A Query-Driven Approach to Entity Resolution based on Data Ordering

Relatori: Prof. Sonia Bergamaschi
Prof. Felix Naumann

Candidato: Giacomo Amici

Finalità Tesi

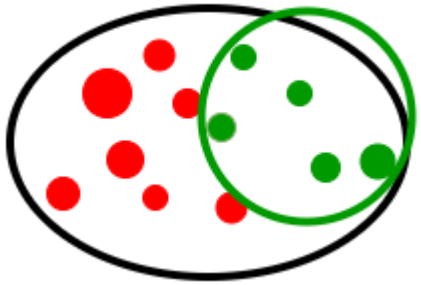
- Sviluppare un metodo **progressivo** e **query-driven** per l'Entity Resolution + Data ordering

Progressivo:

- Ottenere entità risolte ed ordinate prima della completa pulizia del dataset

Query-Driven:

- Considerare solo la porzione di dataset che ci interessa



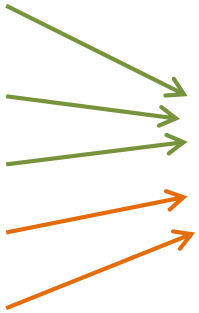
Outline

1. Entity Resolution
2. Traditional vs progressive method
3. OrdER: algorithm
4. OrdER: extensions
5. Results

Entity Resolution

Processo di identificazione e unione dei record che rappresentano lo stesso oggetto del mondo reale

Smartphones
i-Phne 10
i-Phone new
i-Phoone X
Galxy S10
Galaxi S1.0



Smartphones
i-Phone 10
Galaxy S10

ER per: lotta al terrorismo, identificazione spam, salute pubblica, lettura automatica…

Data Redundancy e Dirty Data

- Conoscenza come potente risorsa
- Conoscenza sotto forma di dati (sporchi)

Integrazione da sorgenti diverse:

- Dati duplicati
- Dati incompleti
- Dati disordinati

Metodo tradizionale vs progressivo

Approccio tradizionale:

Data Cleaning (ER)

Data Ordering

- - Costo computazionale alto
- - Risultati al termine

Progressivo

Data cleaning (ER)

Data Ordering

- - Costo computazionale basso
- - Risultati progressivi

OrdER

Idea: applicare Query SQL a dataset

SELECT Name, MIN(Price) as mp
FROM Smartphones
GROUP BY _
ORDER BY mp DESC

Smartphone	Price
i-Phne 10	15
i-Phone new	30
i-Phoone X	20
Galaxy S10	40
Galaxi S1.0	20

1. Galaxy S10, 20
2. i-Phone 10, 15

OrdER

Utilizzo di 2 liste:

Entity Resolution ⟶ Solving List

Data Ordering ⟶ Ordering List

Records	Price
a	20
b	30
c	80
d	70
e	60
f	40
g	10
h	50

Solving key

SL:

30	40	10	50	70	60	80	20
b	f	g	h	d	e	c	a

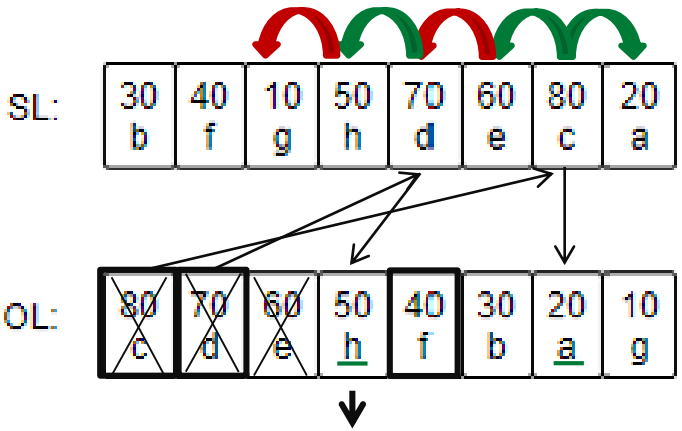
Ordering key
(ORDER BY)

OL:

80	70	60	50	40	30	20	10
c	d	e	h	f	b	a	g

OrdER

Merging function: MIN
Sorting order: Desc



bounds: [50, 20]

Algorithm:

1) Take next record in OL and find it in SL
2) While value in OL < *bound:*
   emit entity
3) Compare the neighbors of the record in SL
   - Match: compare next neighbors
   - Not Match: stop
4) Take the minimum value as the new bound of the entity
5) Forget greater values of the same entity

OrdER-HAVING

Problema: se volessimo applicare un filtro ai record?

