Optimized Union of Non-disjoint Distributed data sets

Itay dar, Tova milo, Elad Verbin
Tel Aviv University

OUTLINE

- Motivation
- (Optimal) Union Plan
- Compact Information Gathering
- Experimental Evaluation
- Related Work & future work

MOTIVATION

- P2P setting
- Download of data items from several sources
  - In MANCOOSI – download packages (info on packages) residing on several sources (peers)
- Sources often overlap and contain common items
- We want to avoid transmission of redundant information

MOTIVATION (cont.)

- Abstractly can be viewed as a union query
- Define the notion of optimal union plan (that minimize redundant data transmission)
- Devise efficient algorithm to compute and execute such plans
- Optimally exploit the network capabilities
- A key challenge is the lack of global map of items distribution

FORMAL PROBLEM DEFINITION

- A Peer To Peer Environment
  - Each peer p_i is holding a set of data items items(p_i)
  - All data items have the same size
  - A Simple network model
    - communication is discreet - Working in rounds
    - Communication is considered Reliable
    - Each peer has a static upload and download rate
      - Download(p_i)
      - Upload(p_i)
    - There are no other networks constraints

Union Plan

- Union plan - a set of tuples of the form (from,to,item,time) s.t
  - No bandwidth constraints are breached
  - All items in items(P) are sent to the target eventually
- Optimal plan - the maximal time point is minimal
- Direct Plan
- Non redundant plan
- Theorem: there always exist a direct non-redundant plan that is optimal

Union Plan (2)

- Proof sketch
  - Each plan can be transformed into a non redundant plan
  - We can remove all the item sent on path which don't reach the target
  - We can look at the set of items each peer is sending out from his on local items, all the sets are disjoint and cover all the union set.
  - The plan for this sets of items is optimal, we show in the next part we can build an optimal direct plan given a disjoint set of items so both plans are equivalent and we are done.

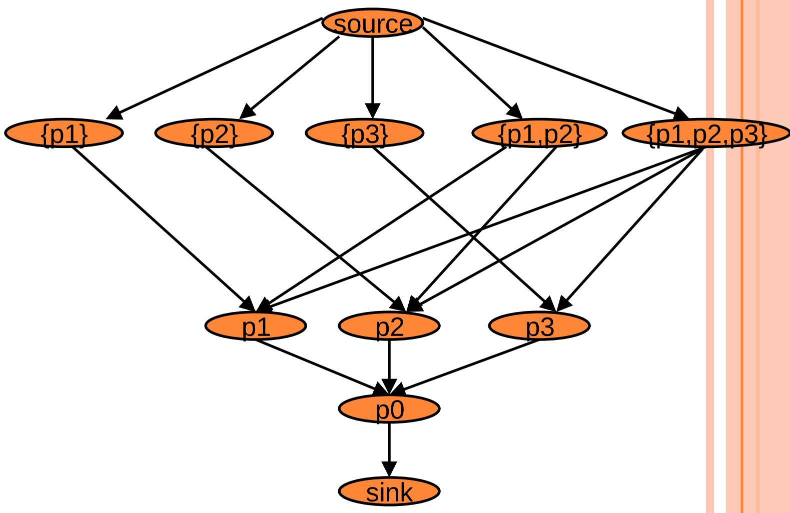
Optimal Union Plan

- Global Knowledge Solution
  - Oracle knows the items each peer holds
- Assign data algorithm
  - Decide which item will be sent by which peer
- Send data algorithm
  - Create the concrete plan which tells when each peer should send his data items

Assign data algorithm

- Decide which peer send which data items
- Using CheckTime(t) algorithm
  - Which Assigns data to peers given the number of rounds, notify upon failure.
  - Equivalence class is the set of items that resides **only** in a given set of peers.
  - Using max flow Computation the items from each equivalence class will be split among the equivalence class members

