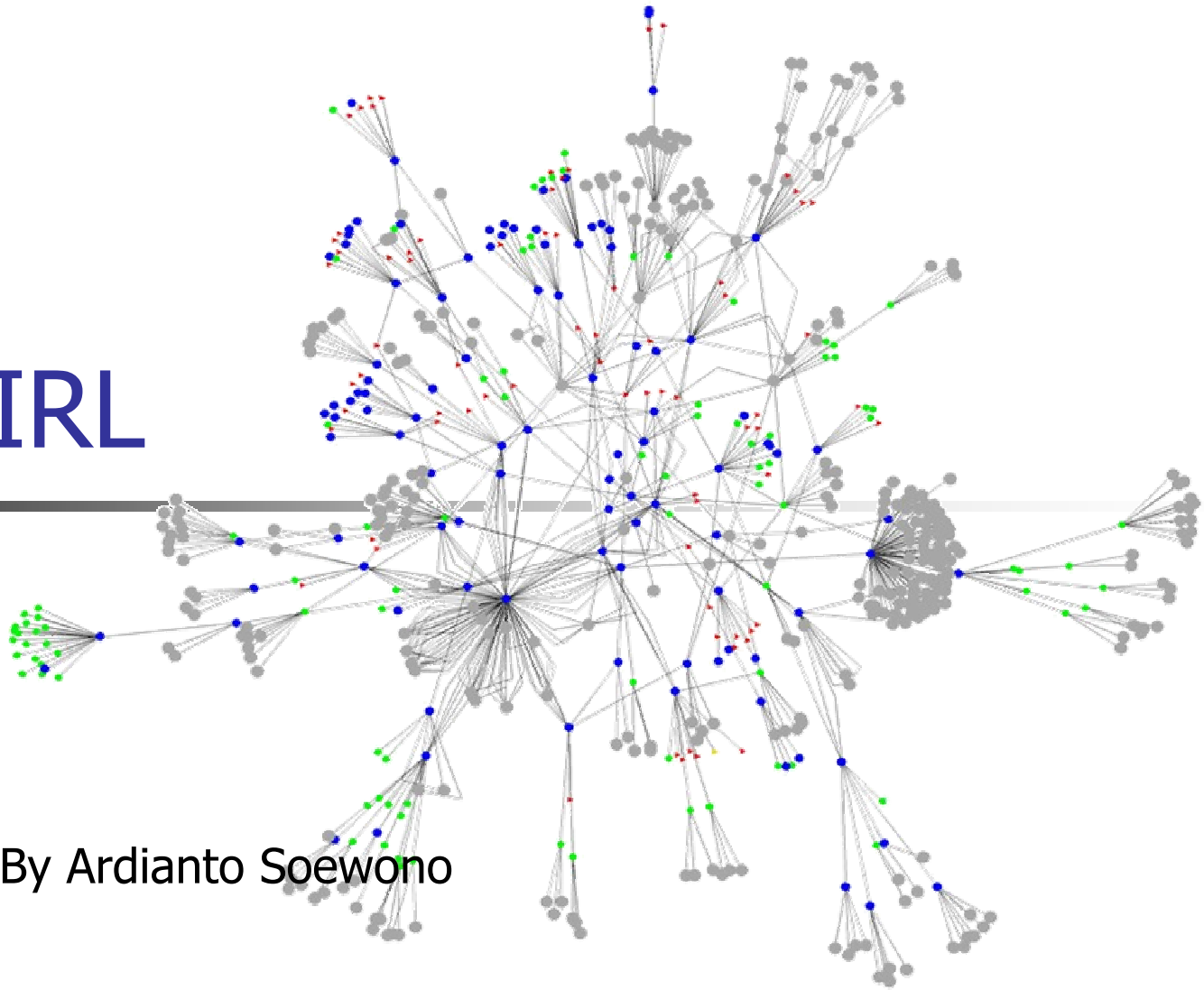
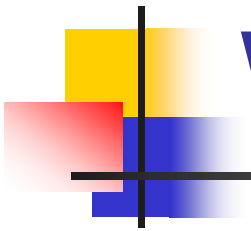
WHIRL

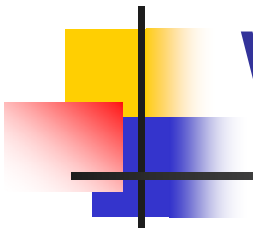


Presented By Ardianto Soewono



What's WHIRL?

- WHIRL stands for Word-based Heterogeneous Information Retrieval Logic.
- WHIRL is a probability database logics used to integrate database that doesn't have common domains.
- As reflected on the name, WHIRL uses words-matching procedure to join tables.



