Michael J. Cafarella Alon Halevy Nodira Khoussainova

Data Integration for the Relational Web *

GRUPPO 18

Alessandro Gottardi
Vincenzo Laudizio
Silvia Umiliacchi

* Work done while all authors were at Google, Inc.

Introduzione

Il Web è un contenitore infinito di dati "raw". La possibilità di COMBINARLI e RIPROPORLI presenta enormi potenzialità.

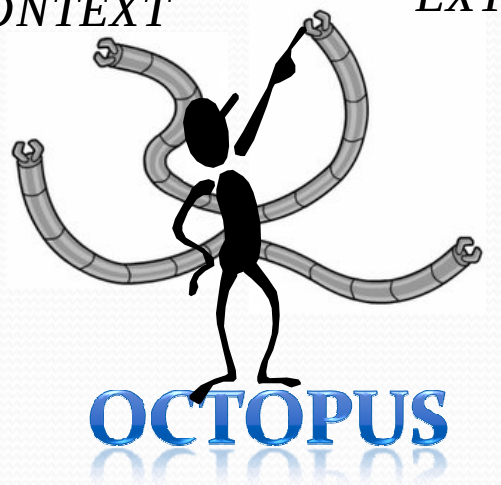
tuttavia

Integrare dati è un'operazione complessa e onerosa con i sistemi attuali in quanto:

- Assumono che le sorgenti rilevanti siano definite a priori.
- Ipotizzano di avere dati stabili.
- Ipotizzano di avere dati puliti e pronti all'uso.
- La semantica dei dati è spesso implicita nella pagina in cui essi sono presenti.

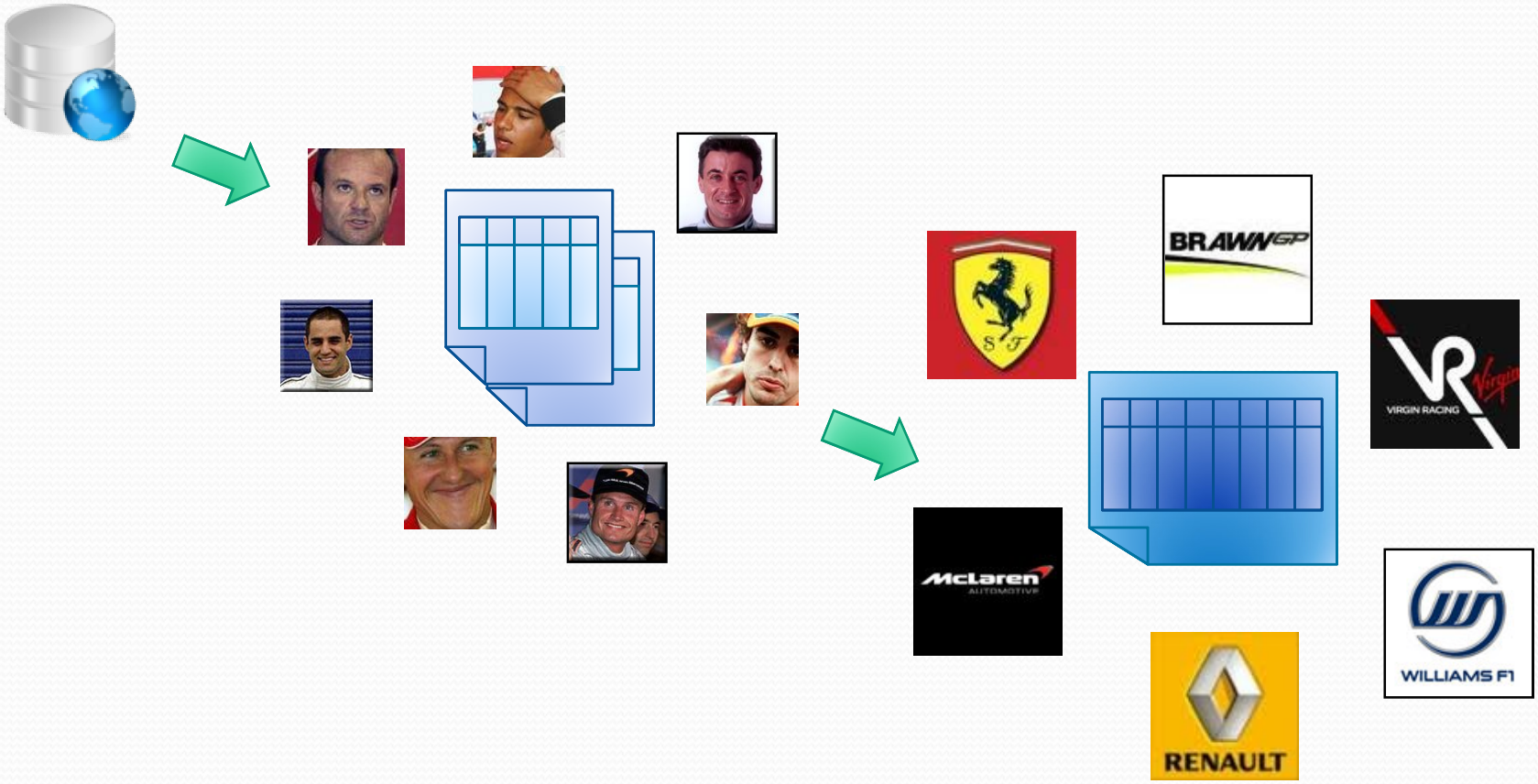
CONTEXT *EXTEND*

SEARCH .....