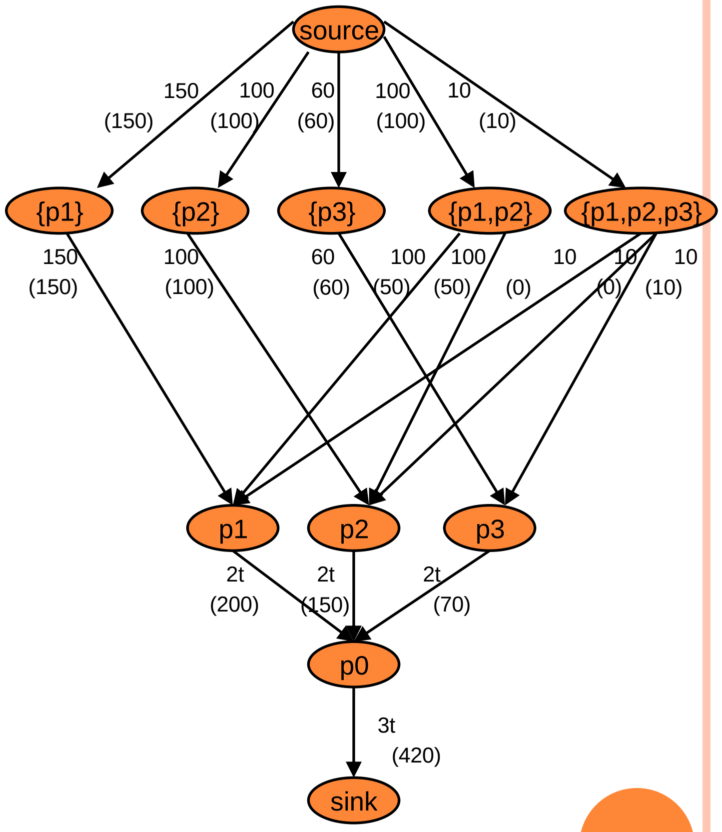
CheckTime



- Graph vertex structure
  - Source vertex
  - equivalence class layer
  - Peers layer
  - Target vertex
  - Sink vertex
- Graph edges structure
  - From Source to each equivalence class vertex (equivalence class size as the edge weight)
  - Each equivalence class vertex to all his peer members vertex (equivalence class size as the edge weight)
  - Each peer has an edge to the target vertex (weight equals the amount of data units he can send in t rounds)
  - The target vertex is connected to the sink (weight equals the amount of data the target can possibly receive in t rounds)

Example

- 3 peers p_1,p_2,p_3 each can upload 2 item at each round
- The target p_0 can receive 3 items at each round
- p_1 and p_2 share 100 items
- All three peers share 10 items
- p_1 also holds 150 items
- p_2 also holds 100 items
- p_3 also holds 60 items



Assign data (3)

- Searching for minimum time using check time (search boundaries using send data)
- Complexity
  - Polynomial in the size of the graph which is exponential in the number of peers
- Correctness proof sketch
  - Plan -> flow (trivial)
  - Flow -> plan (needs send data part)

SEND DATA ALGORITHM

- Decide the peers data sending order
- Naïve solution
- Why naïve is not good enough ?
  - 3 peers each can send 2 items each round and get 3 items each round
  - P0 have 300 items
  - P1 and p2 have 50 items
  - Naïve ends after 175 rounds
  - Non naïve ends after 150 rounds
- Time to finish
  Bottleneck metric

```
SendData(input: P, p0;   output: U)
t := 0; U := ∅;
for each p ∈ P
   ♯items(p) := |items(p)|;
   time_to_finish(p) := ♯items(p)/upload(p);
end for
while there exists some peer p with ♯items(p) > 0
   t := t + 1;
   ♯send(p) := 0 for every p ∈ P;
   free := download(p0);
   while free > 0
      choose a peer p, among those with ♯send(p) < upload(p),
        where time_to_finish(p) is maximal.
      ♯send(p) := ♯send(p) + 1;
      ♯items(p) := ♯items(p) − 1;
      time_to_finish(p) := ♯items(p)/upload(p);
      ♯free := free − 1;
   end while
   for each p ∈ P, with ♯send(p) > 0, add to U instructions
   to send, at time t, ♯send(p) new items from p to p0;
end while
return U;
```

Figure 3.2: The SendData Algorithm

Send data algorithm (2)

- correctness proof sketch
  - Time to finish invariant
    - If(time_to_finish(pi) > time_to_finish(pj) at any round then time_to_finish(pi) > time_to_finish(pj) – 1
- Assign data correctness
  - Allocating bandwidth according to send data
    - Flow Constraints are not breached due to algorithm nature
    - we shall look at the first non saturated round
      - one of the peer sending data there has been sending data from the start, and will do so till the end (bottleneck)
      - the edge from the peer to the target vertex enforce the plan time.
- Send Data Complexity
  - O(m*n+nlogn)

Send data algorithm

- Optimized version
  - Bandwidth allocation is fixed during consecutive rounds
  - We also need to change the plan format
  - Groups of peer with different time to finish gets the same amount of bandwidth to the group until 2 groups get merged.
  - The bandwidth allocation inside a group during a time interval doesn't matter – so we make it regular (compress plan size)
- Complexity
  - $O(P^2)$