Why WHIRL?

- There is no common domain in the heterogeneous databases.

  Exp : “Microsoft” and “Microsoft Corp”, “AT&T” and “AT&T Lab”

- Mapping from local domain to global domain is hard.

  The mapping from name constants to real entity can be different from databases to databases

- High accuracy is needed in order to avoid erroneous or missing answers to user queries.

Document Vector

- WHIRL uses document vector as data model
  - There is set of vocabulary terms T.
  - A fragment of text will be represented by a document vector, each component corresponds to a term $t \in T$
  - Each component of document vector v which corresponds to a term t will be denoted as v^t
  - The weight scheme is as follow :

    1. If v^t is not present in the document represented by v, weight = 0
    2. Else :

$$\hat{V}^t = (\log(TF_{V_t}) + 1).\log(IDF_t)$$

Where TF_{Vt} = the number of items term t occurs in document v

IDF_t = |C| / $|C_t|$

where C_t = subset of documents C that contains term t

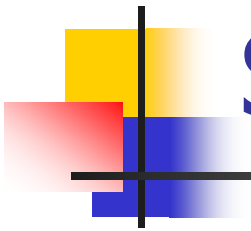
Document Vector (cont.)

- The similarity of 2 document vectors v and w can be computed as follows :

$$sim\ (v, w) = \sum_{t \in T} v^t . w^t$$

- The idea of this scheme is giving more weight to infrequent terms.

  Exp : Terms “Inc”, “Ltd” will have less weight than “AT&T” and “Microsoft.



Scoring

- An EDB consists of a term vocabulary T and set of relations $\{p_1, p_2, \ldots, p_n\}$.
- Each relation p with arity k has set of tuples, tuples(p).
- Every tuple $<v_1, \ldots, v_k>$ has k components and each components v_i is document vector over T
- Every tuple has a score, that is, score($<v_1, \ldots, v_k> \in$tuples(p))
- In a base relation, this score is initialized to 1.

Conjunctive Queries

- Conjunctive query in WHIRL is written as B_1^B_2^...^B_k where each B_i is a literal.
- There are 2 types of literals
  - EDB literal is written p(X_1, ... , X_k) where p is the name of relation and X is variable symbol.

    The score of EDB literal will be :

    $score(B\theta) = score(< X_1\theta,..., X_k\theta > \in p)$ if $< x_1\theta,..., x_k\theta > \in p$

    $score(B\theta) = 0$ otherwise.
  - Similarity literal is written X ~ Y

    The score of similarity literal will be :

    $score\ (B\theta) = sim\ (X\theta, Y\theta)$
- We define the score for conjunction query as follow :

$$score\ (Q\theta) = \prod_{i=1}^{n} score\ (B_i\theta)$$

Union of Conjunctive Queries

- A basic WHIRL clause is written $p(X_1, \ldots, X_k) \leftarrow Q$
- Union of queries in WHIRL is called view
- Support of view is defined as a set of triples $(A \leftarrow Q, \theta, s)$ satisfying conditions :
  - $(A \leftarrow Q) \in V$
  - $A\theta = a$ and $Q\theta$ is ground
  - $score(Q\theta) = s$ and $s > 0$
- Score of $\langle x_1, \ldots, x_k \rangle$ is :

$$score(< X_1,...,X_K >\in p) = 1 - \prod_{<C,\theta,s>\in support(p(X_1,...,X_k))} (1-s)$$

A* Search

```
procedure A*(r, s_0, goalState(·), children(·))
begin
    OPEN := {s_0}
    while (OPEN ≠ ∅) do
        s := argmax_{s' ∈ OPEN} h(s')
        OPEN := OPEN - {s}
        if goalState(s) then
            output (s, h(s))
            exit if r answers printed
        else
            OPEN := OPEN ∪ children(s)
        endif
    endwhile
end
```

Functions & States

- WHIRL uses 2 functions :
  - Inverted index : map terms $t \in T$ to the tuples that contain them
  - Index(t,p,i) : return tuples $<v_1,\ldots,v_i,\ldots,v_k>$ in tuples(p) in which $v_i^t > 0$
- States :
  - Pairs of <θ,E> where θ=substitution and E=set of exclusion
    - E is a pair of <t, Y> where t = term, Y=variable
    - E is E-valid if it satisfies condition :

$$\forall < t, Y > \in E, (Y\theta)^t = 0$$

  - Initial States: <0,0>
  - Goal States: <θ,E> in which θ is ground.

