

AD-HOC AGGREGATIONS OF RANKED LISTS IN THE PRESENCE OF HIERARCHIES

SIGMOD 2008, VANCOUVER

Nilesh Bansal, Nick Koudas

University of Toronto

Sudipto Guha

University of Pennsylvania

Ranked Lists are Ubiquitous

- Google Trends
 - ▣ List of popular search queries posed at Google
- Flickr
 - ▣ List of popular tags sorted by their popularity
- Alexa
 - ▣ List of top websites based on traffic ranks
- BlogScope
 - ▣ List of top keywords in the blogosphere

Ranked Lists can be Temporal

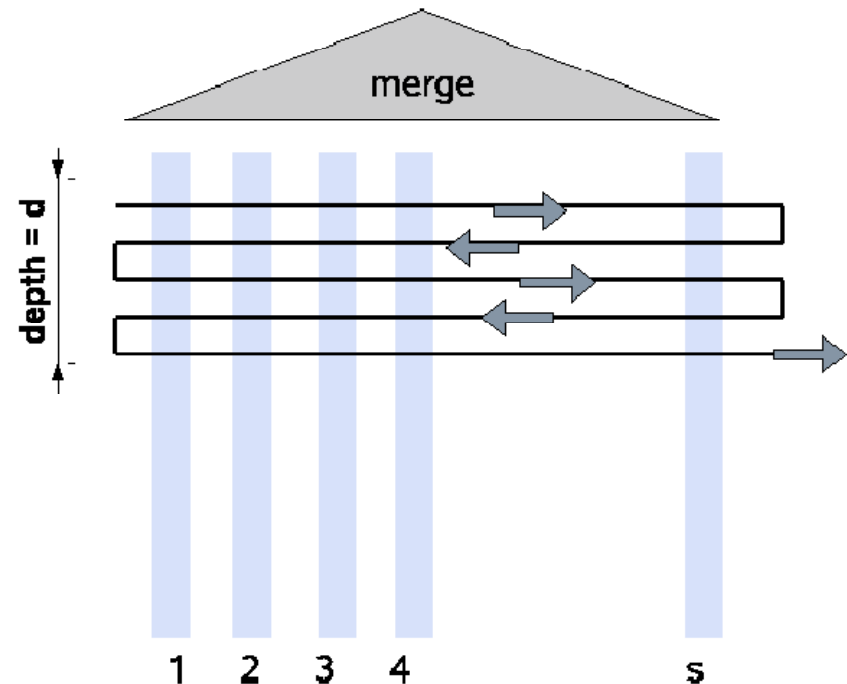
3

- Google Trends
 - ▣ List of top queries for each day with counts
- Flickr
 - ▣ List of top tags for each hour with popularity scores
- Alexa
 - ▣ List of top websites for each month with visitor counts
- BlogScope
 - ▣ List of top keywords for each day with popularity scores

Top-K is Solved

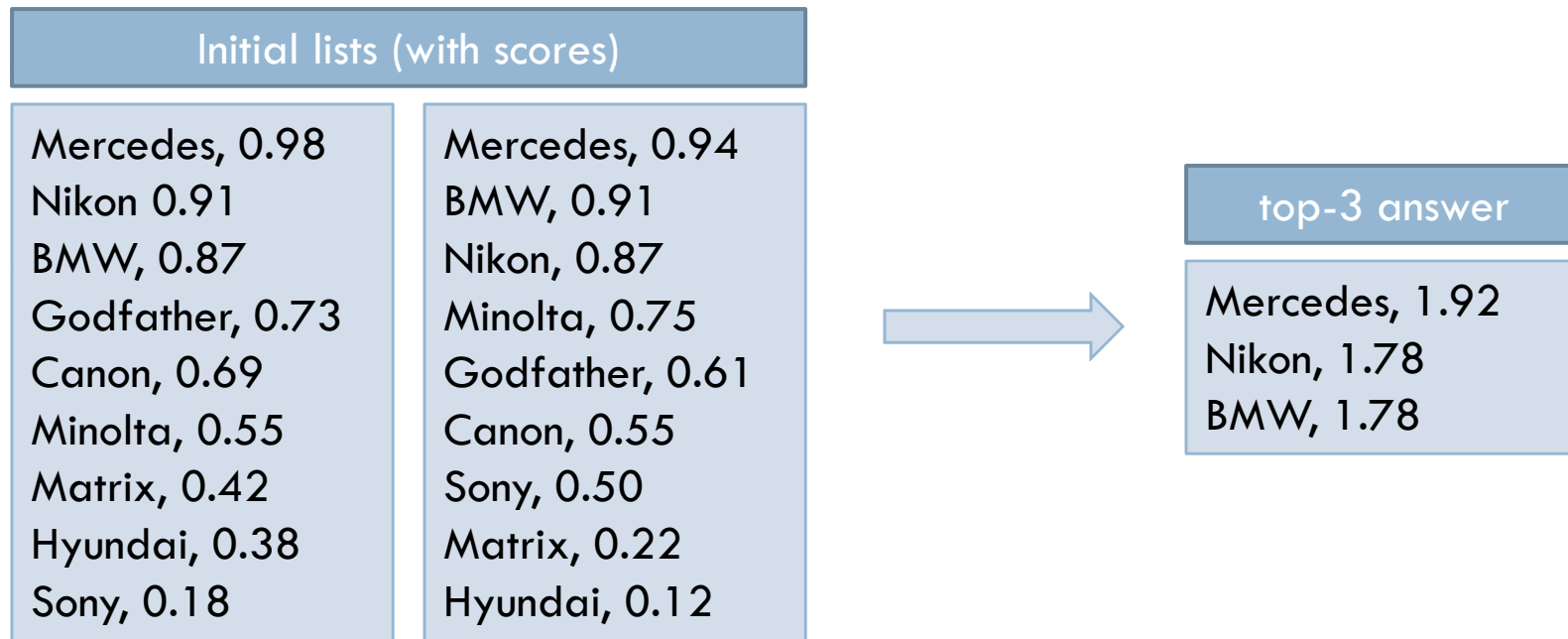
4

- Find top-k elements from a collection of lists with highest aggregate score
- Threshold Algorithm
 - Well Studied



Top-K is Simple

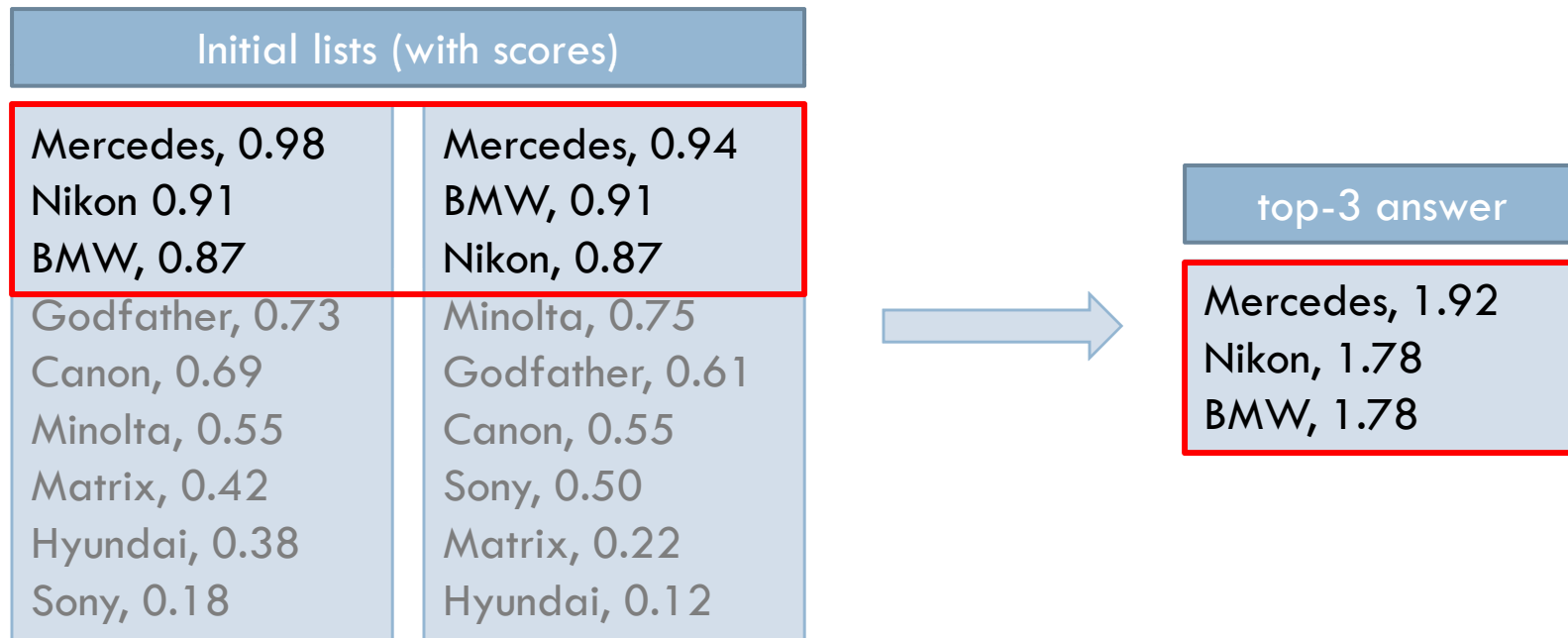
Given two lists, one for each day, of top keywords for that day in the blogosphere, **find top-3 most talked about keywords.**