Compact Information Gathering

- Deriving the Plan
- Executing the Plan
- The c-Cluster Algorithm

Deriving the Plan

- Assign data needs
  - The peers upload and download speeds
  - All equivalence class sizes
- Send data needs
  - The peers upload and download speeds
  - Each peer data items AD allocation size

EQUIVALENCE CLASS SIZES ESTIMATION

- Bottom k sketches
  - Computing jaccard distance $$\frac{|s_1 \cap \ldots \cap s_j|}{|s_1 \cup \ldots \cup s_j|}$$
  - Estimating set size using interpolation

$$v_i = \frac{|s_i|}{|s_1 \cup \ldots \cup s_j|}$$

  - s_i is known, v_i is computed, so we can compute
- By using the Inclusion exclusion formula we can compute $\left|\bigcap_{p\in\hat{P}} items(p) - \bigcup_{p\in P-\hat{P}} items(p)\right|$

  as it equals

$$\begin{aligned} |\textstyle\bigcap_{p\in\hat{P}}| & - && \textstyle\Sigma_{p\in P-\hat{P}} \, |\, items(p)\,| \\ & + && \textstyle\Sigma_{p,p'\in P-\hat{P}} \, |\, items(p) \cap items(p')\,| \\ & - && \textstyle\Sigma_{p,p',p"\in P-\hat{P}} \, |\, items(p) \cap items(p') \cap items(p")\,| \\ & + && \ldots \\ & \ldots && \\ & -(+) && |\textstyle\bigcap_{p\in P-\hat{P}} items(p)\,| \end{aligned}$$

Equivalence class sizes estimation

- Si is choose such it’s the biggest group.
- Drawbacks
  - Exponential number of computation in the Inclusion exclusion formula
  - Error builds up during computation.
  - Computing the distance for a high number of groups is inaccurate.

Executing the Plan

- Each peer needs to know which items he needs to send according to the plan
- To Do So we need to identify each item set membership.
- Using Compressed Wrapped bloom filters
  - Bloom filter
  - Compressed Bloom filters
  - Compressed Wrapped bloom filters

The c-Cluster Algorithm

- Scalability problem
  - Exponential number of sets
  - Estimation breaks down with too many sets involved
- c-Cluster Algorithm
  - Estimating Replication level

```
c-Cluster(input: P, p_0, c > 1, r;   output: U)
while the number of redundant items in P is above
the threshold r
   divide P into pairwise disjoint clusters (subsets of peers)
      of size c
   call AssignData for each cluster;
   for each p ∈ P,
      remove from items(p) all the elements that where
      not assigned to p by the AssignData;
end while
call SendData to obtain a union plan U for the peers;
return U;
```

Figure 4.1: The c-Cluster Algorithm

Experimental Evaluation

- Syntactic results
  - Model settings
    - 3 * 1024 *1024 data items
    - 750k down 75k up adsl cable line
    - Number of peers varied from 2 till 65
- Parameters to tune
  - Cluster size
  - Bloom filter size
  - Replication threshold

Experimental Evaluation

- Comparison Metrics
  - PR
    - Naïve algorithm
      - Send data from the peers in a round robin manner
    - PR = (Plan time + plan creation time)/ naïve time
    - PR-data = (Plan time) / naïve time
  - Error rate
  - Performance vs. optimal where possible

SYNTACTIC RESULTS

- C cluster size
  - 2 was chosen due to high bloom filter overhead with larger c sizes
- Bloom filter type and size
  - 25 peers experiment

bloom filter	error rate %	PR
4	3.155	0.15
4 w&c	2.639	0.15
8	0.190	0.20
8 w&c	0.145	0.20
16	0.005	0.28
16 w&c	0.003	0.28