Generating States

- Exploding States :
  - For a state s = <θ, E>, pick EDB literal $p(y_1, \ldots, y_k)$ in which Y is unbound by θ.
  - Construct all states of the form $<\theta \cup \{y_1=v_1, \ldots, y_k=v_k\}, E>$ such that
    - $<v_1, \ldots, v_k> \in$ tuples(p)
    - $\theta \cup \{y_1=v_1, \ldots, y_k=v_k\}$ is E-valid
- Constraining a States :
  - For a state s = <θ, E>, pick constraining literal X~Y and some term t (probably a term with highest weight) in document Xθ such that $(t, Y) \in E$
  - Will produce 2 sets :
    - Singleton set containing state s' = <θ, E'> where $E'=E \cup \{(t, Y)\}$
    - Set s_t contains all states $<\theta_i, E>$ in which $\theta_i = \theta \cup \{y_1=v_1, \ldots, y_k=v_k\}$ for $<v_1, \ldots, v_k> \in$ index(t,p,l) and θ_i is E-valid

Heuristic Function

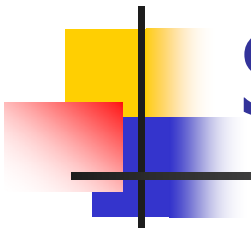
- A* search algorithm uses heuristic function to determine which state has the highest score
- A heuristic function h is said to be admissible if for all states s and all states s' reachable from s, h(s)≥ h(s')
- Heuristic function is defined as follows :

$$h(\theta, E) = \prod_{i=1}^{k} h'(B_i, \theta, E)$$

Where :

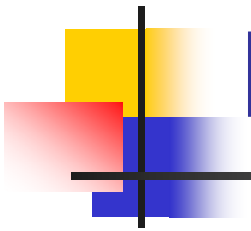
$$h'(B_i, \theta, E) = score(B_i\theta)$$ if $B_i\theta$ is ground, or

$$h'((X \sim Y), \theta, E) = \sum_{t \in T: <t, Y> \notin E} x^t .maxweight(t, p, l)$$ otherwise



Steps to Generate States

1. Choose the smallest relation in the query
2. Do the explode state for the relation
3. Add each of the state to the OPEN list
4. Find the highest score state by using heuristic function
5. If the highest score state is ground, output it
6. Do the constraining state to the highest score state with constraining literal defined in the query
7. Repeat to step 4 until we have r-tuples in output



Example

- Query : const(IO) ^ p(Company, Industry) ^ Industry~IO

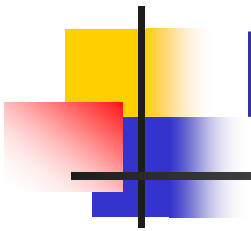
  IO = "telecommunications services and/or equipment"

  - Start with the smallest relation, in this case IO
  - Explode IO, the result is single document IO, named it as state s_1 and placed it in OPEN list
  - Because s_1 is the only state in the OPEN list, removed it and do the constraining nodes to s_1
  - Choose term t which has the highest weight, probably "telecommunications". The inverted index is used to find all tuples containing the words "telecommunications"

    All of these substitutions are ground, so the value of h(.) is the actual score of each tuple.
  - Put all of these tuples as states in OPEN list, together with new state s_1' containing the exclusion <telecommunications, Industry>

Example (cont.)

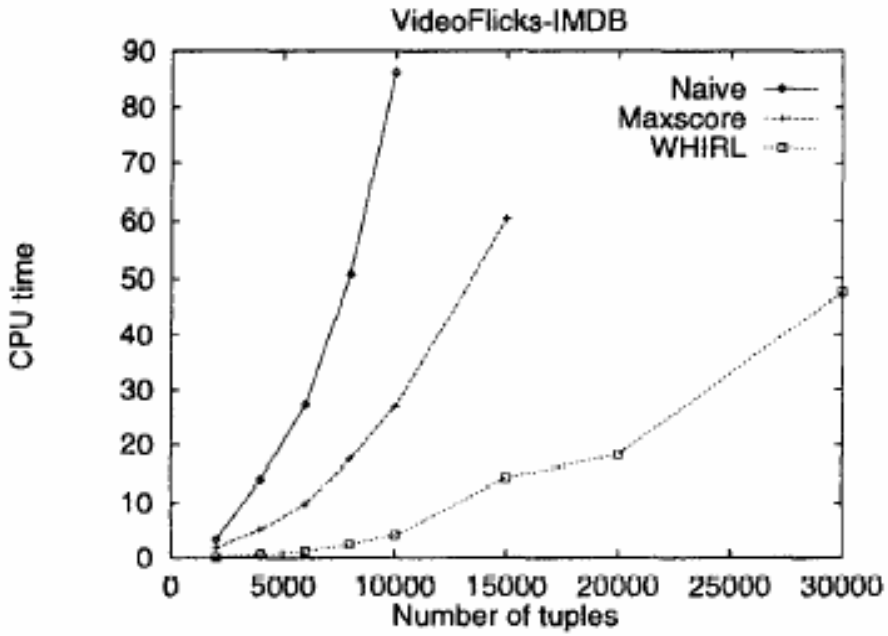
- Choose a state to be removed from the OPEN list
- Since we choose a term that has maximum weight, it might be that $h(s_1')$ is less than h(.) from other states
- If it is, ground substitution will be removed from OPEN list, and an answer will be output
- If $h(s_1')$ is higher than the best goal statem we do again the constraining nodes to s1' which means choosing a term from "services", "and", "or", "equipment" which has the highest weight
- Do like what have we done before, until we found r-answer as the output