Introduzione

“Piloti di Formula 1 e scuderie per cui hanno gareggiato nella loro carriera”



Octopus

CARATTERISTICHE

- Fornisce degli operatori per aiutare l'utente nel processo di integrazione dei dati presenti nel web:
  - *SEARCH*
  - *CONTEXT*
  - *EXTEND*
- Best-effort operators: non un solo output corretto a fronte di un dato input, ma quello di qualità massima (simile al ranking effettuato dai motori di ricerca)
- L'utente può interagire con il sistema:
  - modificare l'output
  - aggiungerne altri
  - combinare gli operatori

Utilizzo degli operatori

classifica mondiale F1 2008

SEARCH("piloti di Formula 1")

Lewis Hamilton	Inghilterra
Felipe Massa	Brasile
Sebastian Vettel	Germania

project(c=3)

classifica mondiale F1 2009

Jenson Button	Inghilterra	19/01/1980
Sebastian Vettel	Germania	03/07/1987
Rubens Barrichello	Brasile	23/05/1972

CONTEXT

Lewis Hamilton	Inghilterra	2008
Felipe Massa	Brasile	2008
Sebastian Vettel	Germania	2008

CONTEXT

Jenson Button	Inghilterra	2009
Sebastian Vettel	Germania	2009
Rubens Barrichello	Brasile	2009

union

Lewis Hamilton	Inghilterra	2008
Felipe Massa	Brasile	2008
Sebastian Vettel	Germania	2008
Jenson Button	Inghilterra	2009
Sebastian Vettel	Germania	2009
Rubens Barrichello	Brasile	2009

EXTEND(c1, "scuderie")

Lewis Hamilton	Inghilterra	2008	Mclaren Mercedes
Felipe Massa	Brasile	2008	Sauber, Ferrari
Sebastian Vettel	Germania	2008	Sauber, Toro Rosso, Red Bull
Jenson Button	Inghilterra	2009	Williams, Renault, Honda, Brawn GP
Sebastian Vettel	Germania	2009	Sauber, Toro Rosso, Red Bull
Rubens Barrichello	Brasile	2009	Jordan, Stewart, Ferrari, Honda, Brawn GP

Estrarre informazioni dal Web

- Come estrarre dati e tabelle rilevanti da una pagina web?

  Il problema consiste nel separare tabelle destinate al layout da tabelle con contenuto informativo

  ➡ Si usano tecniche descritte dagli stessi autori nel paper "Uncovering the Relational Web":

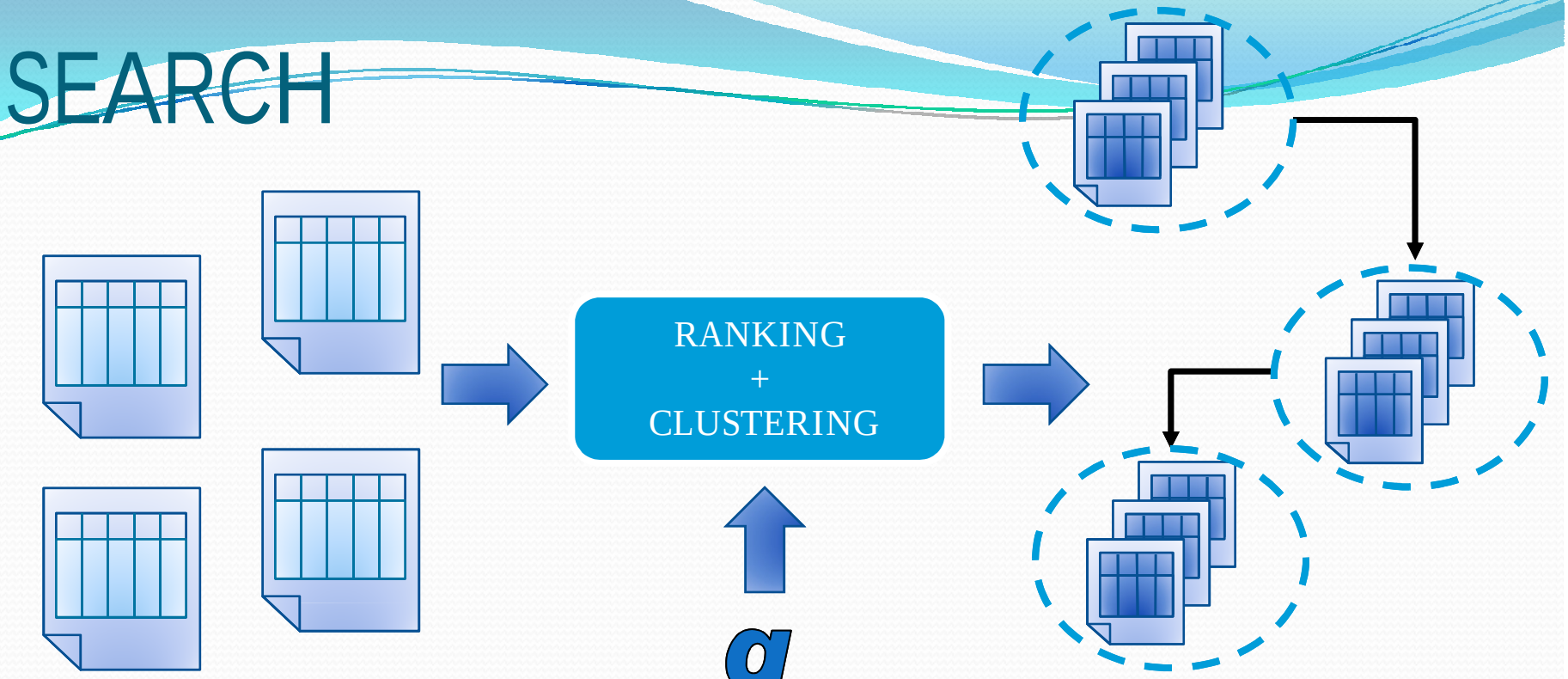
  - Filtrare i contenuti di una pagina HTML

    ➡ WebTable System

  - Estrapolare tabelle relazionali basandosi su tecniche statistiche

    ➡ Metadata Recovery

SEARCH



insieme di tabelle estratte dal web

query

lista di cluster di tabelle ordinata per rilevanza secondo *q*

Ranking: consente di recuperare rapidamente le tabelle più rilevanti per l'utente.

Clustering: consente di accorpare le tabelle simili in modo che siano unificabili.

SEARCH - *SimpleRank*

Sottopone la query *q* ad un motore di ricerca. Estrae le tabelle ordinatamente dagli indirizzi restituiti.

- Semplice ☺
- Il search engine effettua il ranking sulla base di tutto il contenuto di una pagina:
  non sempre a pagine rilevanti corrispondono tabelle rilevanti. ☹
- In caso vi siano più dataset all'interno di una stessa pagina il ranking segue l'ordine di presentazione ☹

SEARCH - *SCPRank*

Un'idea migliore: classificare non l'intera pagina, ma le tabelle estratte.

SCPRank: data la query *q* calcola per ogni cella *c* di una tabella un valore di Simple Conditional Probability. Lo score di una tabella è pari al massimo score di una delle sue colonne, calcolato come somma dei valori di SCP per cella.