Figure 5.3: Bloom filters of varying sizes

REPLICATION LEVEL THRESHOLD

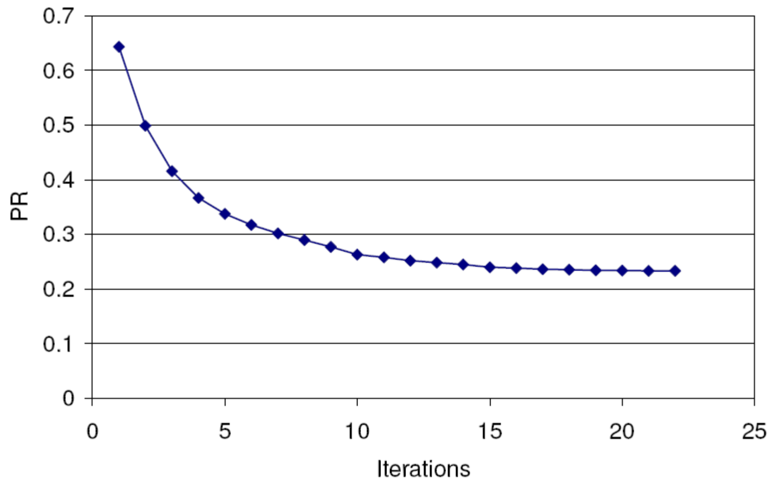
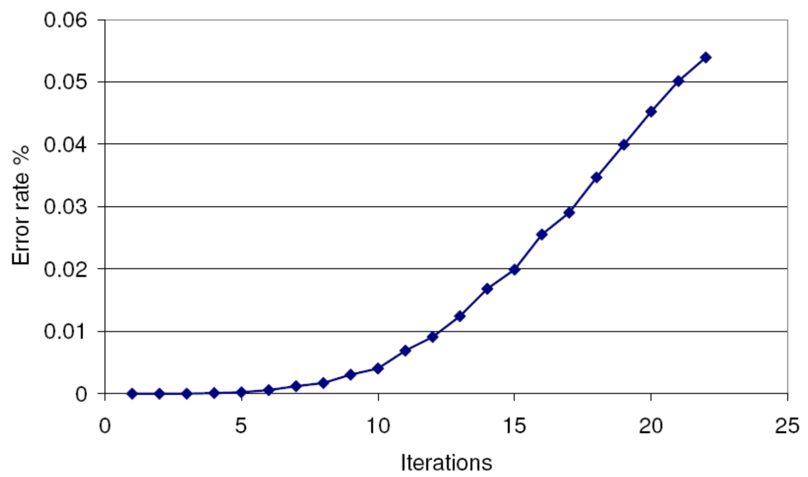


