.)
- Mappings are not always one-to one
    - Need to apply business logic
- Training data not available
    - Data for schema mappings is even more scarce

Example Schema Mapping

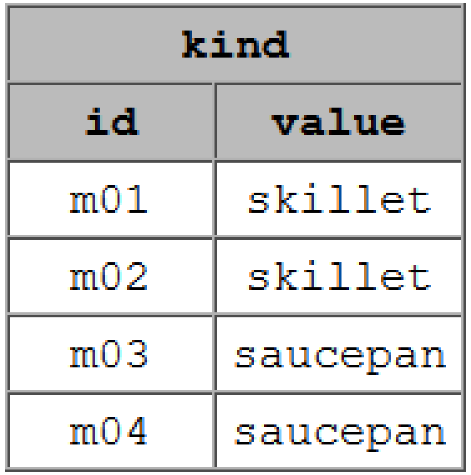


cookware			
name	**type**	**material**	**price**
c01	skillet	cast iron	50
c02	saucepan	steel	40
c03	skillet	steel	30
c04	saucepan	aluminium	20

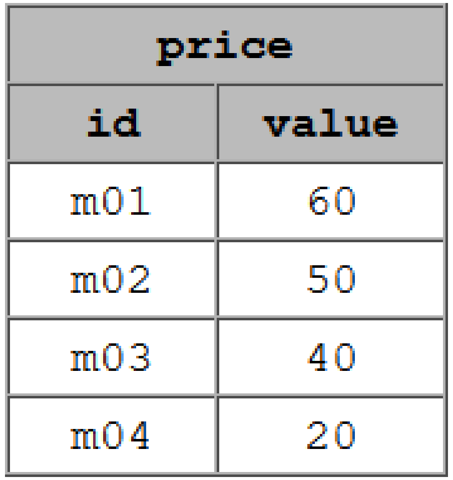
Example Schema Mapping



cookware			
name	type	material	price
c01	skillet	cast iron	50
c02	saucepan	steel	40
c03	skillet	steel	30
c04	saucepan	aluminium	20

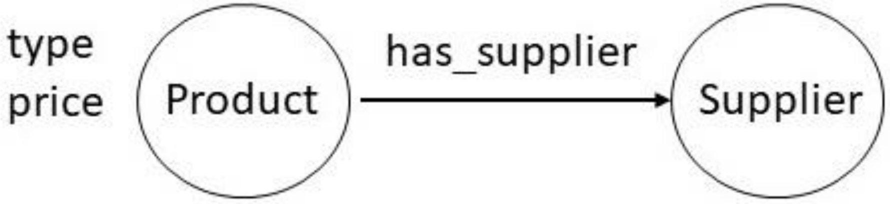


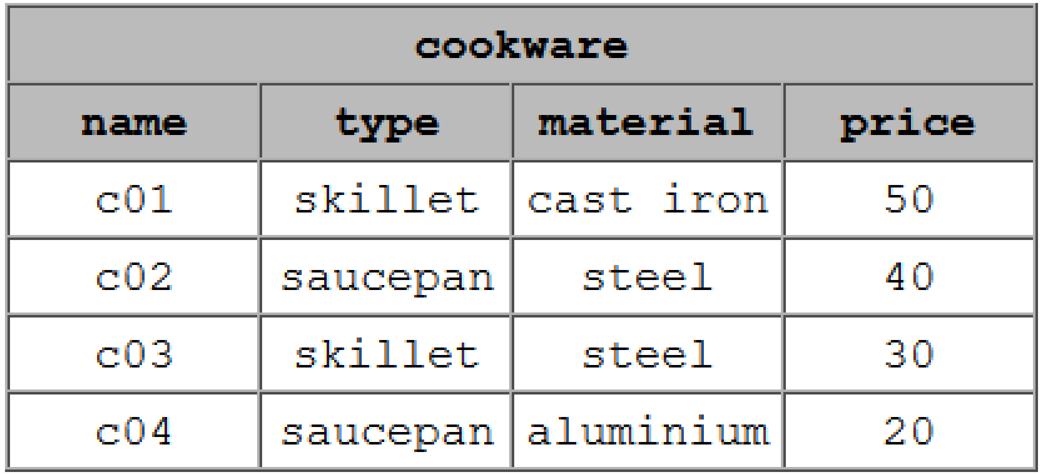
kind	
id	value
m01	skillet
m02	skillet
m03	saucepan
m04	saucepan