SCP(*q*,*c*) misura quanto è più probabile che un documento contenente *c* contenga anche *q*, rispetto ad una stringa arbitrariamente scelta.

q

T

Column A	Column B	Column C
V1	V3	V5
V2	V4	V6

$$\text{SCPRank}(T) = \max \begin{cases} (SCP(q,V1)+SCP(q,V2), \\ (SCP(q,V3)+SCP(q,V4)), \\ (SCP(q,V5)+SCP(q,V6)) \end{cases}$$

SEARCH - SCPRank

Calcolo di SCP:

QUANTE CELLE IN UNA TABELLA?

$$scp(q;c) = p(q,c)^2 / p(q)p(c)$$

$$p(q) = \frac{\text{totale delle pagine web contenenti } q}{\text{totale delle pagine web}}$$

$$p(q,c) = \frac{\text{totale delle pagine web contenenti } q \text{ e } c}{\text{totale delle pagine web}}$$

QUANTE PAGINE NEL WEB?

DI QUANTI TOKEN SONO COMPOSTI *q* E *c*?

- Si calcolano gli score solo per le prime r righe di ogni tabella ☺
- Si considera solo un sottoinsieme del Web ☺

SEARCH

PERFORMANCES

Si è richiesto a due giudici (selezionati casualmente da Amazon Mechanical Turk) di valutare la rilevanza di ogni tabella rispetto ad una determinata query con un voto compreso tra 1 e 5. Una tabella è considerata rilevante se entrambi i giudici hanno espresso un voto >= 4.

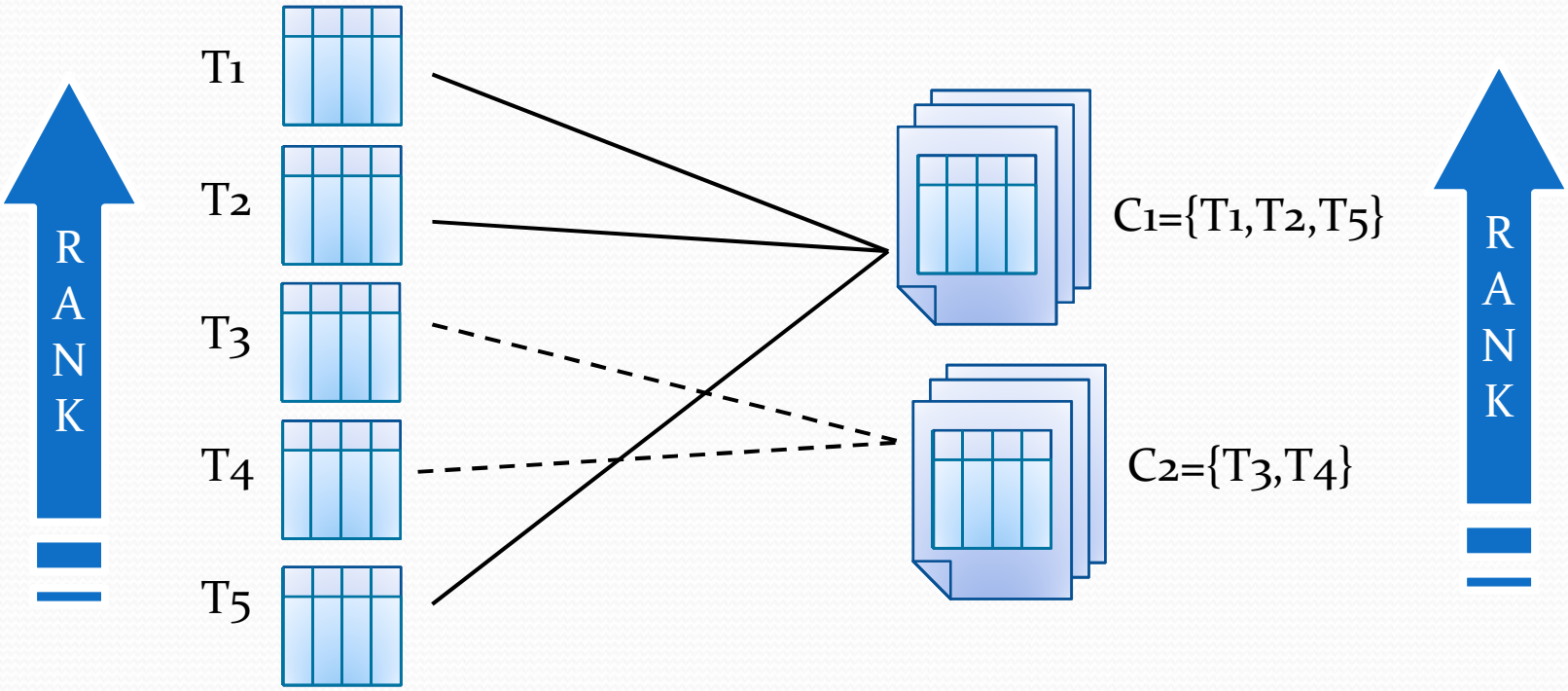
Misuriamo la qualità di ogni algoritmo di ranking calcolando la percentuale di risultati rilevanti nei primi 2, 5, 10 risultati ottenuti.

ALGORITMO	TOP 2	TOP 5	TOP 10
SimpleRank	27%	51%	73%
SCPRank	47%	64%	81%

- SCPRank ha performances nettamente superiori a SimpleRank.
- Nei primi 2 risultati troviamo tabelle rilevanti circa la metà delle volte.
- Nei primi 10 risultati troviamo tabelle rilevanti in più dell'80% dei casi.

SEARCH - Clustering

Data la lista ordinata delle tabelle, per ogni tabella T viene calcolata la distanza di similarità con le altre. Se tale distanza è maggiore di una determinata soglia le tabelle fanno parte dello stesso cluster. I cluster così ottenuti vengono poi ordinati calcolando per ognuno la media dei punteggi di rilevanza delle tabelle che lo compongono.



Come definire la funzione di distanza di similarità?

SEARCH - *TextCluster*

Calcola la distanza coseno fra i testi delle tabelle interessate.