Figure 5.1: PR after each iteration



- 25 peers experiment

SYNTACTIC RESULTS

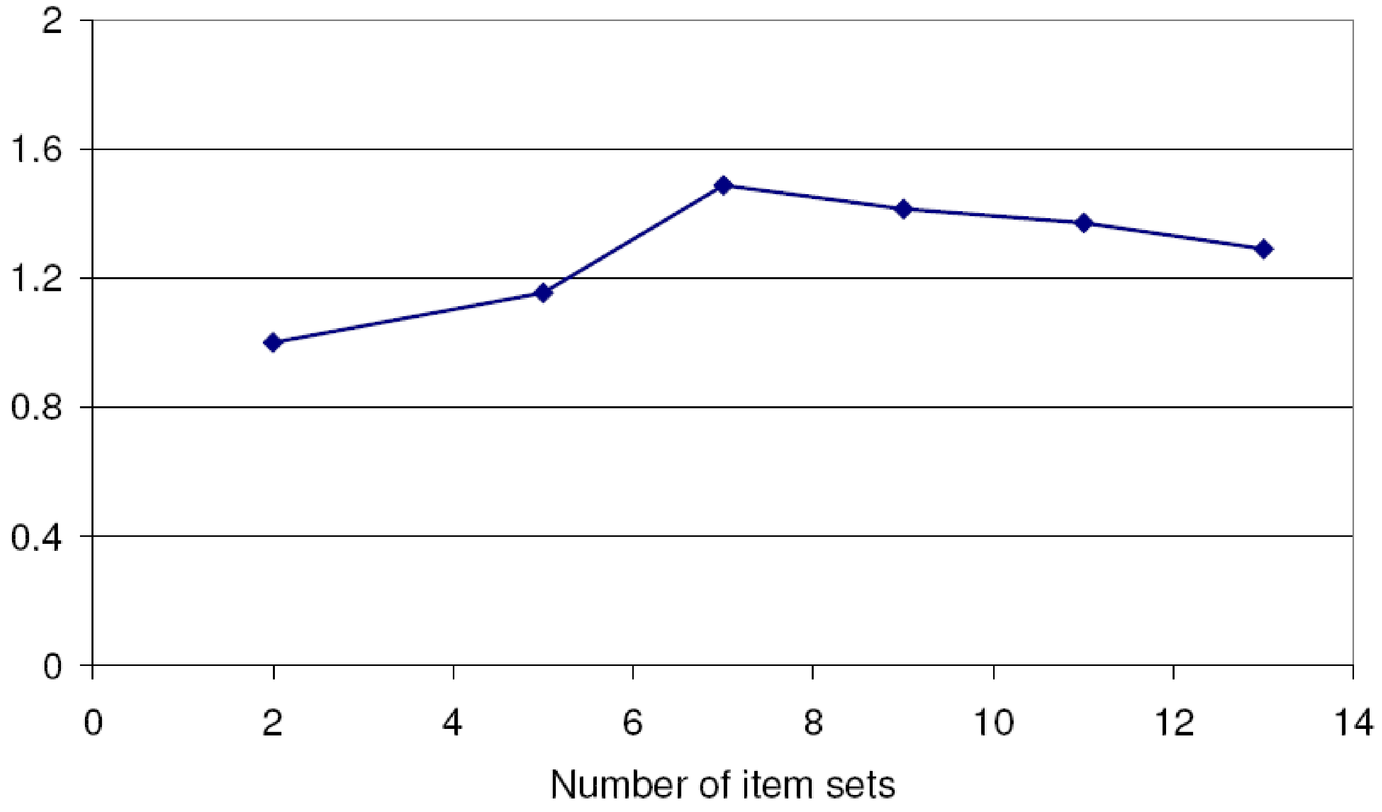


Figure 5.7: Time / optimal plan time