Top-K is Simple

6

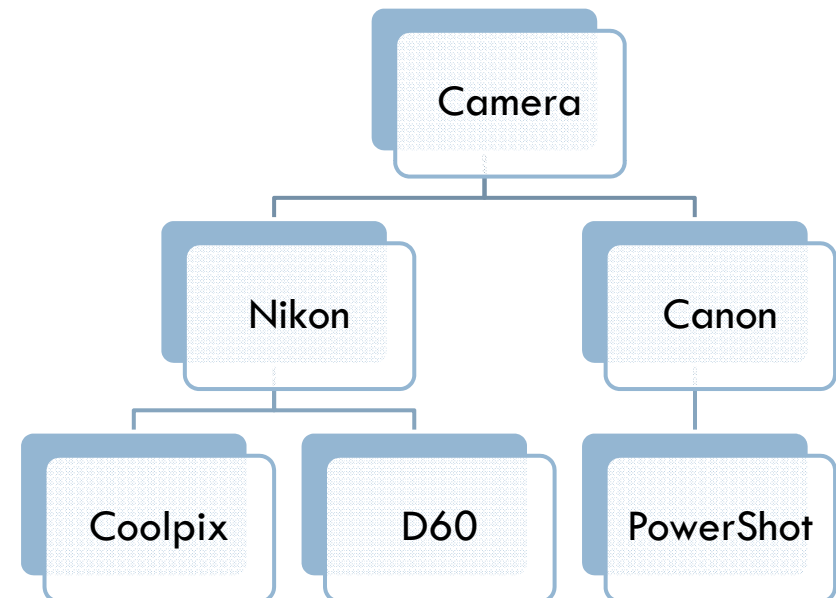
Given two lists, one for each day, of top keywords for that day in the blogosphere, **find top-3 most talked about keywords.**



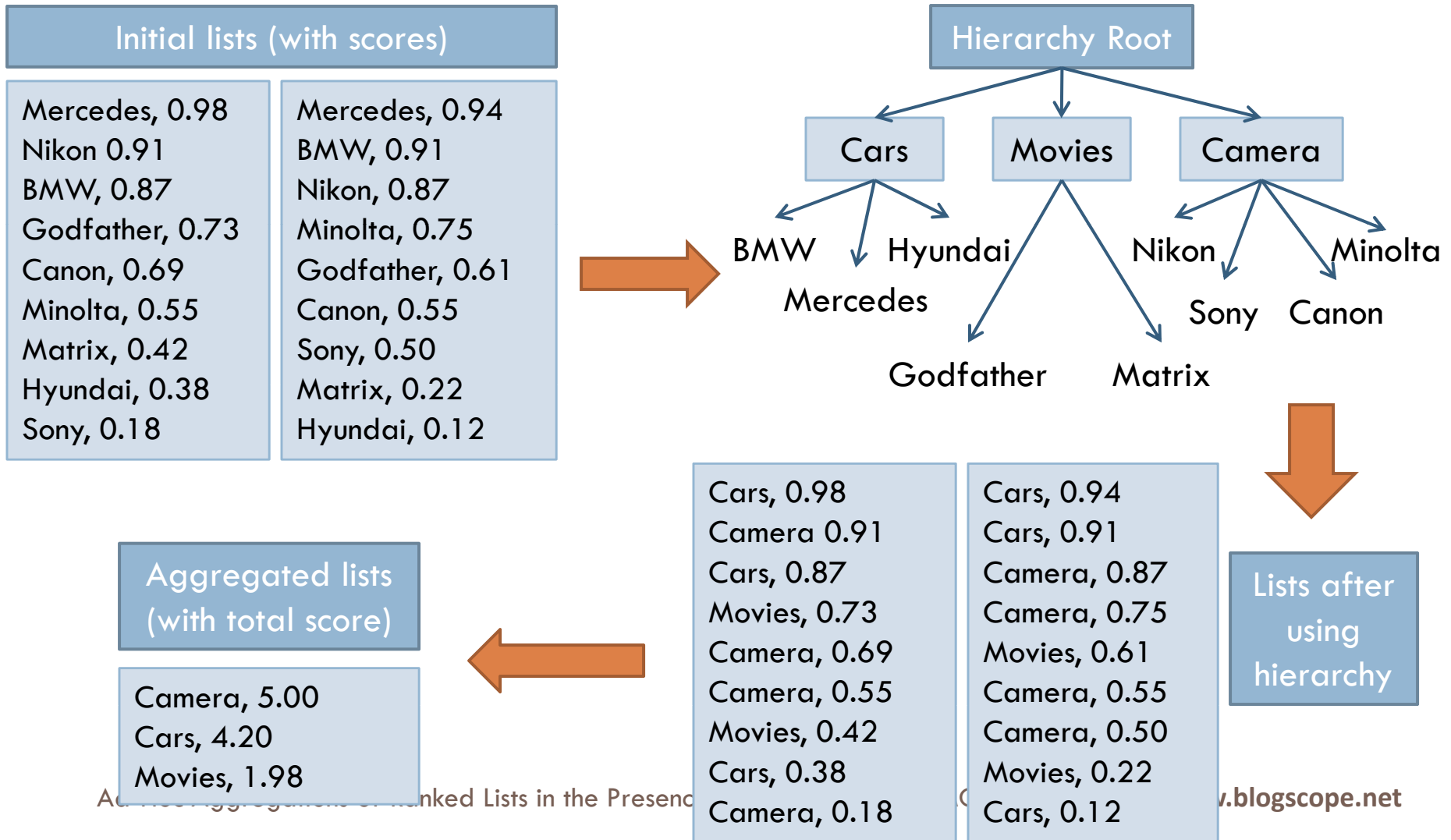
Hierarchies are Natural

7

- Dmoz, Yahoo Directory
 - ▣ Hierarchical categorization websites
- Product Stores, EBay
 - ▣ Hierarchical categorization of products
- Concept Hierarchy
 - ▣ Keywords can be mapped to topics