	TABELLA1	TABELLA2
TERMINE1	w11	w12
TERMINE2	w21	w22
TERMINE3	w31	w32

Pesi tf.idf:

$w_{ij} = tf_{ij} \times idf_i$

Term Frequency (normalized):

$tf_{ij} = frequenza_{ij} / max_i \{frequenza_{ij}\}$

Inverse Document Frequency:

$idf_i = \log$ (numero tabelle / numero tabelle contenenti termine i)

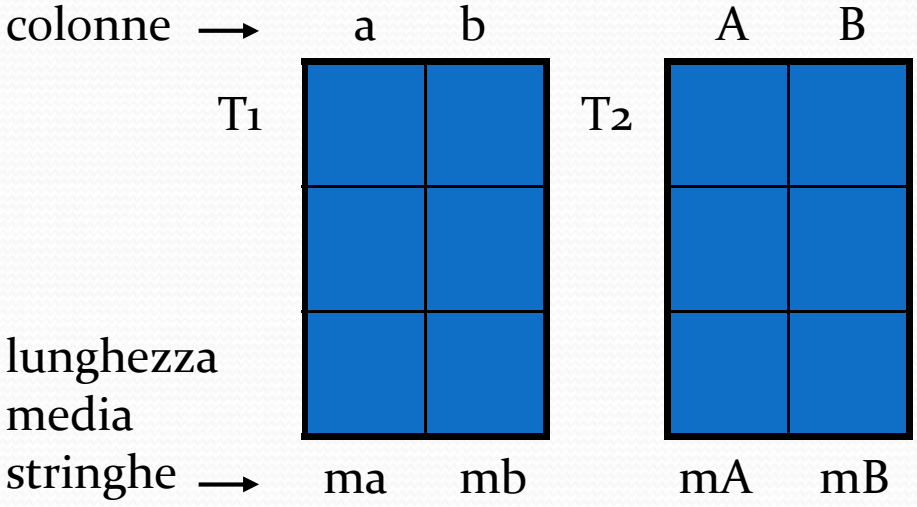
$$sim(T1,T2) = \frac{T1 \bullet T2}{\|T1\| \times \|T2\|} = \frac{\sum_{i=1..3} wi1 \times wi2}{\sqrt{(\sum_{i=1..3} wi1)} \times \sqrt{(\sum_{i=1..3} wi2)}}$$

Per tabelle anche aventi lo stesso schema, ma nessun termine in comune la similarity distance è nulla ☹

SEARCH - *SizeCluster*

Un diverso approccio: <u>stringhe dello stesso tipo avranno lunghezza simile.</u>

SizeCluster: calcola un similarity score fra colonne che valuta la differenza fra le lunghezze medie delle loro stringhe. La distanza di similarità fra tabelle è la somma dei punteggi del miglior match fra colonne.



s(aA) = score (ma-mA)

s(aB) = score (ma-mB)

s(bA) = score (mb-mA)

s(bB) = score (mb-mB)

Similarità(T1,T2) = max { (s(aA)+s(bB)), (s(aB)+s(bA)) }

ColumnTextCluster: come SizeCluster, ma calcola lo score fra colonne utilizzando la distanza coseno.

SEARCH

PERFORMANCES

Si è richiesto a due giudici (selezionati casualmente da Amazon Mechanical Turk) di valutare la similarità di due tabelle con un voto compreso tra 1 e 5. Due tabelle sono considerate simili se entrambi i giudici hanno espresso un voto >= 4.

Misuriamo la qualità degli algoritmi di clustering calcolando la percentuale di raggruppamenti di dimensione *k* in cui esiste almeno una tabella simile a quella giudicata più rilevante per una determinata query.

ALGORITMO	K= 2	K= 5	K= 10
SizeCluster	70%	88%	97%
TextCluster	67%	85%	91%
ColumnTextCluster	70%	88%	97%

- Anche quando il cluster ha soli due elementi la qualità è interessante nel 70% dei casi
- Le performances aumentano con l'aumentare delle dimensioni del cluster

SEARCH

PERFORMANCES

Valutiamo inoltre la qualità degli algoritmi di clustering calcolando il valore medio di similarità fra le tabelle appartenenti allo stesso raggruppamento al variare della dimensione del cluster *k*.

ALGORITMO	K= 2	K= 5	K= 10
SizeCluster	3.17	2.82	2.57
TextCluster	3.32	2.85	2.53
ColumnTextCluster	3.13	2.79	2.48

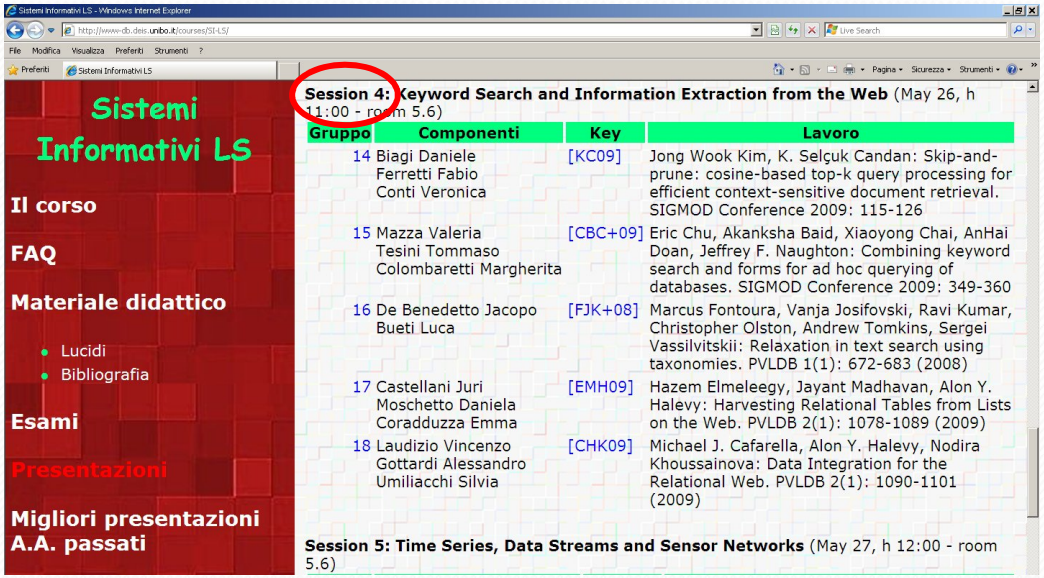
- Performances simili per i vari algoritmi
- Anche tecniche semplici come SizeCluster permettono di ottenere buoni risultati

CONTEXT

Componenti	Gruppo
Laudizio Vincenzo	18
Gottardi Alessandro	18
Umiliacchi Silvia	18
….	….