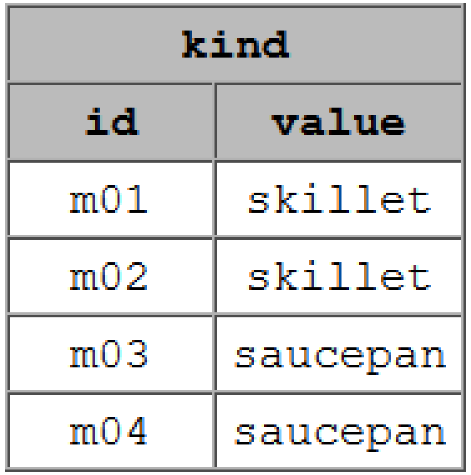
price	
id	value
m01	60
m02	50
m03	40
m04	20

Example Schema Mapping

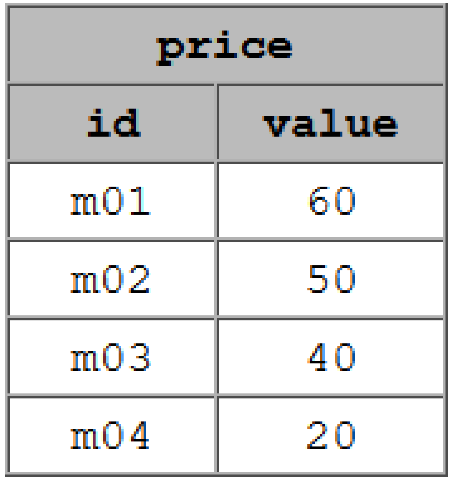




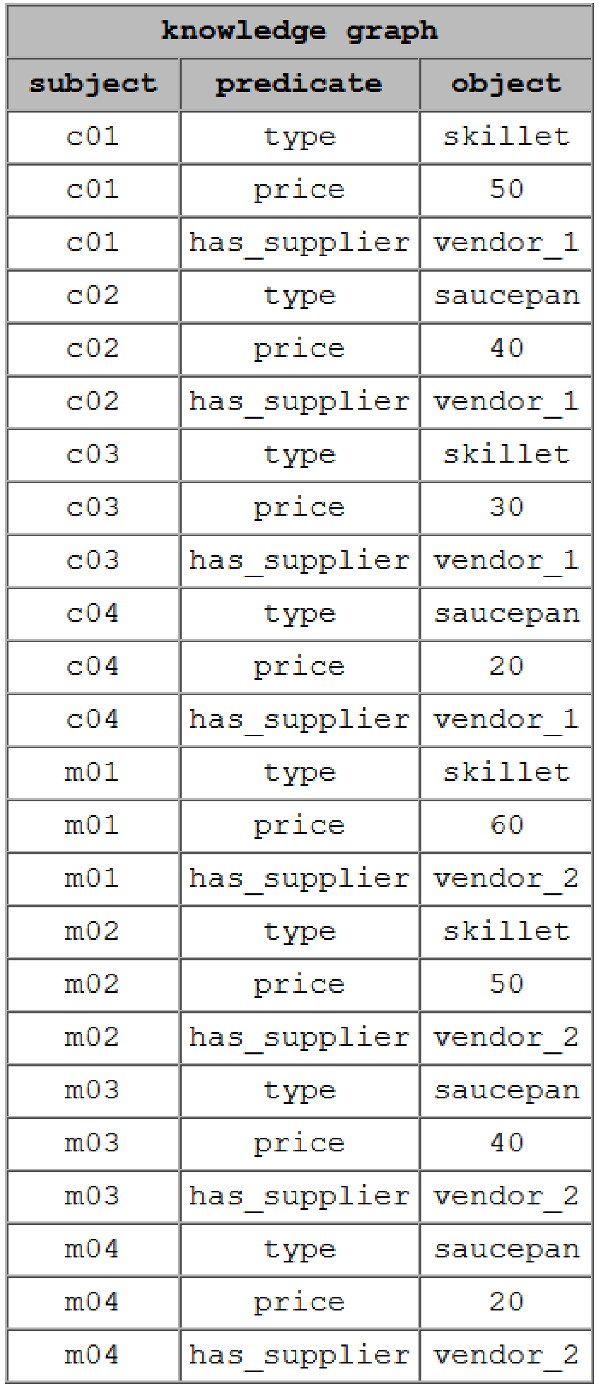
cookware			
name	**type**	**material**	**price**
c01	skillet	cast iron	50
c02	saucepan	steel	40
c03	skillet	steel	30
c04	saucepan	aluminium	20