Top-K with Hierarchies is Interesting



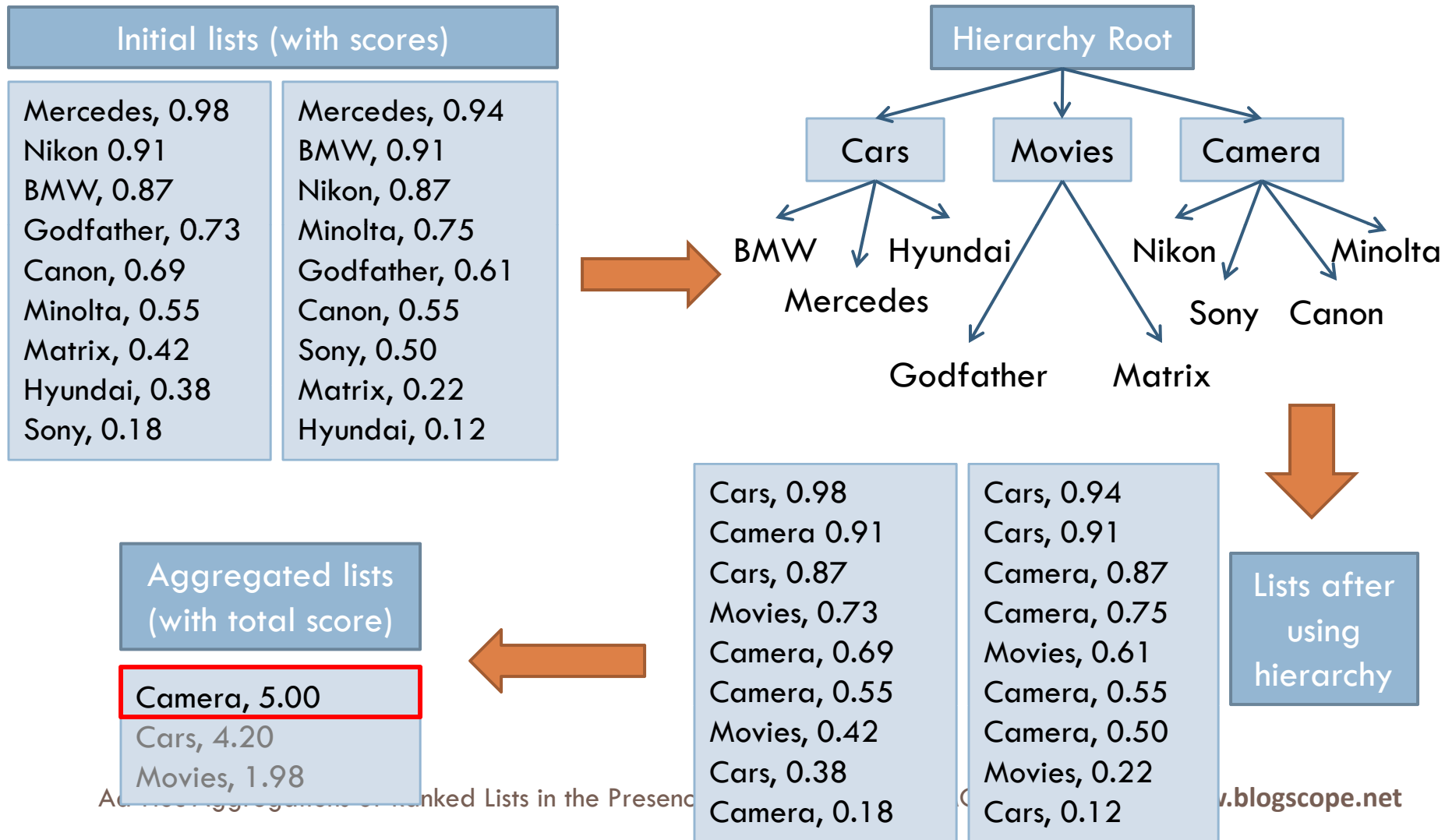
Top-K with Hierarchies is Important



9

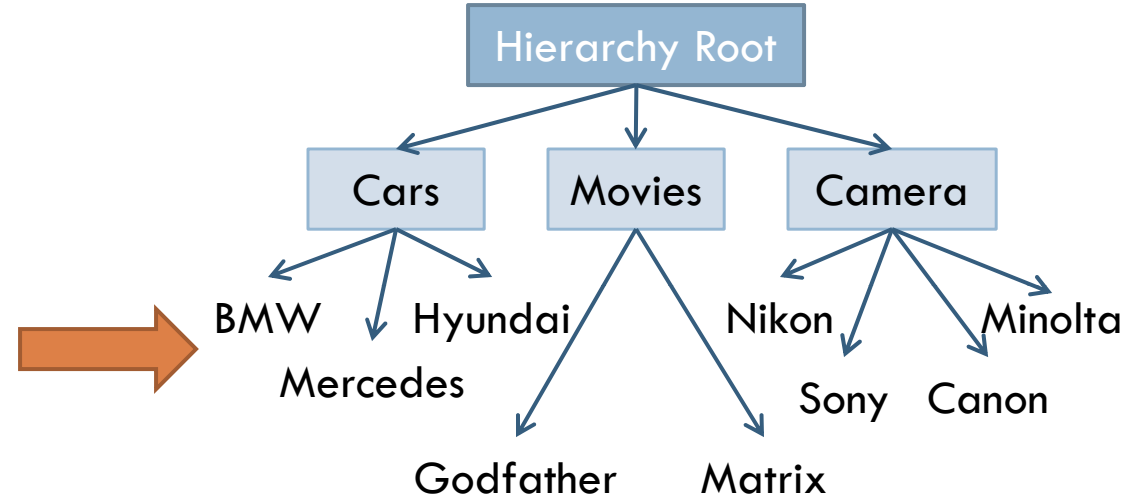
- Given website traffic rankings from Alexa and Dmoz web directory, find most visited website categories
- Given most used keywords from BlogScope, and a keyword-to-topic hierarchy, find most talked about topics
- Input
 - ▣ A collection of ranked lists (with scored items)
 - One list for each time step
 - ▣ Hierarchy
- Output: Top-K after ‘mapping’ items in input lists utilizing the hierarchy

Top-K with Hierarchies is Difficult



Top-K with Hierarchies is Difficult

Initial lists (with scores)	
Mercedes, 0.98	Mercedes, 0.94
Nikon, 0.91	BMW, 0.91
BMW, 0.87	Nikon, 0.87
Godfather, 0.73	Minolta, 0.75
Canon, 0.69	Godfather, 0.61
Minolta, 0.55	Canon, 0.55
Matrix, 0.42	Sony, 0.50
Hyundai, 0.38	Matrix, 0.22
Sony, 0.18	Hyundai, 0.12



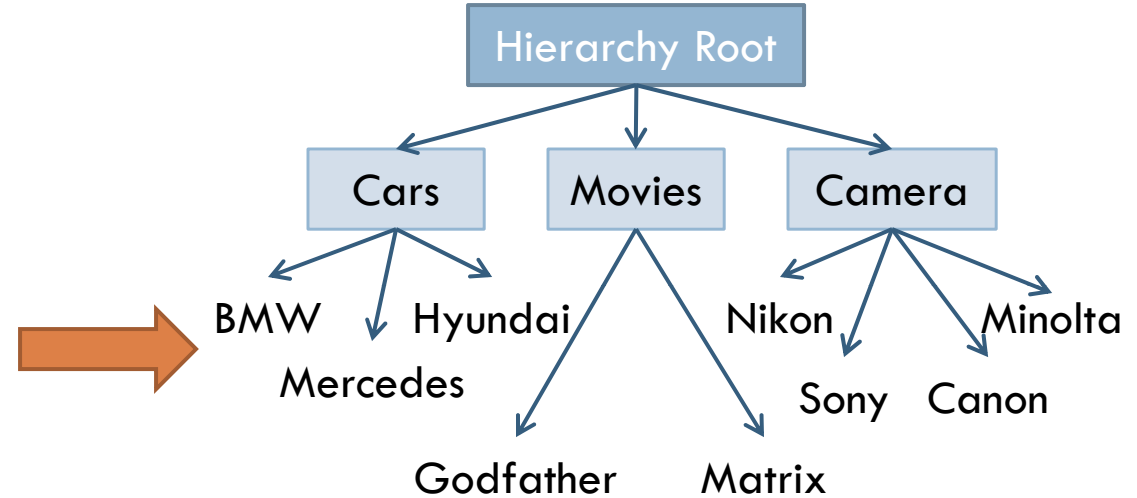
Cars, 0.98	Cars, 0.94
Camera, 0.91	Cars, 0.91
Cars, 0.87	Camera, 0.87
Movies, 0.73	Camera, 0.75
Camera, 0.69	Movies, 0.61
Camera, 0.55	Camera, 0.55
Movies, 0.42	Camera, 0.50
Cars, 0.38	Movies, 0.22
Camera, 0.18	Cars, 0.12