CONTEXT

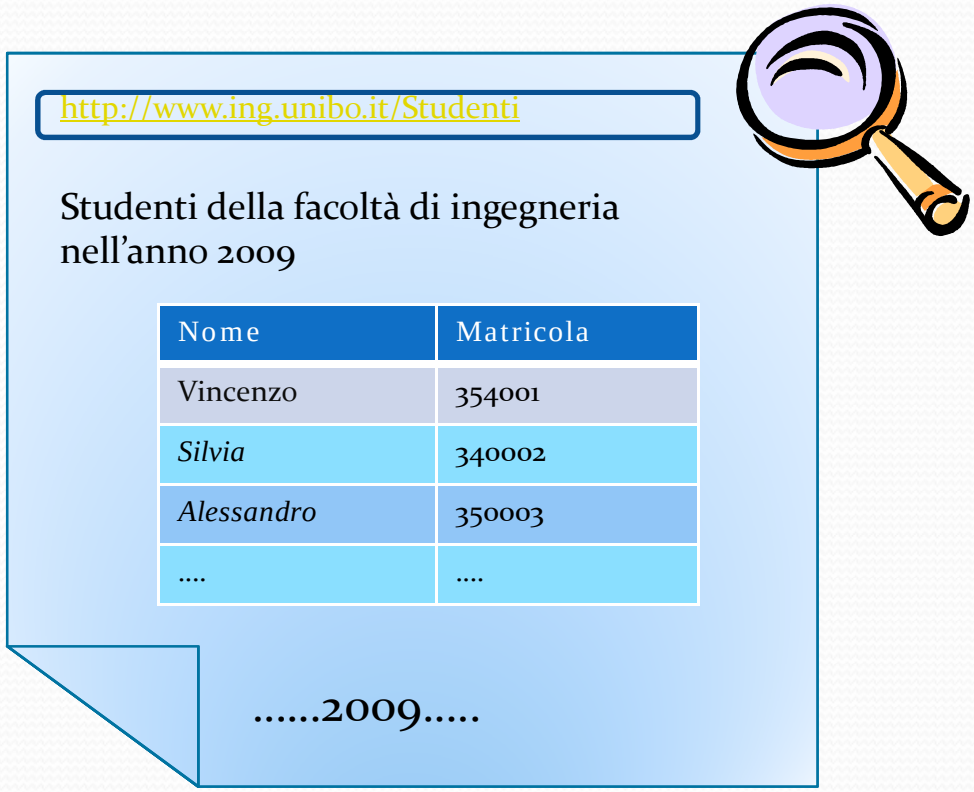
Componenti	Gruppo	Sessione
Laudizio Vincenzo	18	Sessione 4
Gottardi Alessandro	18	Sessione 4
Umiliacchi Silvia	18	Sessione 4
….	….	…



Aggiunge ad una tabella estratta da una pagina Web una o più colonne usando dati presenti all'interno della pagina stessa

CONTEXT - *SignificantTerms*

Considera la pagina web da cui è stata estratta la tabella sorgente e restituisce in output i K termini con score tf-idf più elevato presenti nella pagina ma NON nella tabella.

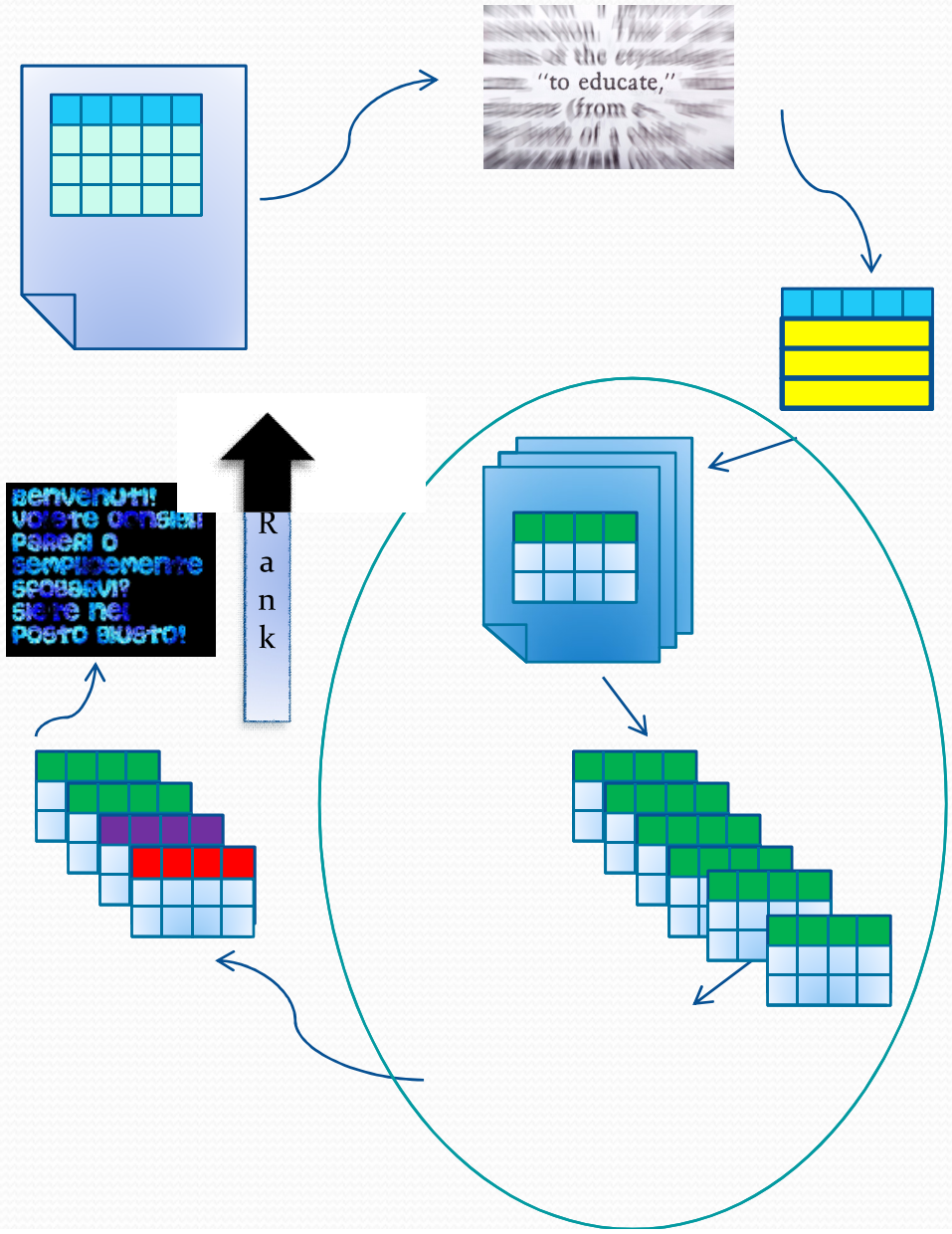


- Semplice ☺
- Si basa sull'assunzione che un termine rilevante per una pagina sia ripetuto spesso all'interno della pagina stessa ☹

CONTEXT - *RVP*

L'analisi va oltre la pagina sorgente della tabella estratta.