kind	
id	**value**
m01	skillet
m02	skillet
m03	saucepan
m04	saucepan

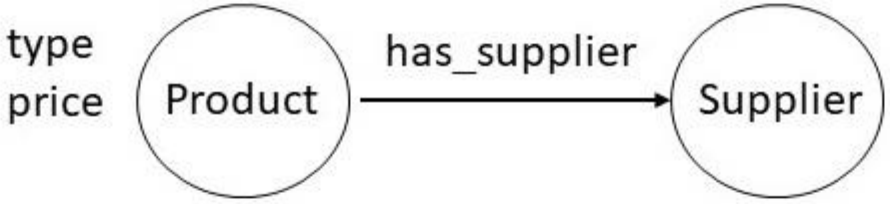


price	
id	**value**
m01	60
m02	50
m03	40
m04	20

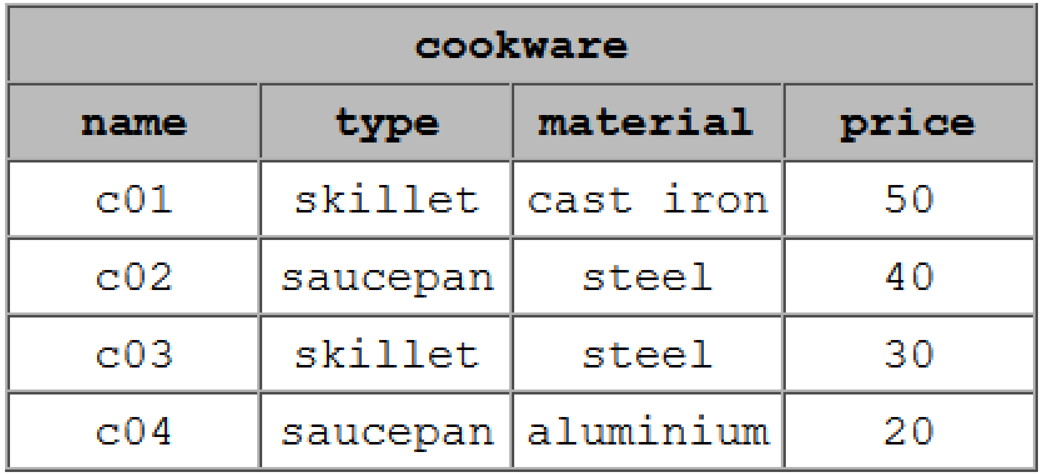


knowledge graph		
subject	**predicate**	**object**
c01	type	skillet
c01	price	50
c01	has_supplier	vendor_1
c02	type	saucepan
c02	price	40
c02	has_supplier	vendor_1
c03	type	skillet
c03	price	30
c03	has_supplier	vendor_1
c04	type	saucepan
c04	price	20
c04	has_supplier	vendor_1
m01	type	skillet
m01	price	60
m01	has_supplier	vendor_2
m02	type	skillet
m02	price	50
m02	has_supplier	vendor_2
m03	type	saucepan
m03	price	40
m03	has_supplier	vendor_2
m04	type	saucepan
m04	price	20
m04	has_supplier	vendor_2