WHERE

SELECT Name, MIN(Price) as mp
FROM Smartphones
WHERE Name Like «i-Phone»
ORDER BY mp DESC

i-Phon	60
i-Phone	70
iPhone	40
i-Phne	20

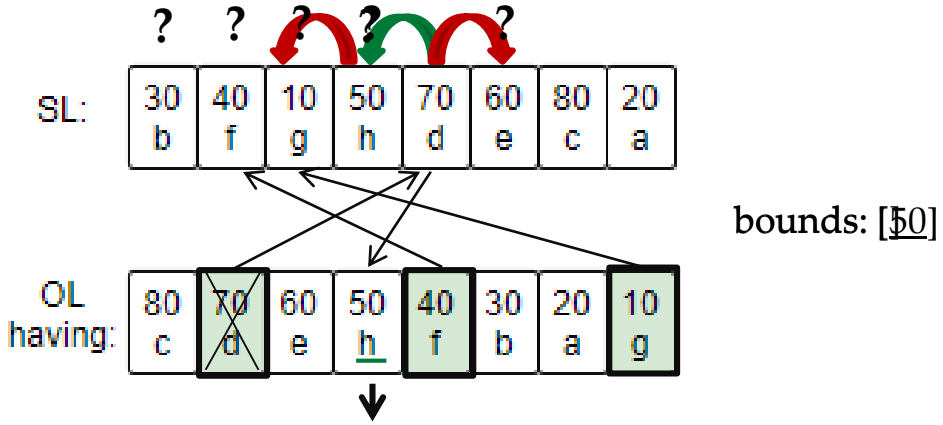
OrdER
HAVING

SELECT Name, MIN(Price) as mp
FROM Smartphones
GROUP BY _
HAVING Name Like «i-Phone»
ORDER BY mp DESC

i-Phon	60
i-Phone	70
iPhone	40
i-Phne	20

OrdER-HAVING

Merging function: MIN
Sorting order: Desc



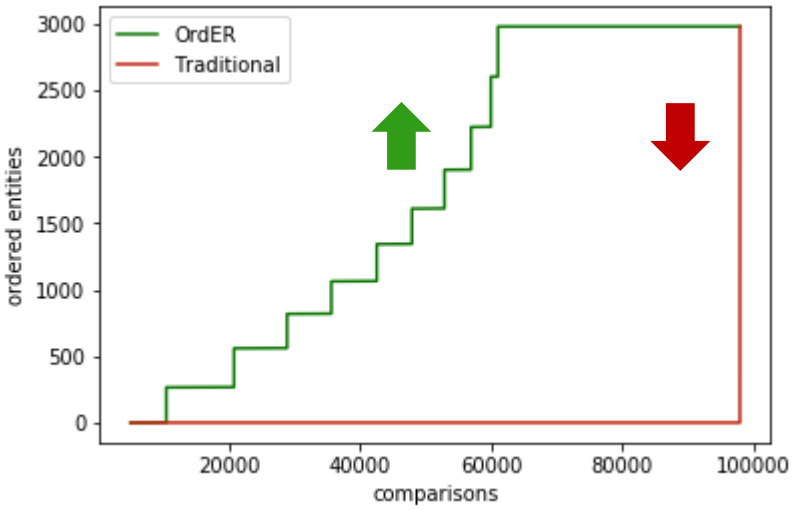
Algorithm:

1) Take next record in OL that satisfies HAVING clause and find it in SL
2) If SL neighborhood's least value is greater than *bound*:
   Compare neighbors
   Else: go to 1)
3) Take the minimum value as the new bound of the entity
4) Forget greater values of the same entity
5) If OL terminated:
   1. emit entity
   2. start over

Risultati

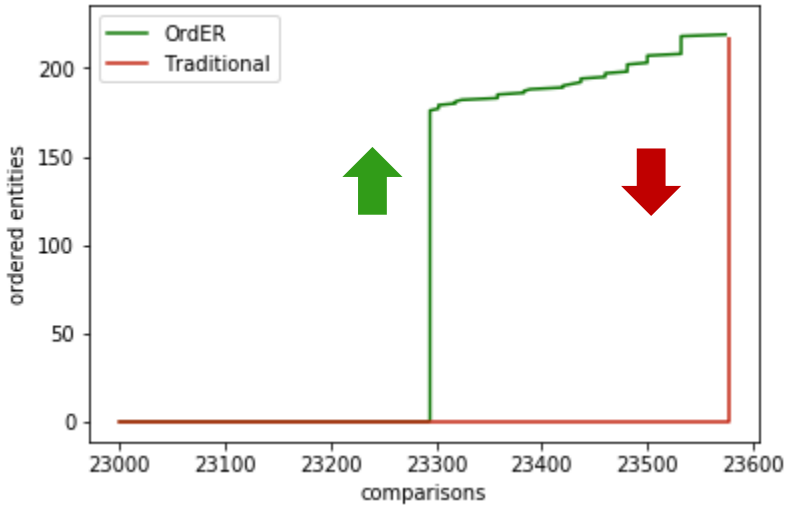
DBLP-ACM datasets (Clean-Clean)

Results	traditional	OrdER
Entities	4112	4112
Comps	97,990	63,691



CORA dataset (Dirty)

Results	traditional	OrdER
Entities	222	223
Comps	182,850	23,800



Sviluppi futuri…

- Applicazione su dataset completamente non strutturati
- Estensione a merging function non-boundable

Work in Progress

OrdER - Multiple Solving List

Problema: alcuni record potrebbero non essere identificati

Estendere OrdER con multiple Solving Lists