1. Si individuano i termini significativi, sig_terms, nella pagina della tabella in esame tramite l'algoritmo Significant Terms.
2. Per ogni riga della tabella :
   - viene effettuata una ricerca web e vengono memorizzate le prime top-k pagine, da cui vengono estratte le tabelle in esse presenti.
   - Vengono aggiunte ad una lista list_of_tables solo le tabelle che contengono almeno uno dei termini significativi presenti in sig_terms.
3. Si riuniscono tutti i termini presenti in list_of_tables e si ordinano per frequenza, ovvero per n° di tabelle in cui compaiono.
4. Vengono restituiti solo i primi k termini.



CONTEXT - *Hybrid*

SignificantTerms e RVP sono complementari:

- Significant Terms individua termini di contesto presenti all'interno della pagina che ospita la tabella.
- RVP individua termini di contesto per una tabella effettuando una ricerca sul web.

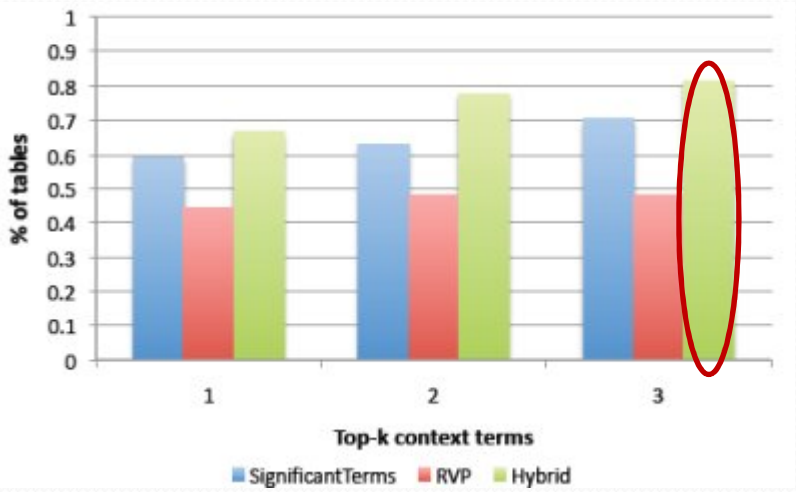
Hybrid: applica entrambi gli algoritmi e restituisce i termini di contesto recuperati alternando ordinatamente i risultati.

CONTEXT

PERFORMANCES

Si è richiesto a due degli autori di analizzare manualmente le pagine sorgenti delle tabelle più rilevanti per ogni query al fine di individuare termini di contesto ritenuti utili. Termini selezionati da entrambi gli autori fanno parte del test-set.

Per ogni algoritmo è stata calcolata la percentuale di tabelle per le quali è stato individuato un vero valore di contesto tra i primi K (k=1,2,3) generati dall'algoritmo stesso.

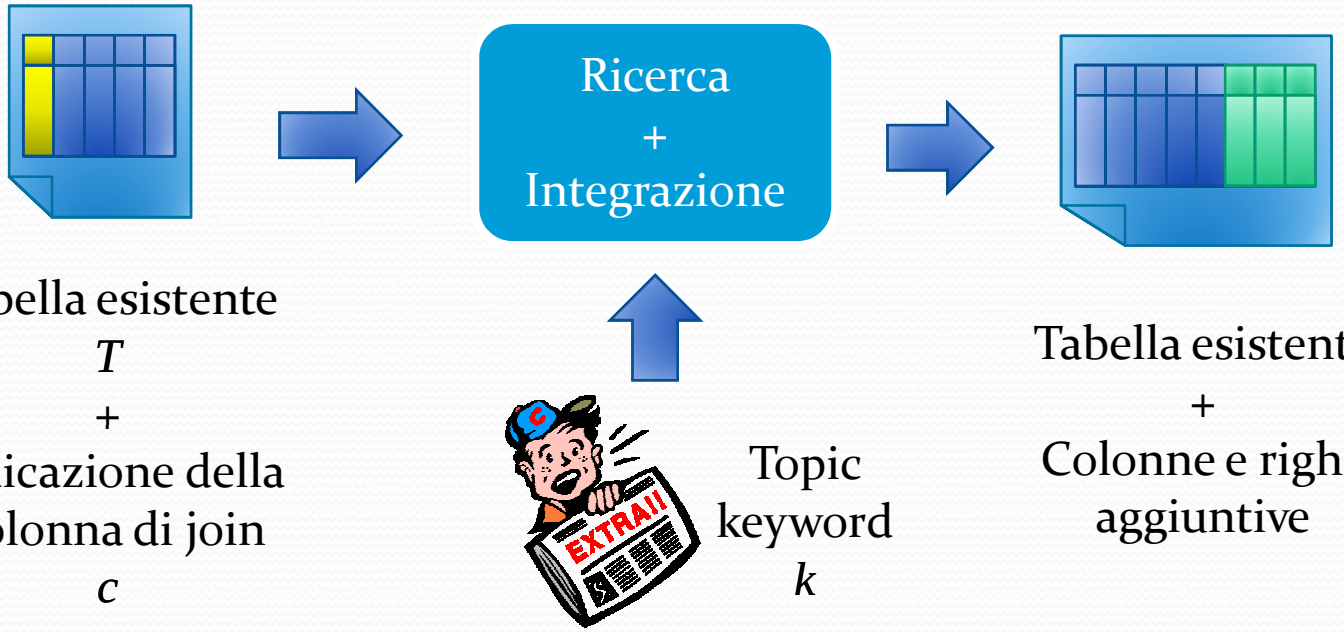


- Hybrid decora una tabella con termini utili più dell'**80%** delle volte con K= 3.
- Significant Terms è efficiente e semplice da implementare.
- RVP non supera Significant Terms , ma è comunque utilizzato per implementare l'algoritmo ibrido, che risulta essere il migliore.