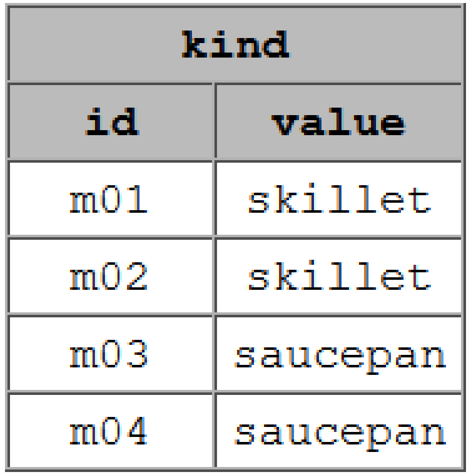
Example Schema Mapping



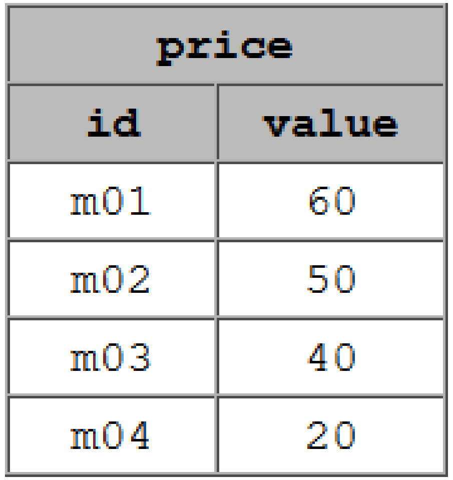
knowledge_graph(ID,type,Type) :- cookware(ID,TYPE,MATERIAL,PRICE)
knowledge_graph(ID,price,PRICE) :- cookware(ID,TYPE,MATERIAL,PRICE)
knowledge_graph(ID,has_supplier,vendor_1) :- cookware(ID,TYPE,MATERIAL,PRICE)



cookware			
name	**type**	**material**	**price**
c01	skillet	cast iron	50
c02	saucepan	steel	40
c03	skillet	steel	30
c04	saucepan	aluminium	20