Aggregated lists (with total score)
Camera, 0.00
Cars, 1.92
Movies, 0.00

Lists after using hierarchy

Top-K with Hierarchies is Difficult

Initial lists (with scores)	
Mercedes, 0.98 Nikon 0.91 BMW, 0.87	Mercedes, 0.94 BMW, 0.91 Nikon, 0.87
Godfather, 0.73 Canon, 0.69 Minolta, 0.55 Matrix, 0.42 Hyundai, 0.38 Sony, 0.18	Minolta, 0.75 Godfather, 0.61 Canon, 0.55 Sony, 0.50 Matrix, 0.22 Hyundai, 0.12



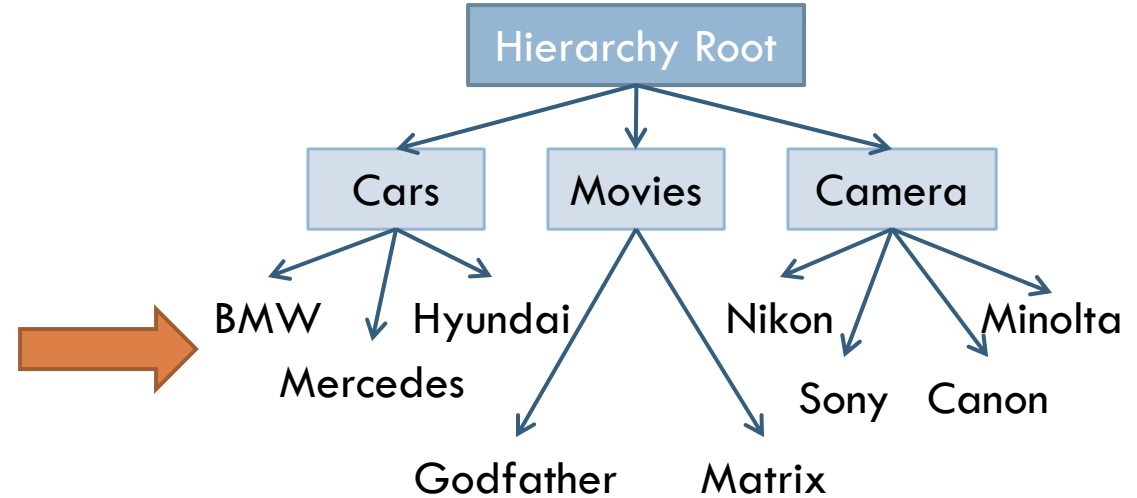
Cars, 0.98 Camera 0.91 Cars, 0.87	Cars, 0.94 Cars, 0.91 Camera, 0.87
Movies, 0.73 Camera, 0.69 Camera, 0.55 Movies, 0.42 Cars, 0.38 Camera, 0.18	Camera, 0.75 Movies, 0.61 Camera, 0.55 Camera, 0.50 Movies, 0.22 Cars, 0.12

Aggregated lists (with total score)
Camera, 1.78
Cars, 3.70
Movies, 0.00

Lists after using hierarchy

Top-K with Hierarchies is Difficult

Initial lists (with scores)	
Mercedes, 0.98	Mercedes, 0.94
Nikon 0.91	BMW, 0.91
BMW, 0.87	Nikon, 0.87
Godfather, 0.73	Minolta, 0.75
Canon, 0.69	Godfather, 0.61
Minolta, 0.55	Canon, 0.55
Matrix, 0.42	Sony, 0.50
Hyundai, 0.38	Matrix, 0.22
Sony, 0.18	Hyundai, 0.12



Cars, 0.98	Cars, 0.94
Camera 0.91	Cars, 0.91
Cars, 0.87	Camera, 0.87
Movies, 0.73	Camera, 0.75
Camera, 0.69	Movies, 0.61
Camera, 0.55	Camera, 0.55
Movies, 0.42	Camera, 0.50
Cars, 0.38	Movies, 0.22
Camera, 0.18	Cars, 0.12

Aggregated lists (with total score)
Camera, 2.53
Cars, 3.70
Movies, 1.34

Lists after using hierarchy

Outline

- Motivation
- **Formal Problem Setting**
- Stopping Conditions
 - ▣ Probabilistic Guarantees
 - ▣ Approximate
- Pre-computation
- Experiments

