W4. Lecture Notes — By Junyi

SUMMARY

- ▼ creating KG from structured data → data integration problem
- ▼ how the data elements from a new data source should be added to the knowledge graph → schema mapping problem
    - labor intensive! though bootstrapping is possible
- ▼ Recognizing if two instances refer to the same object in the real-world → record linkage problem
    - key: efficiency
    - toe-step approach with blocking and matching
        - leverage random forests and active learning

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

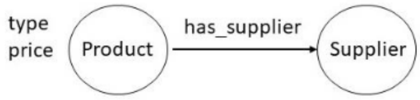
- ▼ Practical challenges
    - hard to understand schema
        - large tables, unhelpful names
    - Mappings are not always one-to-one → apply business logic

- Training data (for schema mapping) nor available

▼ Example of schema mapping

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



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(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)

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)

- ▼ Specifying schema mapping

- ▼ Bootstrapping schema mapping
  - ▼ linguistic mapping
    - leverage the name
      - use IRIs and sameAs links
    - stemming, synonym, hypernym
      - Cname and customer name
      - automobile and vehicle
      - book and publication
    - common substrings/pronunciation

        - amount received/amount received
        - bell vs belle
    - leverage documentation string
        - extract keywords, and check semantic similarity
- ▼ mapping based on instances
    - examine the data
        - recognize pattern: phone number, zip code, ISBN, SSN, Date → which attributes can match
- ▼ mapping based on constrains
    - leverage the constraints
    - features
        - bootstrapping results
            - are inexact
            - need human verification
        - save some effort
        - lead to a better story

Record Linkage

▼ 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

- 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

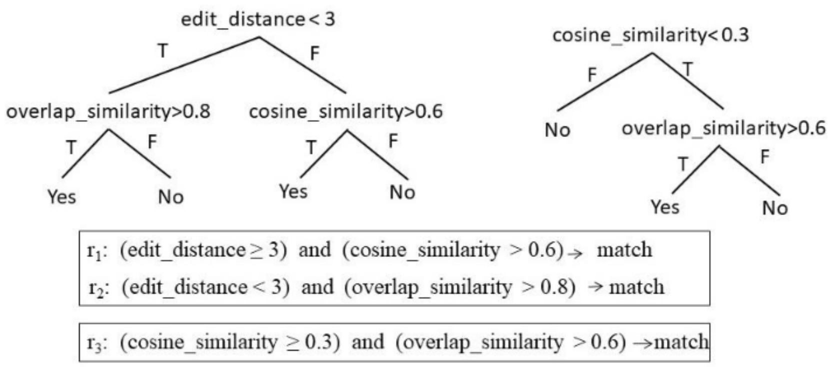
	Table B		
	Company	City	State
b_1	ABC	New York	NY
b_2	Prolific Consulting	California	CA

Blocking
<a1,b1>
<a3,b2>

- ▼ An approach to record linkage
  - ▼ blocking followed by matching
    - random forests
    - active learning
    - rule application
  - Algorithm
    - ▼ Overview
      - express the blocking/matching rules as random forest
      - use active learning to build the random forest
      - efficient application of rules through indexing
    - Techniques
      - ▼ random forest
        - consist of a set of set of rules
        - each rule selects records based on (inexpensive) similarity functions
          - similarity functions
            - edit distance
            - overlap similarity
            - cosine similarity
            - others
          - general principles for selecting similarity functions
            - numeric-valued attributes → exact match, absolute difference, relative difference, and Levenstein distance

- String-valued attributes → edit distance, cosine similarity, Jaccard similarity, and TF/IDF functions

- illustration

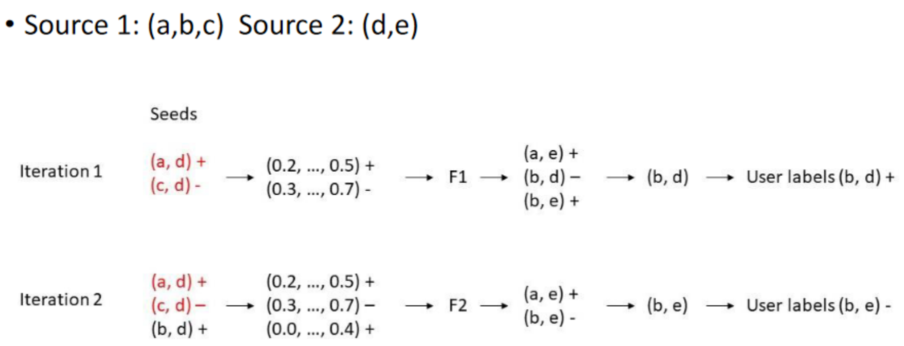


▼ Active learning

▼ algorithm

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

- illustration



- Rule application

▼ Blocking vs Matching

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