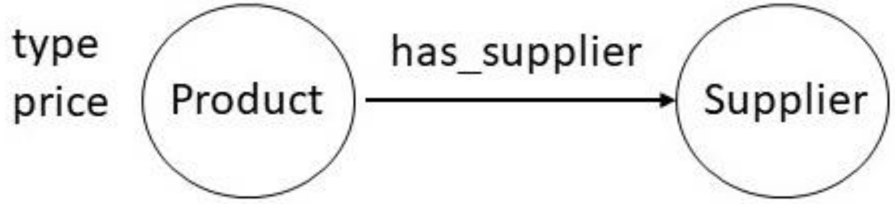


kind	
id	**value**
m01	skillet
m02	skillet
m03	saucepan
m04	saucepan

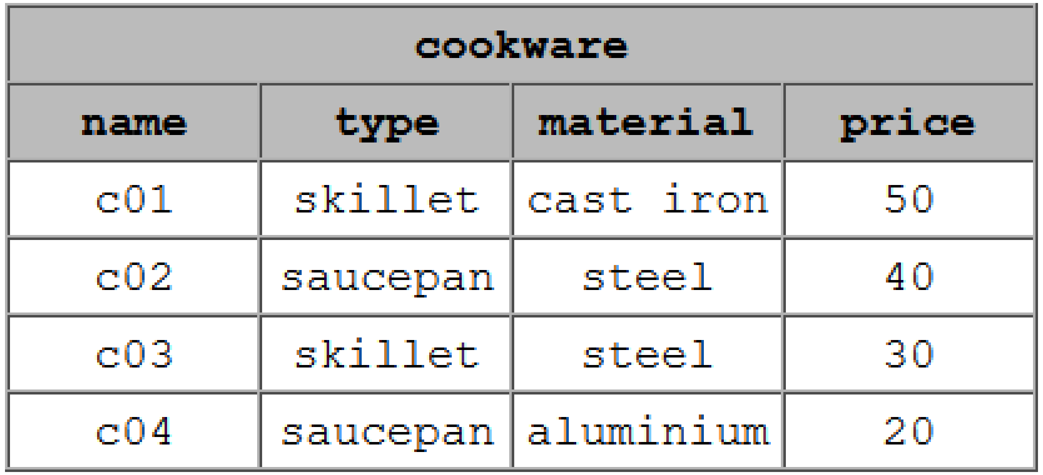


price	
id	**value**
m01	60
m02	50
m03	40
m04	20

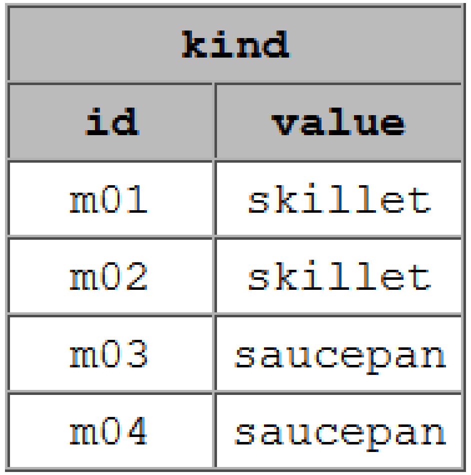
Example Schema Mapping



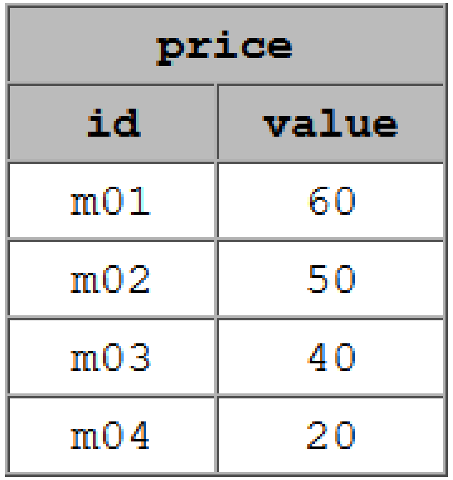
knowledge_graph(ID,type,Type) :- kind(ID,TYPE)
knowledge_graph(ID,price,PRICE) :- price(ID,PRICE)
knowledge_graph(ID,has_supplier,vendor_2) :- kind(ID,TYPE)



cookware			
name	**type**	**material**	**price**
c01	skillet	cast iron	50
c02	saucepan	steel	40
c03	skillet	steel	30
c04	saucepan	aluminium	20



kind	
id	**value**
m01	skillet
m02	skillet
m03	saucepan
m04	saucepan



price	
id	**value**
m01	60
m02	50
m03	40
m04	20

Bootstrapping Schema Mapping

- Linguistic Mapping
- Mapping based on instances
- Mapping based on constraints

Bootstrapping Schema Mapping

- Linguistic Techniques
    - Leverage the name
        - Best solution is to use IRIs and sameAs links
    - Stemming, Synonym, Hypernym
        - Cname and Customer Name
        - Automobile and Vehicle
        - Book and Publication
    - Common substrings/pronunciation
        - Amount Received/Amount Receivable
        - Bell vs Belle
    - Leverage documentation string
        - Extract keywords, and check semantic similarity

Bootstrapping based on Instances

- Examine the data
    - If we can recognize the data contain phone number, zip code, ISBN, SSN, Date that can provide strong guidance for which attributes can match

Bootstrapping based on Constraints

- Leverage the constraints
    - Value range constraints, uniqueness, optionality, cardinality

Bootstrapping Schema Mapping

- Bootstrapping results
  - are inexact
  - need human verification
- Can save some effort
- Lead to a better story

Outline

- Overview
- Schema Mapping
- Record Linkage
- Summary

Record Linkage

- An Example Problem
- An approach to record linkage
    - Blocking followed by Matching
        - Random forests
        - Active learning
        - Rule application