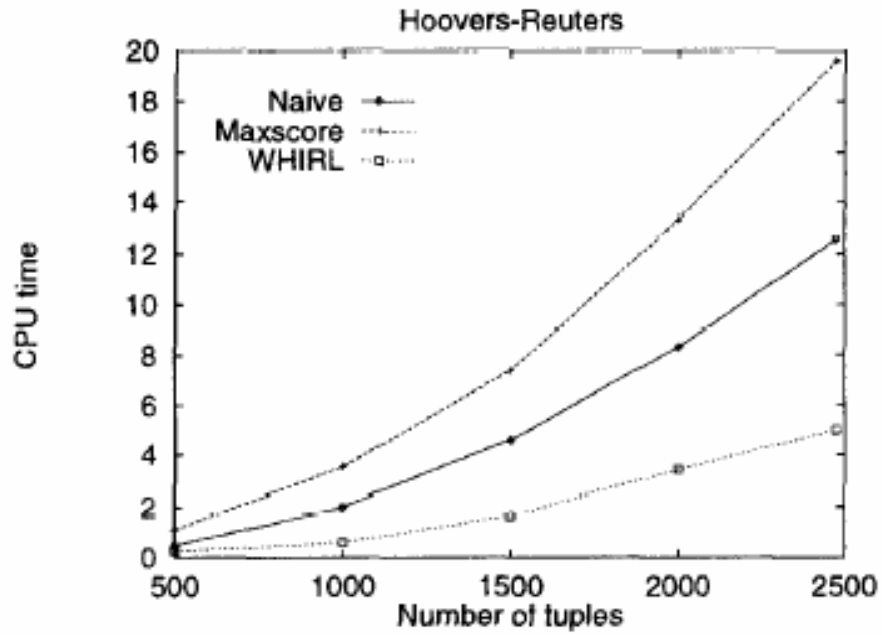


Experimentation

- The experimentation compares WHIRL with 2 other methods
- Naïve method for similarity join
  - Take i-th column of relation p, submit it as IR ranked retrieval query to a corpus corresponding to the j-column of relation q
  - Top r result is merged to find the best r-answer
- Maxscore optimization
  - The same for naïve method, except that maxscore optimization is used to find best r result.

Time Result


VideoFlicks-IMDB
CPU time
Number of tuples
Naive
Maxscore
WHIRL
0
10
20
30
40
50
60
70
80
90
0
5000
10000
15000
20000
25000
30000


Hoovers-Reuters
CPU time
Number of tuples
Naive
Maxscore
WHIRL
0
2
4
6
8
10
12
14
16
18
20
500
1000
1500
2000
2500

Average Precision I

- The average precision is computed as follow :

$$averageprecision = \sum_{k=1}^{r} c(k).\frac{a_k}{k}$$

Where : a_k=number of correct answer in first k

c(k) = 1 if the k-th answer is correct and 0 otherwise

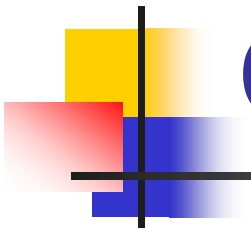
Similarity Joins

Domain	Average Precision
Business	84.6%
Animals	92.1%
Movies	100.0%

Average Precision II

Similarity Joins with Incompatible Schemata

Pairing	Average Precision
movieName/movieName	100.0%
movieListing/movieName	100.0%
movieName/review	98.0%
movieListing/review	94.0%



Conclusion