"An Efficient HC Method for Large Datasets with Map-Reduce"

Relevance:
→ mentions applications to astronomical datasets.

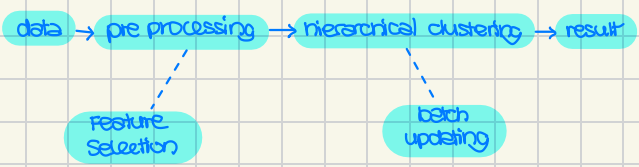
Introduction:
→ large datasets pose challenges for data-mining algorithms to efficiently process data within given constraints (e.g. memory, execution time).

→ to overcome constraints, data mining algos can be implemented with Map-Reduce:
- Map Reduce breaks large datasets into small chunks and processes them in parallel on multiple cluster nodes and scales easily to mine hundreds of TBs of data.

→ low efficiency of HCA stems from two aspects:
i) HCA of large datasets consists of many successive iterations of clustering processes in which feature matrix merging & updating & similarity value modification are common operations.
- invokes file operations of distributed file system and constant input-output (IO) operations.

ii) large dimension of feature vectors demands high memory usages.

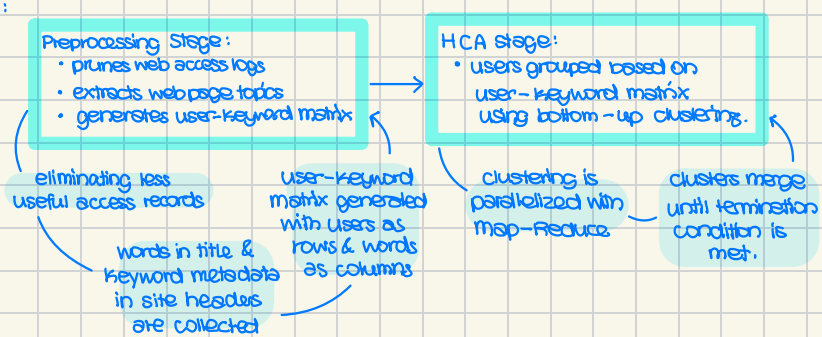
→ two proposed optimization techniques:
i) co-currence based feature selection at the pre-processing stage
ii) batch updating to reduce IO overhead, batching as many IO and communications operations as possible in one iteration.



Overview
→ mining application consists of two major phases:
i) preprocessing of raw data
ii) HC of user groups
} specific data used in this study are web access logs.

heavy IO overhead

old Approach:



Occurrence Based Feature Selection

→ one method to improve efficiency is to reduce the dimension of the user-keyword matrix.
- this can be done with CBFS.

→ motivations for CBFS:
- keywords in title vs. metadata are not equally important in weight; some are even noisy.
- summary of keywords in webpage can be given by sample keywords.

→ CBFS reduces dim. of feature vectors by summarizing a user's interested topics with the most representative keywords.
- for any two keywords, their co-currence frequency is calculated to reflect the relationship in semantics between key words.
- higher element values/ attention degrees (d_{mj}) of feature vectors is given to more related key words.

HCA with Batch Updating

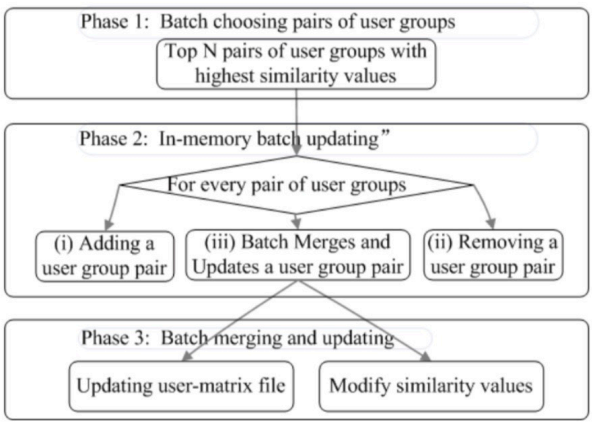


Figure 2. Hierarchical Clustering with Batch Updating

Batch updating is proposed to improve HCA efficiency:
- combines several iterations of clustering into one so that the updating matrix file and modifying similarity values are processed in batch mode.

Uses two new data structures for efficient clustering:

i) C-queue: a queue that stores the top N pairs of user groups with the highest similarity values.
C i j } denotes the pair contains user groups ui & uj

ii) Batch-queue: a queue that stores Cij to be batch processed in one iteration, in order.

Table II
NUMBERS OF ITERATIONS OF BATCH UPDATING AND EXECUTION TIMES WITH DIFFERENT SIZE N OF C-QUEUE

N	# of iter.	Execution time(seconds)
500	1678	46962
100	1680	28603
50	1680	21412
10	1684	26615

Table III
NUMBERS OF UPDATES AND EXECUTION TIMES OF HIERARCHICAL CLUSTERING WITH AND WITHOUT BATCH UPDATING

	# of iter.	Execution time(seconds)
With BU	1680	21412
Without BU	27939	287258

Results:
→ CBFR reduced dim. of feature vectors by 47%.
→ Batch updating decreases iteration of clustering to nearly 1/15.

Takeaways:
→ consider as an alternative to PCA?
- PCA tries to reduce dimensionality by exploring linear dependency.
- PCA is unsupervised.

- feature selection ranks input variables in terms of how useful they are to predict the target value.