SYNTACTIC RESULTS

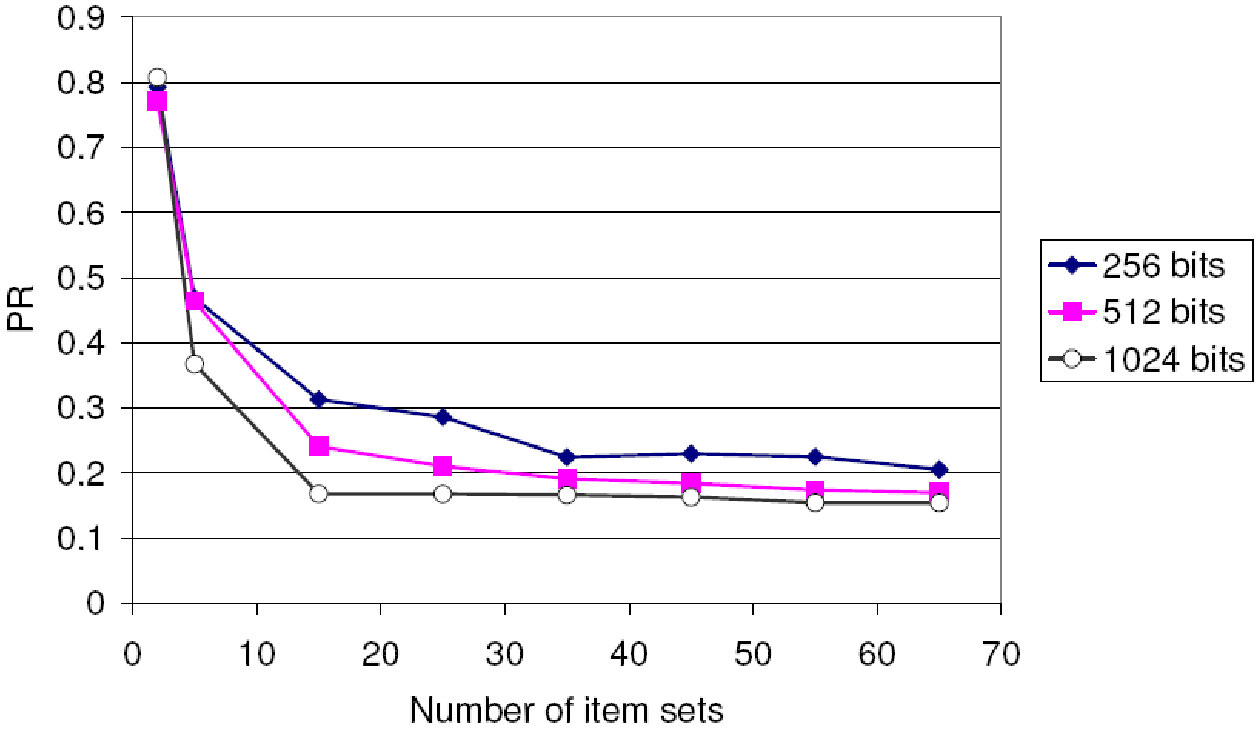
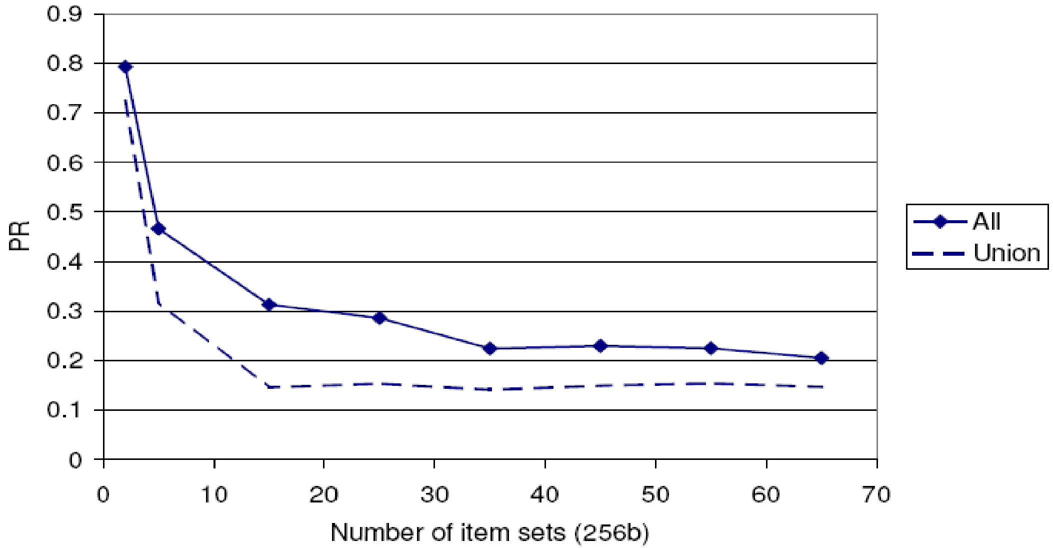
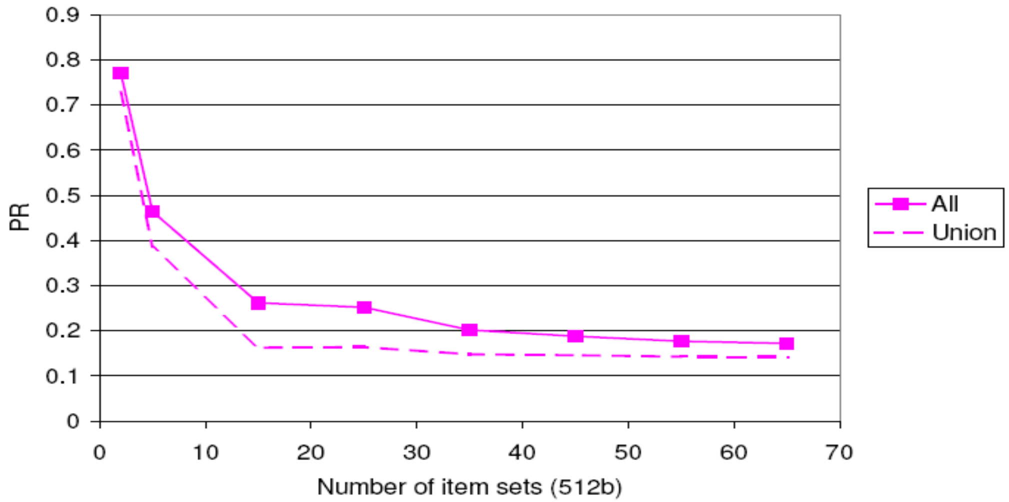