Formal Problem Statement

Given N lists, X_1, X_2, \dots, X_N ; and hierarchy H

Let $x_{ij} = j^{th}$ element in the i^{th} list

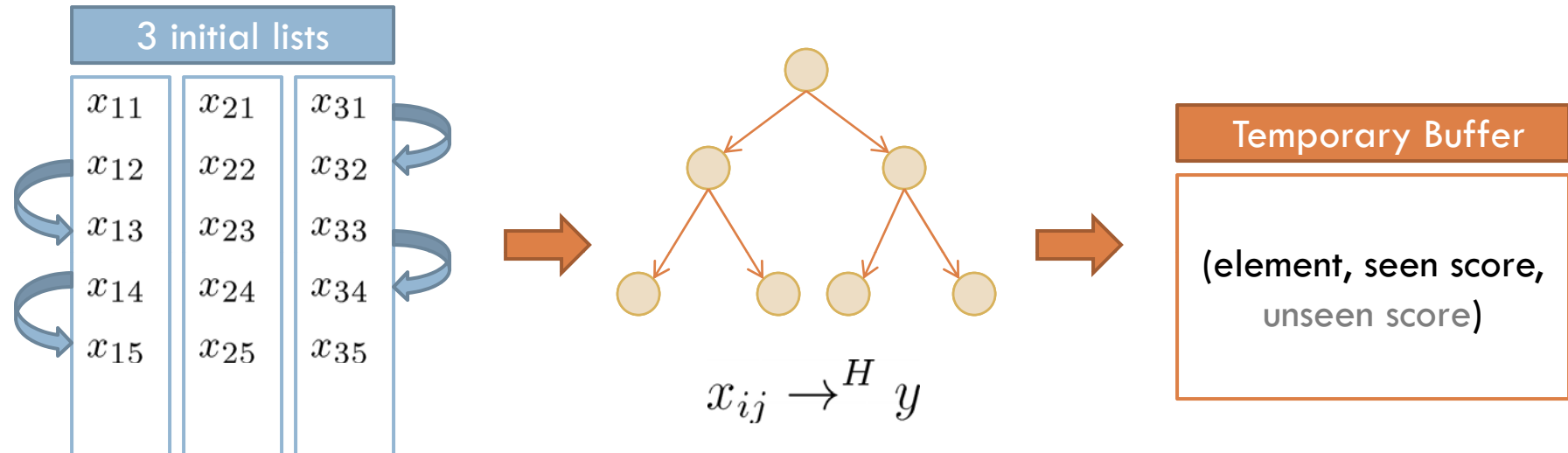
Hierarchy maps x_{ij} to y_{ij}

Hierarchy is used as a many-to-one mapping

Find top- k mapped elements $\{y\}$ with highest aggregate scores

Threshold Algorithm - NRA

16



$seen_score(y)$ = aggregate score of all seen instances of y

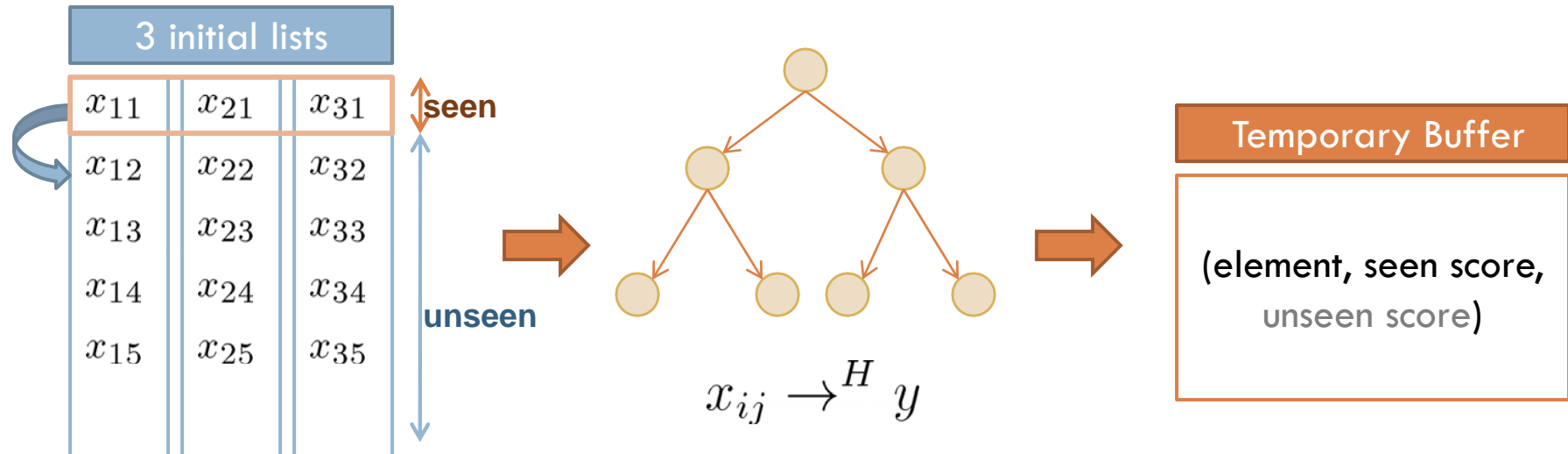
$unseen_score(y)$ = aggregate score of all unseen instances of y

$total_score(y) = seen_score(y) + unseen_score(y)$

$min-k$ = seen score of k^{th} highest element in the buffer

Threshold Algorithm - NRA

17



$seen_score(y)$ = aggregate score of all seen instances of y

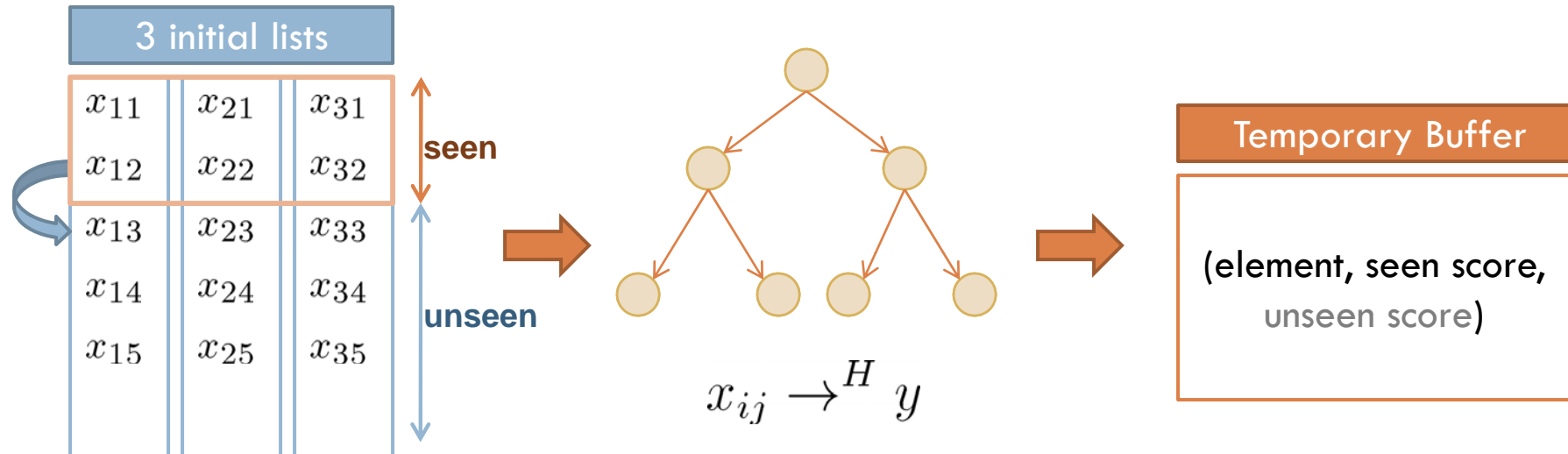
$unseen_score(y)$ = aggregate score of all unseen instances of y

$total_score(y) = seen_score(y) + unseen_score(y)$

min- k = seen score of k^{th} highest element in the buffer

Threshold Algorithm - NRA

18



$seen_score(y)$ = aggregate score of all seen instances of y

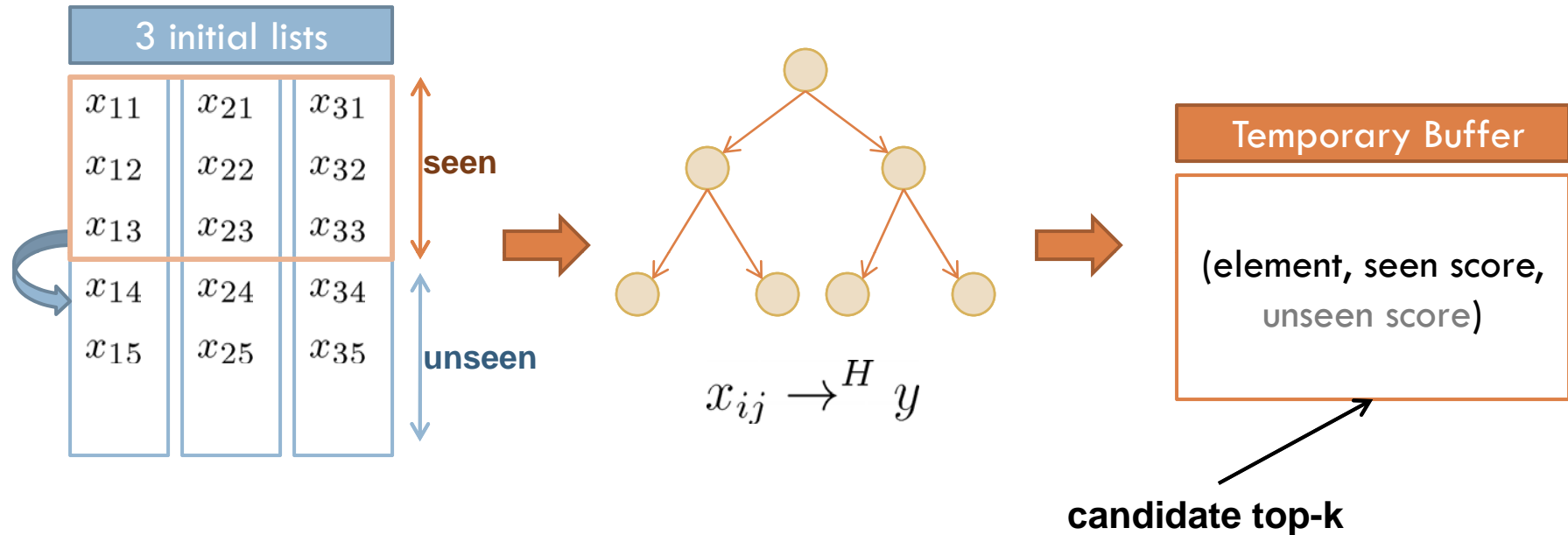
$unseen_score(y)$ = aggregate score of all unseen instances of y