RVP richiede molte ricerche web → la scelta tra Significant Terms e Ibrido dipende dalle risorse a disposizione

EXTEND

- Obiettivo: aggiungere nuove informazioni alle tabelle esistenti



Problemi:

- *schema matching*

  I dati aggiunti dall'EXTEND si riferiscono all'argomento *k* richiesto anche se sono espressi con una terminologia differente rispetto alla keyword?

- *reference reconciliation*

  I dati di cui è stato trovato un match rappresentano lo stesso oggetto del mondo reale?

EXTEND - *JoinTest*

Topic keyword "**k**"
+
Join column "**c**"

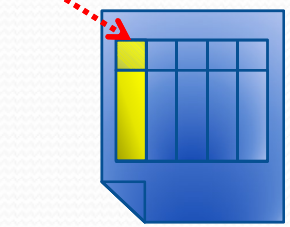


Tabella da estendere

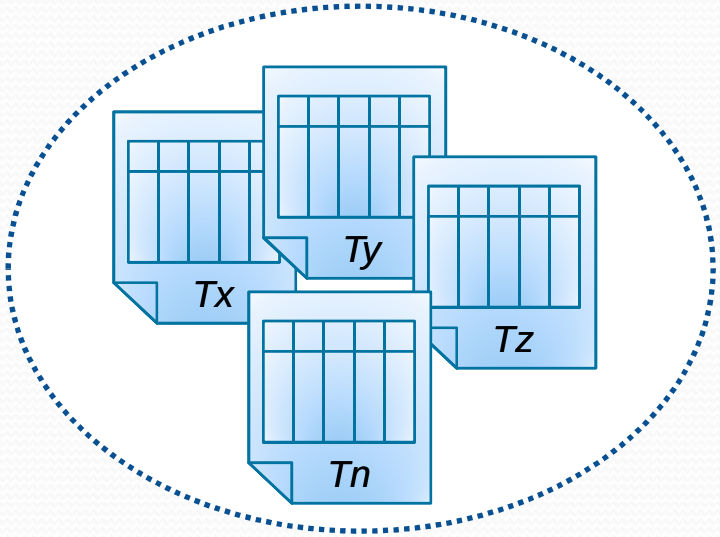
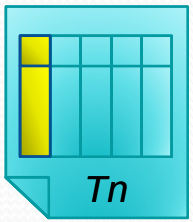


Tabella candidata



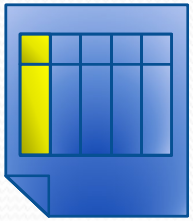
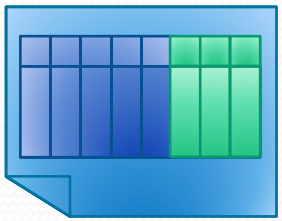


Tabella da estendere

Tabella risultante



Schema matching:
- *Motore di ricerca*

Reference reconciliation:
- *String edit-distance test*

Tabelle non correlate vengono scartate!

EXTEND - *JoinTest*

- Algoritmo semplice: ☺
  1. Query sul Motore di ricerca
  2. Seleziona la tabella candidata che ha il punteggio più elevato

Problemi:

- Dati ottenuti mediante la ricerca possono essere "sporchi" ed incompleti
- Difficilmente è possibile eseguire un join "perfetto"

Soluzione: *join-threshold test*

si utilizza la distanza di *Jaccard*

$$J' = \frac{M_{01} + M_{10}}{M_{01} + M_{10} + M_{11}}.$$

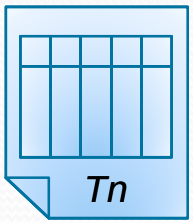
per misurare la compatibilità tra

i valori presenti nella colonna *c* della tabella *T*

ed

i valori presenti in ogni colonna di ogni tabella candidata *T'*

Se J' > threshold ➡ *T'* viene considerata joinable



Le tabelle che superano il test sono ordinate in base al rank fornito dal motore di ricerca nel momento in cui si esegue la query

EXTEND - *MultiJoin*

- Cosa potrebbe farebbe un utente per aggiungere informazioni ad una tabella?
  1. Selezionare uno ad uno il contenuto di una riga di una tabella
  2. Eseguire una serie di ricerche sottoponendo al motore di ricerca

     Il contenuto di ogni riga

     +

     lo specifico argomento (di cui si vogliono ottenere informazioni)

Lavoro oneroso:

esaminare un numero elevato di tabelle risultanti dalla ricerca

controllare se esiste una tabella che

- Abbia rilevanza nei confronti dell'argomento
- Possa essere messa in join con una specifica colonna della tabella di partenza

L'algoritmo MultiJoin intende automatizzare questo processo

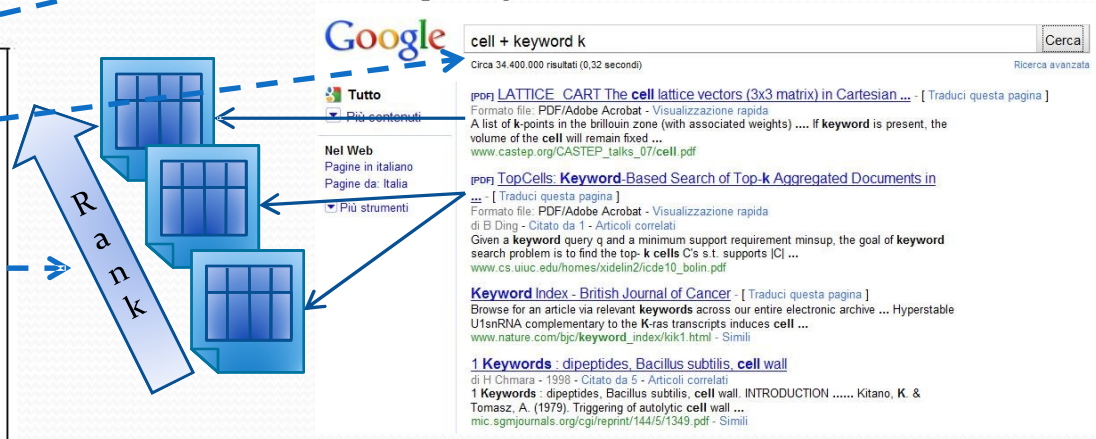
Schema matching:
• Algoritmo column-matching clustering
Reference reconciliation:
• *Motore di ricerca*