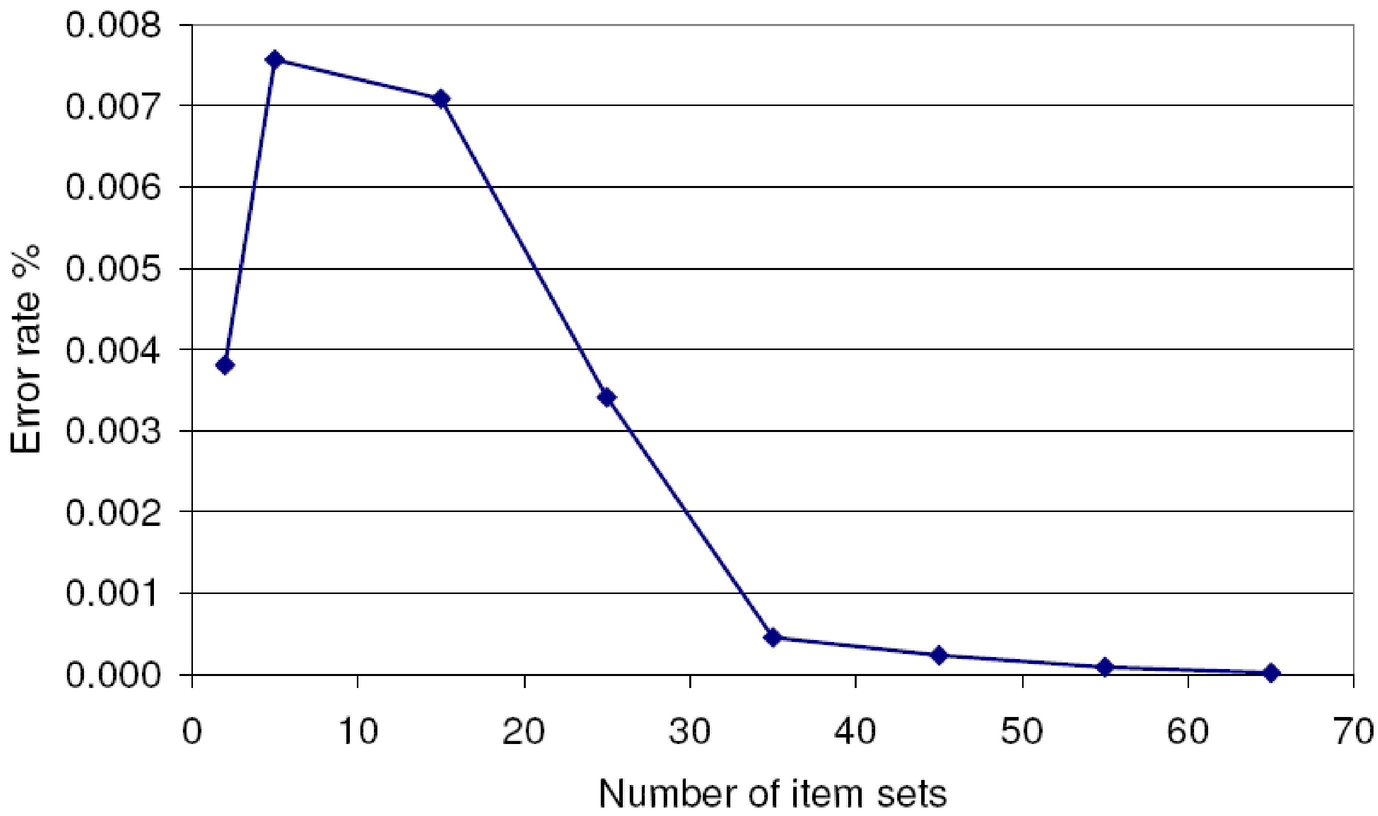
Figure 5.4: Performance Ratio for growing number of sets