$total_score(y) = seen_score(y) + unseen_score(y)$

$min-k$ = seen score of k^{th} highest element in the buffer

Threshold Algorithm - NRA

19



$seen_score(y)$ = aggregate score of all seen instances of y

$unseen_score(y)$ = aggregate score of all unseen instances of y

$total_score(y) = seen_score(y) + unseen_score(y)$

min- k = seen score of k^{th} highest element in the buffer

Outline

- Motivation
- Formal Problem Setting
- **Stopping Conditions**
 - **Probabilistic Guarantees**
 - Approximate
- Pre-computation
- Experiments

Stopping Condition

- Exact top-k: Terminate when $\text{min-}k > \text{total_score}(y)$ for all y not in candidate top-k
 - Very conservative, costly
- **1. Probabilistic Guarantee**
 - With some confidence, say 95%
 - [TWS04] Theobald et al, Top-k query evaluation with probabilistic guarantees, VLDB 2004
- **2. Approximate top-k**
 - Stopping condition with relaxed precision guarantees

Stopping Condition - Probabilistic



22

- Exact top-k – Stopping Condition
 - **Pr[y belongs to exact top-k] = 0** for all y not in candidate top-k

Stopping Condition - Probabilistic



23

- Exact top-k – Stopping Condition
 - $\Pr[y \text{ belongs to exact top-}k] = 0$ for all y not in candidate top-k
- Probabilistic Guarantee – Stopping Condition
 - **$\Pr[y \text{ belongs to exact top-}k] < e$**
 - For constant e ; confidence = $1 - e = 0.95$

Stopping Condition - Probabilistic



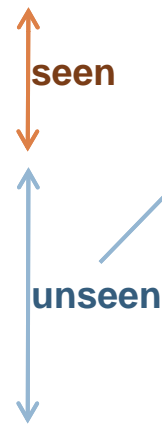
24

- Exact top-k – Stopping Condition
 - $\Pr[y \text{ belongs to exact top-}k] = 0$ for all y not in candidate top-k
- Probabilistic Guarantee – Stopping Condition
 - $\Pr[y \text{ belongs to exact top-}k] < e$
 - For constant e ; confidence = $1 - e = 0.95$
 - $\Pr[\text{unseen_score}(y) < \text{min-}k - \text{seen_score}(y)] < e$
- min-k and seen_score(y) are known
 - Need to estimate (probability distribution) of unseen_score(y)

Probabilistic TA with Hierarchies

Terminate when $\Pr[\text{unseen_score}(y) < \text{min-}k - \text{seen_score}(y)] < \epsilon$
for all y not in candidate top- k

3 mapped lists		
y_{11}	y_{21}	y_{31}
y_{12}	y_{22}	y_{32}
y_{13}	y_{23}	y_{33}
y_{14}	y_{24}	y_{34}
y_{15}	y_{25}	y_{35}



- Estimate $\text{unseen_score}(y)$
- Knowledge of max number of times $\{y\}$ can appear in a single list
 - ▣ Possible by single scan of the hierarchy
- Knowledge of score distribution in the lists required



Parametric Distribution

- Assume parametric distribution of scores of elements in initial lists
 - Geometric distribution
- Compute distribution of *unseen_score(y)*
 - Possible using numerical tools like Matlab
 - High computational overhead, practically not viable
 - Details in the paper

Pre-computed Histogram

- Maintain a histogram for each initial list
 - ▣ Distribution of scores
- Compute distribution of *unseen_score(y)*
 - ▣ Possible, but needs convolutions of multiple distributions
 - High computational overhead, practically not viable
 - ▣ Details in the paper
- [TWS2004] did not have hierarchies
 - ▣ Number of convolutions required is much less without hierarchies
 - ▣ Top-K with hierarchies is difficult

Outline

- Motivation
- Formal Problem Setting
- Stopping Conditions
 - Probabilistic Guarantees
 - **Approximate**
- Pre-computation
- Experiments

Approximate top-k with Hierarchies

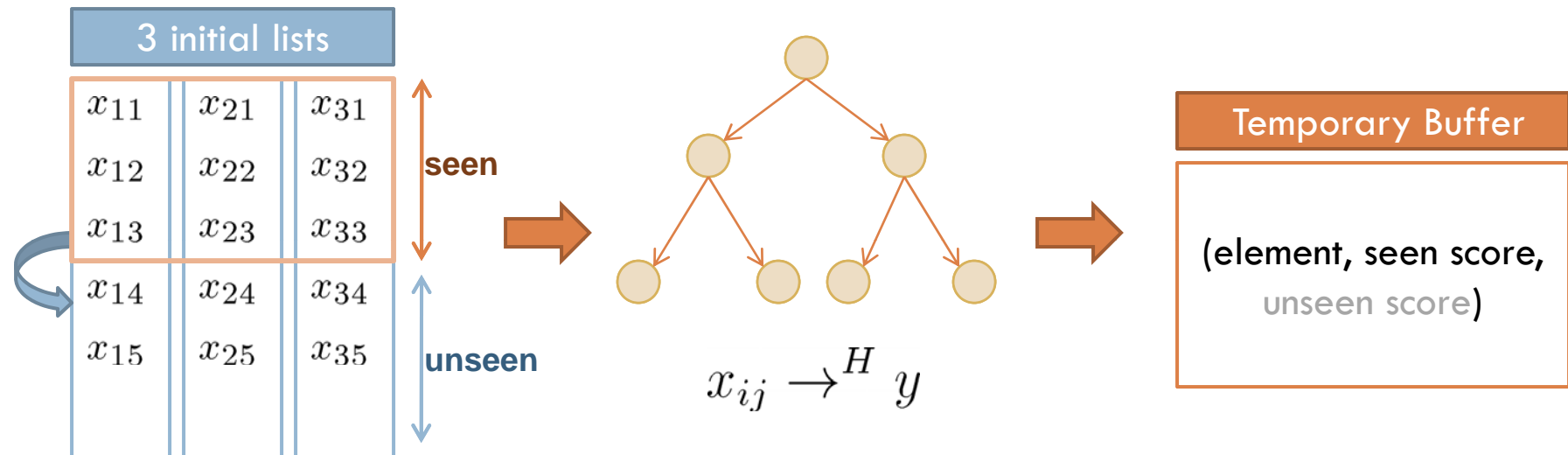


29

- Relaxed precision guarantees
 - But with 100% confidence, no probabilities
- Guarantee: **precision** $> \pi$
 - For **target precision** $\pi < 1$
 - Out of k elements in result set, at least k' will be in true top- k , where $k'/k > \pi$
 - Precision = k'/k = correct results / total results
 - As π is increased, accuracy increases at cost of performance

Approximate top-k Algorithm

30



Terminate when less than $k-k'$ elements not in candidate top-k have their $total_score(y) > min-k$

For exact top-k, $k-k'$ is equal to 0

Outline

- Motivation
- Formal Problem Setting
- Stopping Conditions
 - ▣ Probabilistic Guarantees
 - ▣ Approximate
- **Pre-computation**
- Experiments

Pre-computation

- N total input lists, one for each time step
- Query on s consecutive lists
- Example: One list of popular keywords for each day of year 2007, query for popular keywords during first week of April 2007
 - ▣ $N = 365; s=7$
- Only consecutive lists are aggregated
 - ▣ Can not compute top-k over one list from 12th April and other from 4th May