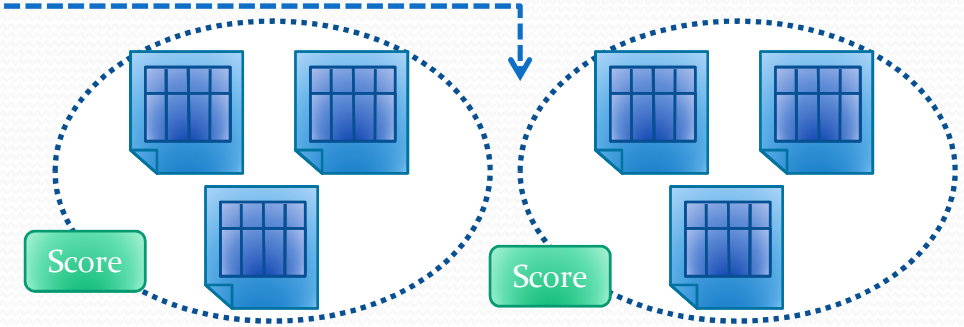
EXTEND - *MultiJoin*

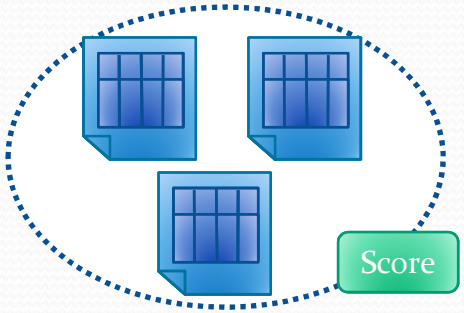
Algoritmo

```
1:  function MultiJoin(column, keyword):
2:  urls = []
3:  for cell ∈ column do
4:     urls += searchengine(cell + keyword)
5:     tables = []
6:     for url ∈ urls do
7:        for table ∈ extract_tables(url) do
8:           table.setscore(table_score(keywords, table))
9:           tables.append(table)
10:       end for
11:    end for
12: end for
13: sort tables
14: clusters = []
15: for table ∈ tables do
16:    cluster = makecluster(table)
17:    cluster.setscore(join_score(table.getScore(),column,
       cluster))
18:    clusters.append(cluster)
19: end for
20: sort clusters
21: return clusters
22:
23: function join_score(tableScore, column, cluster):
24: // Weight w is a parameter of the system
25: scoreCount = len(cluster.getUniqueJoinSrcElts())
26: score = scoreCount / len(column)
27: return (w * tableScore) + (1 − w) * score
```

Join column "c" + Topic keyword "k"







Ad ogni aggiunta di un nuovo membro:
- viene eseguito nuovamente il calcolo dello score
- Lo score è calcolato tenendo conto del rank della tabella appena inserita e dello score del cluster

Indica quante tuple distinte della tabella *T* producono i membri del cluster (Tale valore viene normalizzato)

Risultato: viene scelto il cluster con lo score di copertura più elevato

EXTEND

PERFORMANCES

NOTA: non tutte le tabelle sono estendibili!

-> i dati non esistono o non sono recuperabili tramite il WEB

- Si applica l'operatore SEARCH ad un sottoinsieme delle query di test
- Per ogni query, si utilizza la tabella più rilevante estratta dal Web come sorgente per l'operatore EXTEND
- Si sceglie una colonna *c* di join ed un topic *k*

Si calcola la percentuale delle righe della tabella *T* alle quali vengono aggiunte correttamente nuove informazioni rilevanti

	JoinTest	MultiJoin
Tuple per cui sono stati recuperati dati aggiuntivi	43%	100%
Tuple estese (in media)	60%	33%
Valori aggiuntivi	Dipende dalla tabella	45,5 (in media)

EXTEND

PERFORMANCES

JoinTest

cerca una tabella singola che copra tutte le celle della colonna di join

Trova un risultato in 3 casi su 7

In questi casi solo per il 60% delle tuple è stato possibile effettuare un join

MultiJoin

cerca una estensione per ogni singola tupla della colonna di join

Trova informazioni addizionali per ogni singola tupla!

In media il 33% di queste può essere esteso:

con informazioni presenti nelle tuple appartenenti al **Cluster** con maggiore copertura

tupla estesa → (in media) 45,5 valori di estensione

Quale algoritmo?

Dipende dalla natura dei dati! ☺

Sviluppi futuri

- Possibili miglioramenti degli algoritmi in termini di prestazioni e/o qualità:
  - EXTEND: JoinTest e MultiJoin potrebbero essere combinati, ovvero si potrebbe applicare per primo JoinTest e, qualora esso dovesse fallire, si procede con MultiJoin
  - CONTEXT: RVP richiede molte ricerche web → si potrebbero esplorare strutture di indicizzazione alternative per ridurne il numero

    Come indicizzare?

- Aggiungere semantica agli operatori

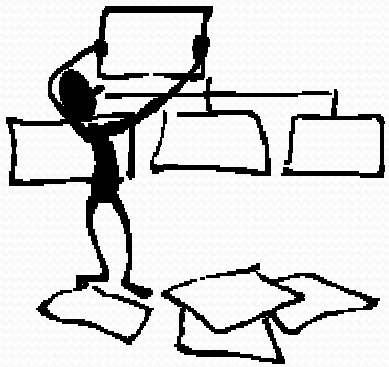
    es. EXTEND: individuazione della scuderia corrente di un pilota di F1

Conclusioni

In breve:

- Octopus è un sistema che aiuta l'utente nell'integrazione di dati provenienti da molteplici sorgenti strutturate presenti nel Web.
- È stato presentato il set degli operatori di base che consentono al sistema di svolgere lavoro utile per l'utente
- Sono stati descritti gli algoritmi che implementano tali operatori in maniera efficace con relative performances.
- Dagli esperimenti si può concludere che :
  - con tutti e tre gli operatori si possono ottenere risultati di buona qualità
  - il sistema migliora la produttività dell'utente
- Alternative?

  Eseguire il lavoro manualmente…

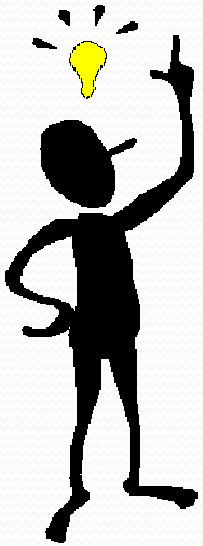


Grazie per l'attenzione

Gruppo 18

Ale

Vins



Sil