Example

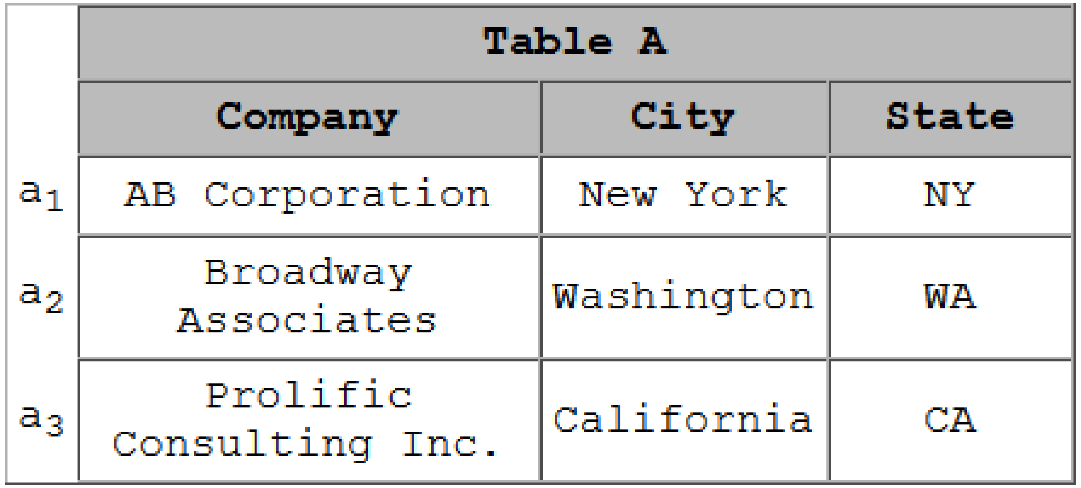


	Table A		
	Company	City	State
a_1	AB Corporation	New York	NY
a_2	Broadway Associates	Washington	WA
a_3	Prolific Consulting Inc.	California	CA

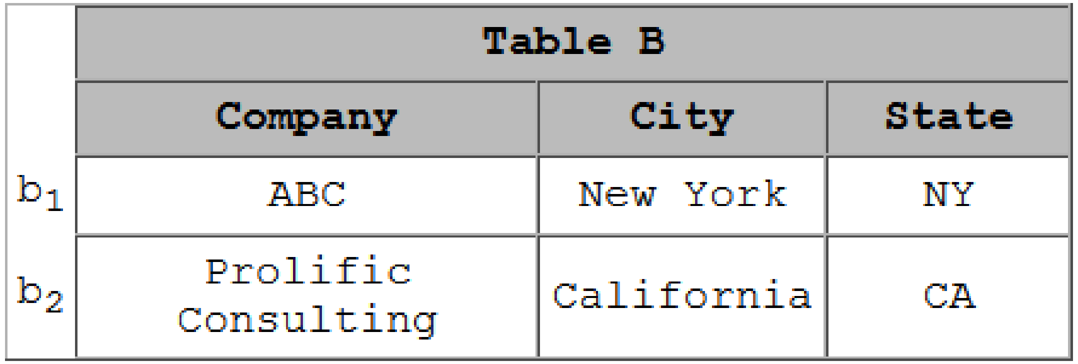


	Table B		
	Company	City	State
b_1	ABC	New York	NY
b_2	Prolific Consulting	California	CA

a1=b1
a3=b2

Inexact Inference

In practice, millions of records

Approach

- Blocking Followed by Matching

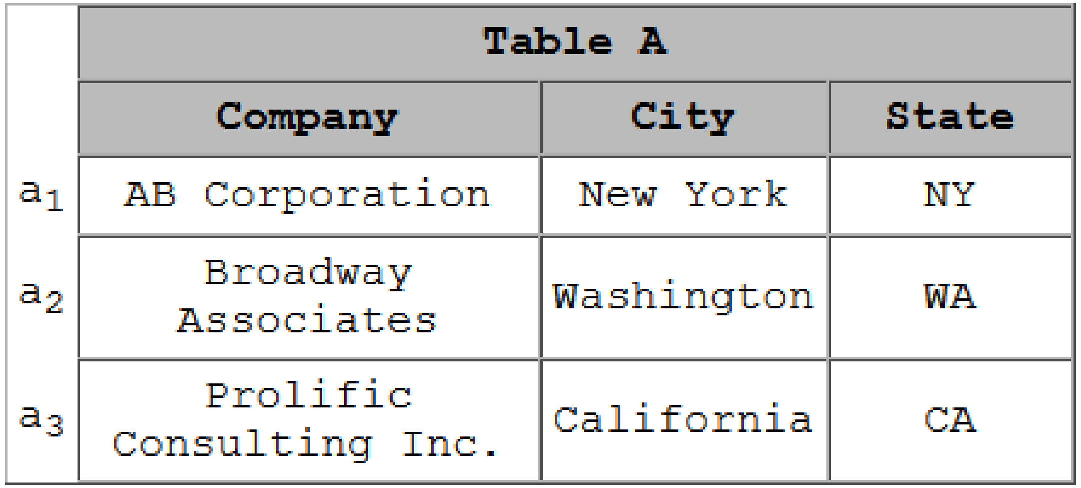


	Table A		
	Company	City	State
a_1	AB Corporation	New York	NY
a_2	Broadway Associates	Washington	WA
a_3	Prolific Consulting Inc.	California	CA