Precomputation – Sparse System



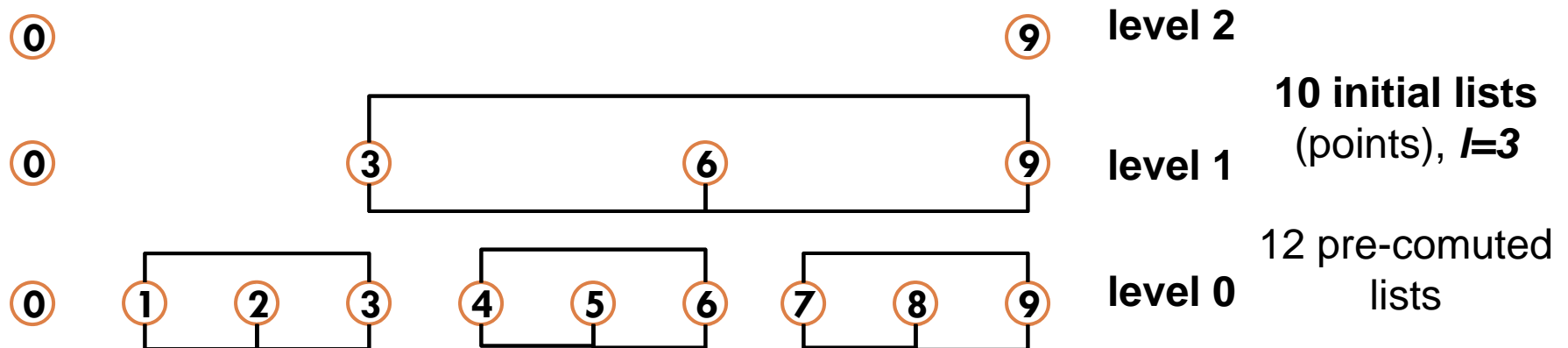
33

- Since lists are available in advance, pre-computation is possible
 - ▣ Hierarchy is supplied at query time
 - Different analysts have different hierarchies
- Sparse Interval Set System
 - ▣ [GKS02] Guha, Koudas, Srivastava, Fast algorithms for hierarchical range histogram construction, PODS 2002

Sparse Set Construction

Given N lists, each one corresponding to a point on number line
Define level j points as $0, l^j, 2l^j, \dots$. Any pair of level j points
between adjacent level $j + 1$ points defines an interval

- For each interval, maintain pre-computed list
 - ▣ [1,2], [2,3], [1,3], [3,6], [4,6], [3,9]

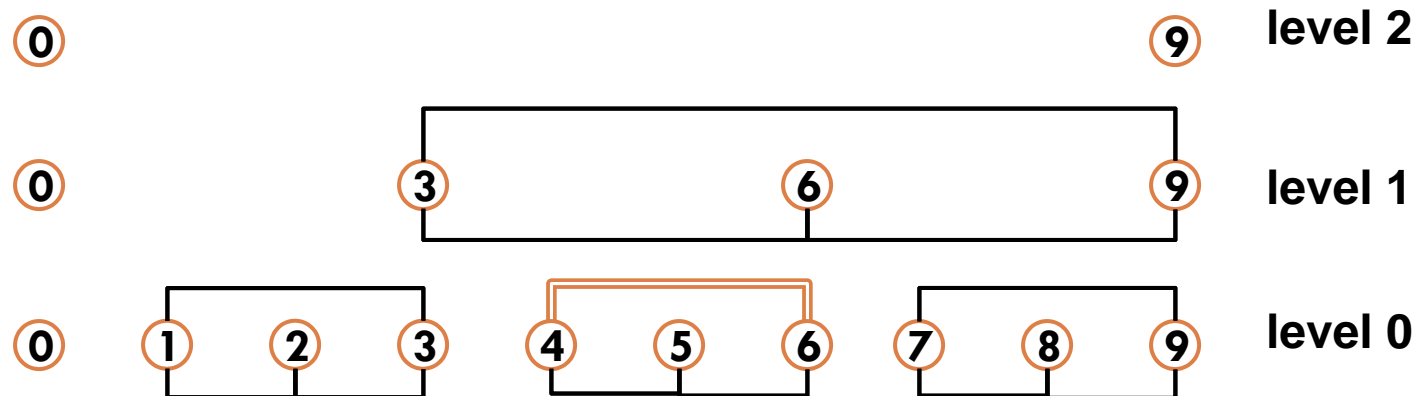
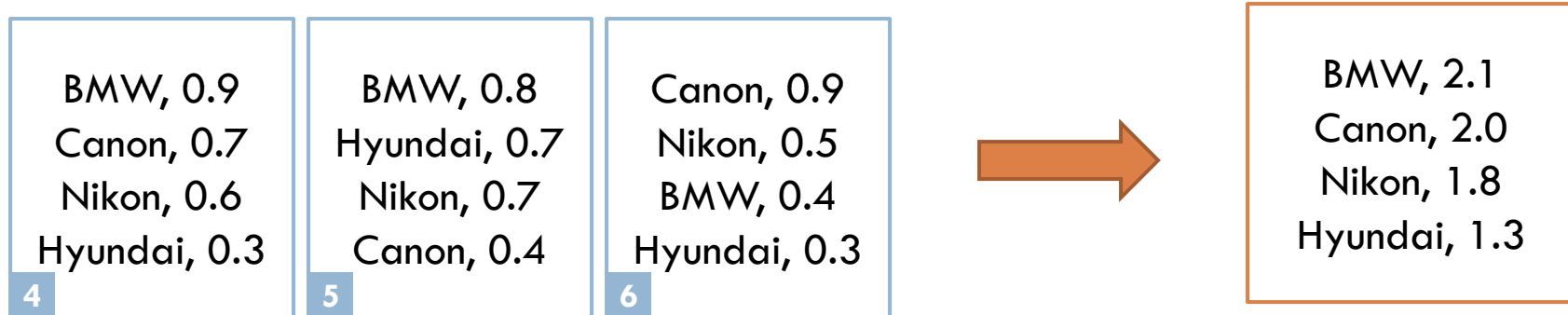


Sparse Set Example

35

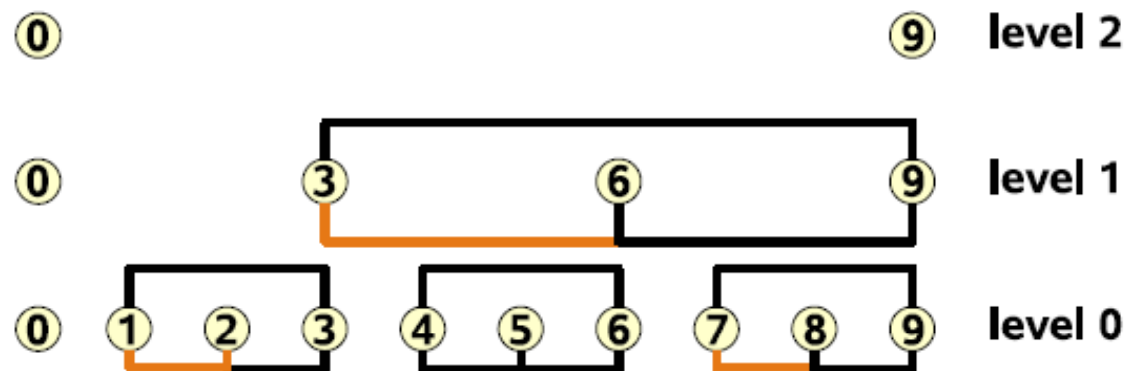
Interval [4,6]:

- Aggregate lists 4, 5, and 6 in advance (without hierarchy)



Querying with Sparse Set

- Assume query for time interval $[1,8]$
 - Requires aggregation over three pre-computed lists
 - $[1,2]$
 - $[3,6]$
 - $[7,8]$
 - As opposed to over 8 lists without sparse set