Syntactic results





Syntactic results



WIKIPEDIA RESULTS

- Using Wikipedia
  - OR queries over synonyms
  - Same parameters as in the syntactic version

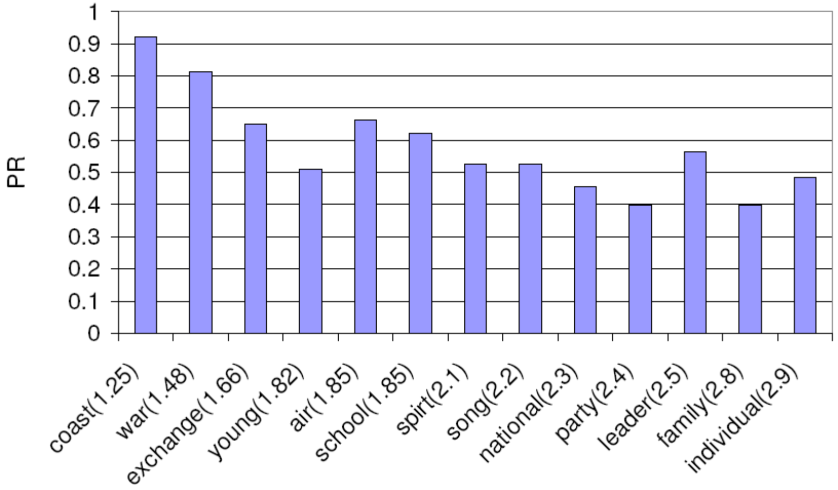


Figure 5.8: PR for varying replication level

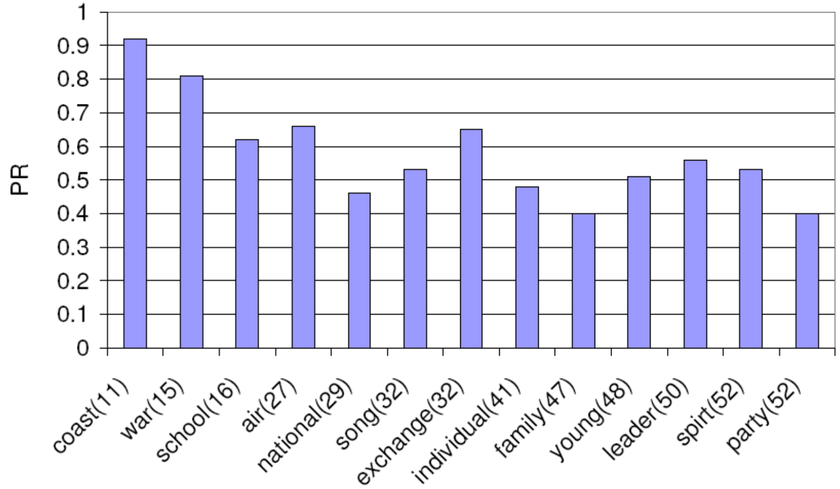
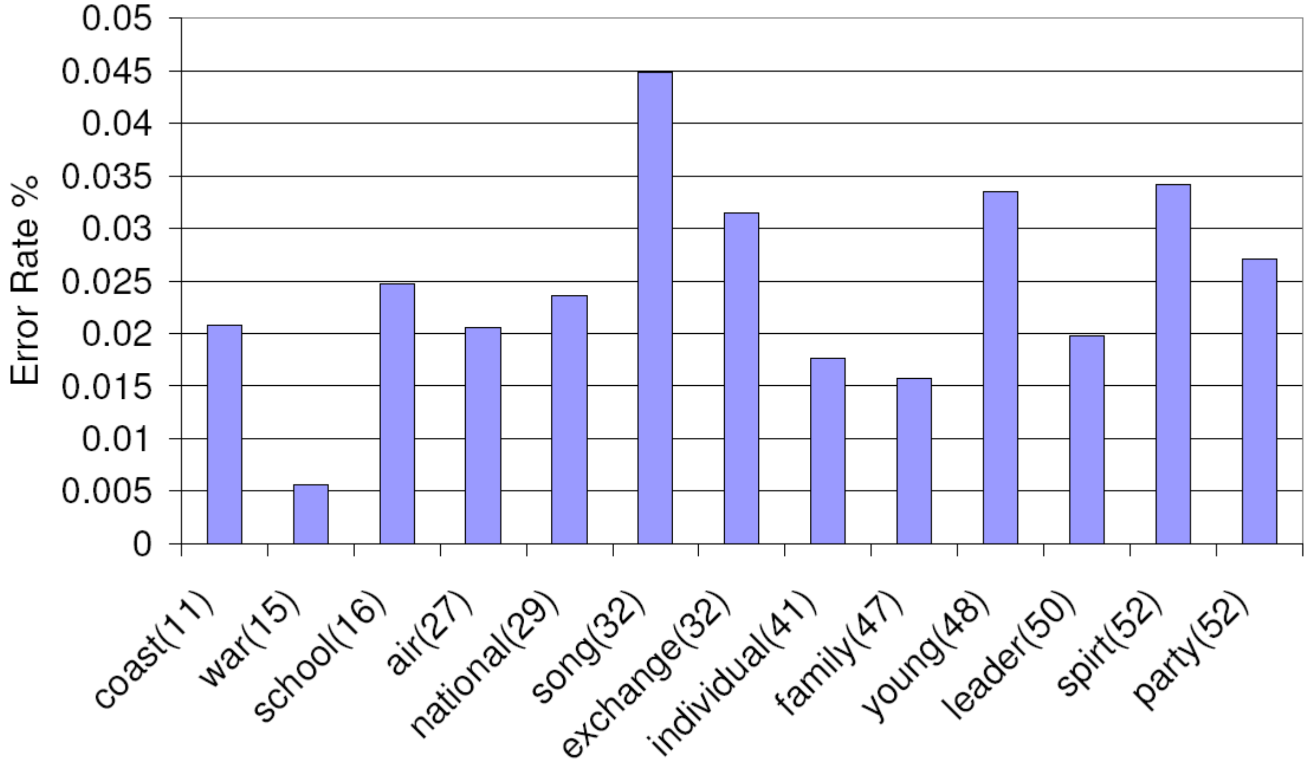


Figure 5.9: PR for varying number of synonyms (unioned sets)

WIKIPEDIA RESULTS



Related Work

- Problem Hardness (Yao)
  - Computing set difference requires passing the entire data set
- Practical set reconciliation (Minsky et al)
  - pairwise sets-reconciliation computing a characteristic polynomial
  - Estimating / guessing the set difference size
  - Passing n points, and factorizing and interpolating to find the missing points.
  - Not so practicable in our context (four seconds to compute a 200 object difference)
- Informed content delivery across adaptive overlay networks (byras et al)
  - Creating a tree of bloom filters
  - Solving again the pairwise case mostly
  - Employ erasure codes methods to solve data loss issue.
  - But they have a high error rate.

Future Work

- Pretty vast
  - Real application usage (emule dht?)
  - Dynamic setting
  - Fault tolerance
  - Scalability issues