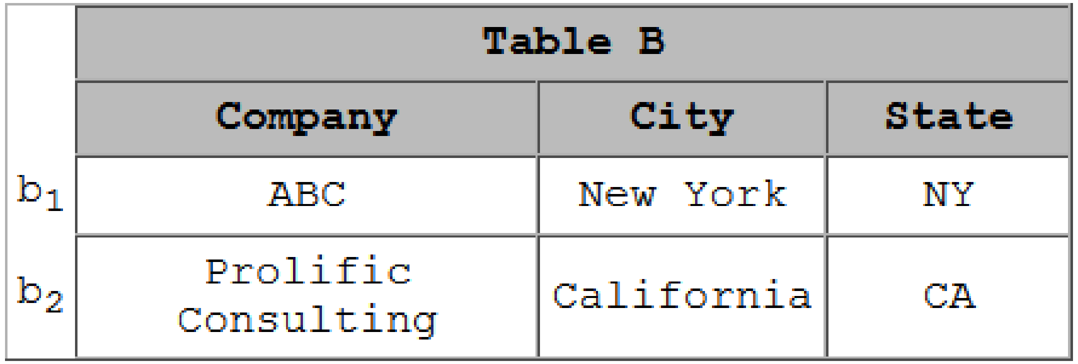


	Table B		
	Company	City	State
b_1	ABC	New York	NY
b_2	Prolific Consulting	California	CA

Blocking
<a1,b1>
<a3,b2>

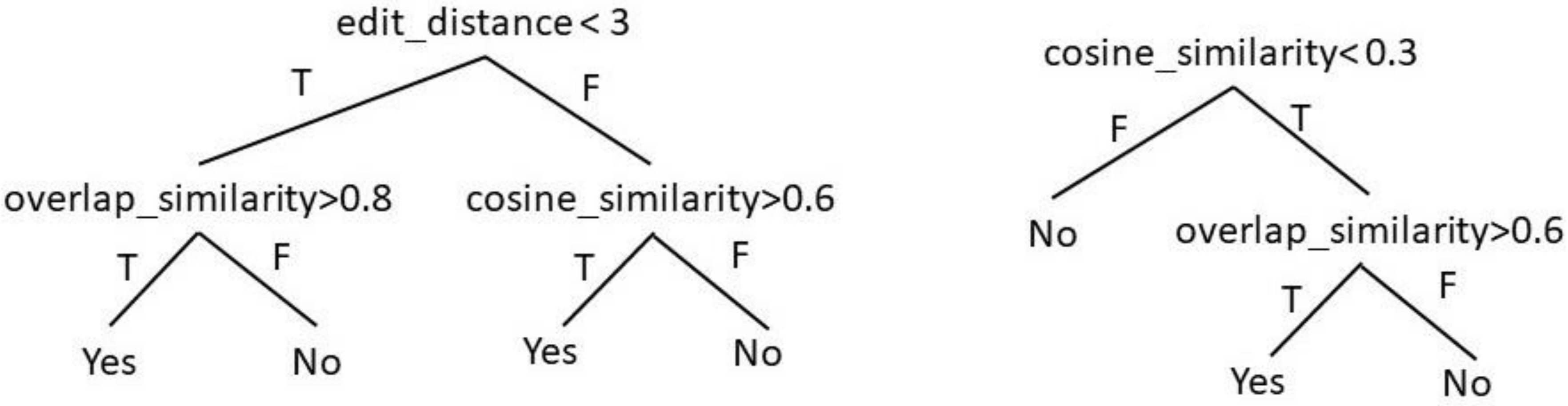
Overview of the algorithm

- Express the blocking/matching rules as a random forest
- Use Active Learning to build the random forest
- Efficient application of rules through indexing

Random Forest

- Consists of a set of set of rules
- Each rule selects records based on (inexpensive) similarity functions
    - Edit distance
    - Overlap similarity
    - Cosine similarity