Sparse Set Guarantees

- Performance guarantee, tunable parameter l
 - ▣ speed vs. pre-computation tradeoff
- For query over s consecutive lists
 - ▣ Aggregation of less than $2\log_l s + 1$ lists required
- Worstcase pre-computation overhead guarantee
 - ▣ No list will participate in more than $O(l^2 \log_l n)$ intervals
- Amenable to streaming maintenance
 - ▣ Sparse set is easily updatable when a new list is added to the system

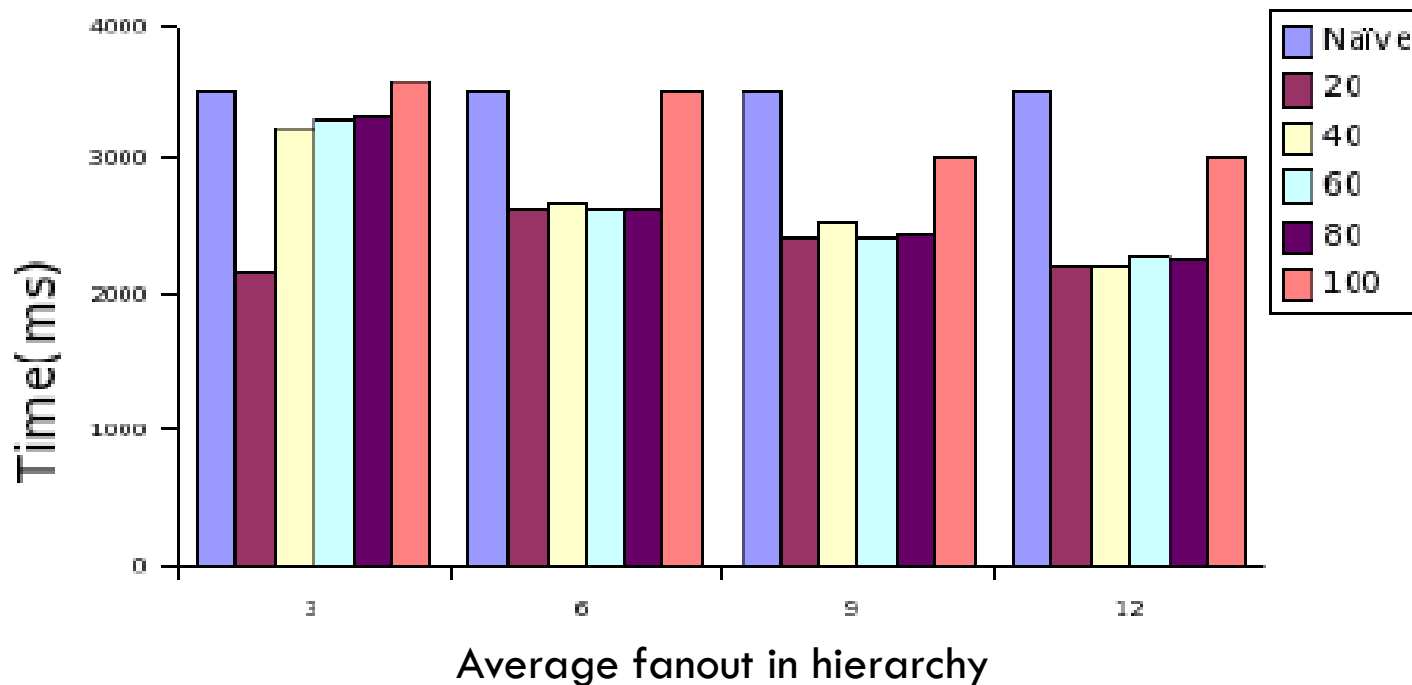
Outline

- Motivation
- Formal Problem Setting
- Stopping Conditions
 - ▣ Probabilistic Guarantees
 - ▣ Approximate
- Pre-computation
- **Experiments**

Experimental Setup

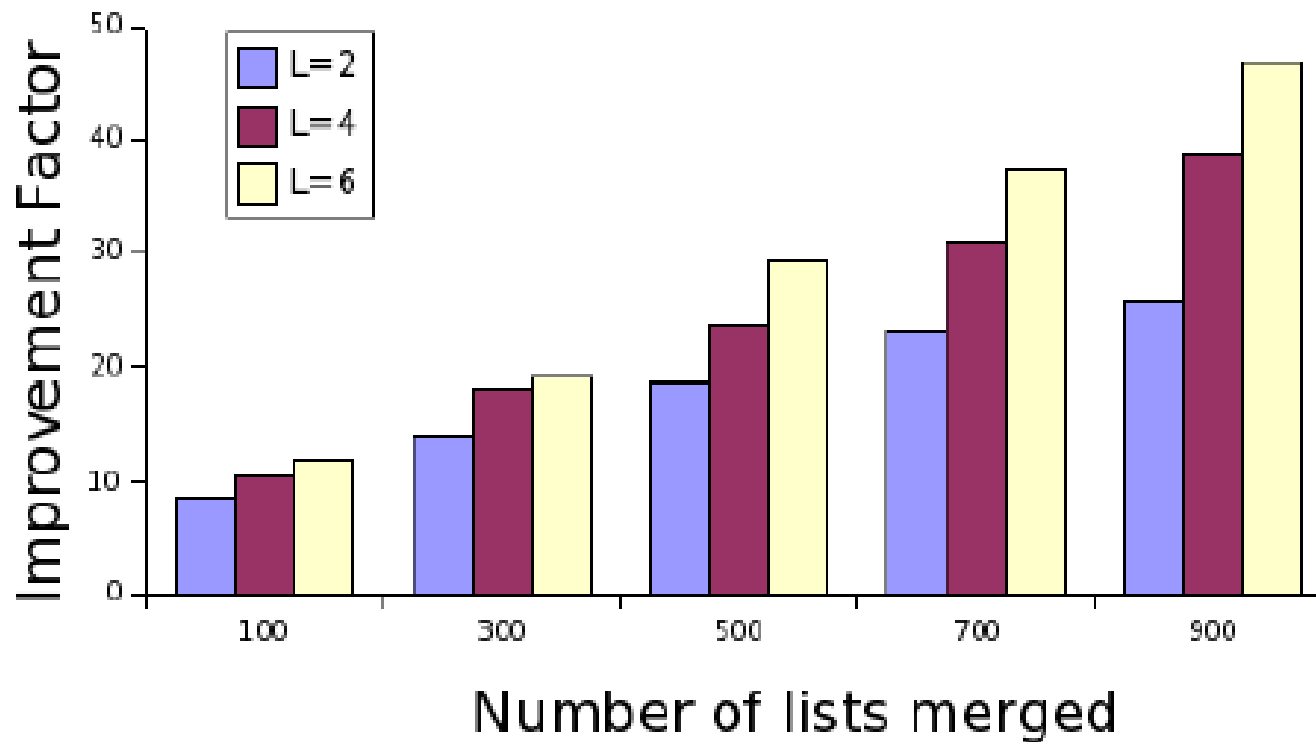
- Probabilistic top-k is not practical in our setting
- Approximate top-k
 - ▣ Performance and accuracy tradeoffs
- Sparse Set - huge performance improvements
- Real data from BlogScope www.blogscope.net
 - ▣ Live blog search and analysis engine
 - ▣ Tracking over 30M blogs, 350M posts, 3TB database
 - ▣ Produces list of top-k keywords from blogosphere, one for each day

Approximate top-k Performance



Aggregating 9 lists from BlogScope, each consisting of 50K keywords, with synthetic hierarchy having different fanout

Sparse Set - Improvement



Conclusions

- Introduced the problem of Top-K with Hierarchies
- Difficult problem to solve with 100% accuracy
- Probabilistic approach is not practical with hierarchies
- Approximate top-k leads to performance gains
- Pre-computation can help a lot
- Future work
 - ▣ Aggregation over multiple levels of hierarchies
 - ▣ Maintenance of continuous top-k in a streaming setting

43

Thanks!

Nilesh Bansal, Sudipto Guha, Nick Koudas