Random Forest



r_1: (edit_distance ≥ 3) and (cosine_similarity > 0.6) → match

r_2: (edit_distance < 3) and (overlap_similarity > 0.8) → match

r_3: (cosine_similarity ≥ 0.3) and (overlap_similarity > 0.6) →match

Random Forest

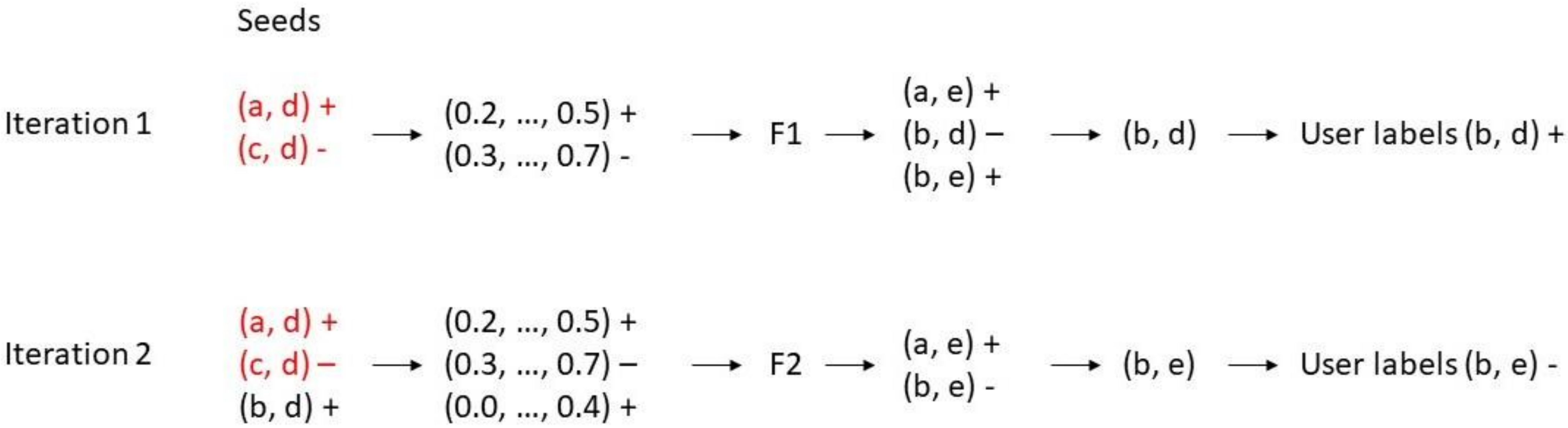
- General principles for selecting similarity functions
  - Numeric-valued attributes such as age, weight, price, etc.
    - exact match, absolute difference, relative difference, and Levenstein distance
  - String-valued attributes
    - edit distance, cosine similarity, Jaccard similarity, and TF/IDF functions.

Active Learning

- Randomly select pairs from the two data sets
    - Ask the users to label them
- Use similarity functions to obtain features
- Learn random forest
- Apply the learned rules to new selected pairs
    - Evaluate the rules
- Iterate

Active Learning

- Source 1: (a,b,c) Source 2: (d,e)



Seeds

Iteration 1: (a, d) + (c, d) - → (0.2, ..., 0.5) + (0.3, ..., 0.7) - → F1 → (a, e) + (b, d) – (b, e) + → (b, d) → User labels (b, d) +

Iteration 2: (a, d) + (c, d) – (b, d) + → (0.2, ..., 0.5) + (0.3, ..., 0.7) – (0.0, ..., 0.4) + → F2 → (a, e) + (b, e) - → (b, e) → User labels (b, e) -

Active Learning

- Once the learning algorithm converges, present the rules to the user
- Retain the rules validated by the user

Rule Application

- Leverage indexing for efficient application of rules
  - Suppose we need to check Jaccard similarity to movie “Sound of Music”
  - If the similarity needs to be greater than 0.7, we need to consider only those movies with length between 3*0.7, and 3/0.7, ie, between 2 and 4
  - An index on the length of movies can help us select which movie records to consider

Blocking vs Matching

- Same algorithmic outline is used except
  - The matching rules are more exact/price
  - The matching is usually verified through human intervention

Summary

- Creating KG from structured sources is a data integration problem
  - Target schema is a knowledge graph
- Schema Mapping Problem
  - Even though bootstrapping is possible, but it is still labor intensive
- Record Linkage Problem
  - Efficiency is a key consideration
  - Two-step approach with blocking and matching
    - Leverage random forests and active learning

Structured Data
Cleaning

Ihab Ilyas
U. Of Waterloo

Lauren Orr
Self-Supervised Entity Disambiguation

Mayank Kejriwal
Entity Resolution in Web Scale KGs