Efficient Semantic-Aware Detection of Near Duplicate Resources

7th Extended Semantic Web Conference

E. Ioannou, O. Papapetrou, D. Skoutas, W. Nejdl

Wednesday, 2nd June 2010, Heraklion, Greece

Outline

1. **Motivation**
2. RDFsim Approach
   - Resource representation
   - Indexing structure
   - Querying for near duplicate resources
3. Experimental Evaluation
4. Conclusions

Motivation

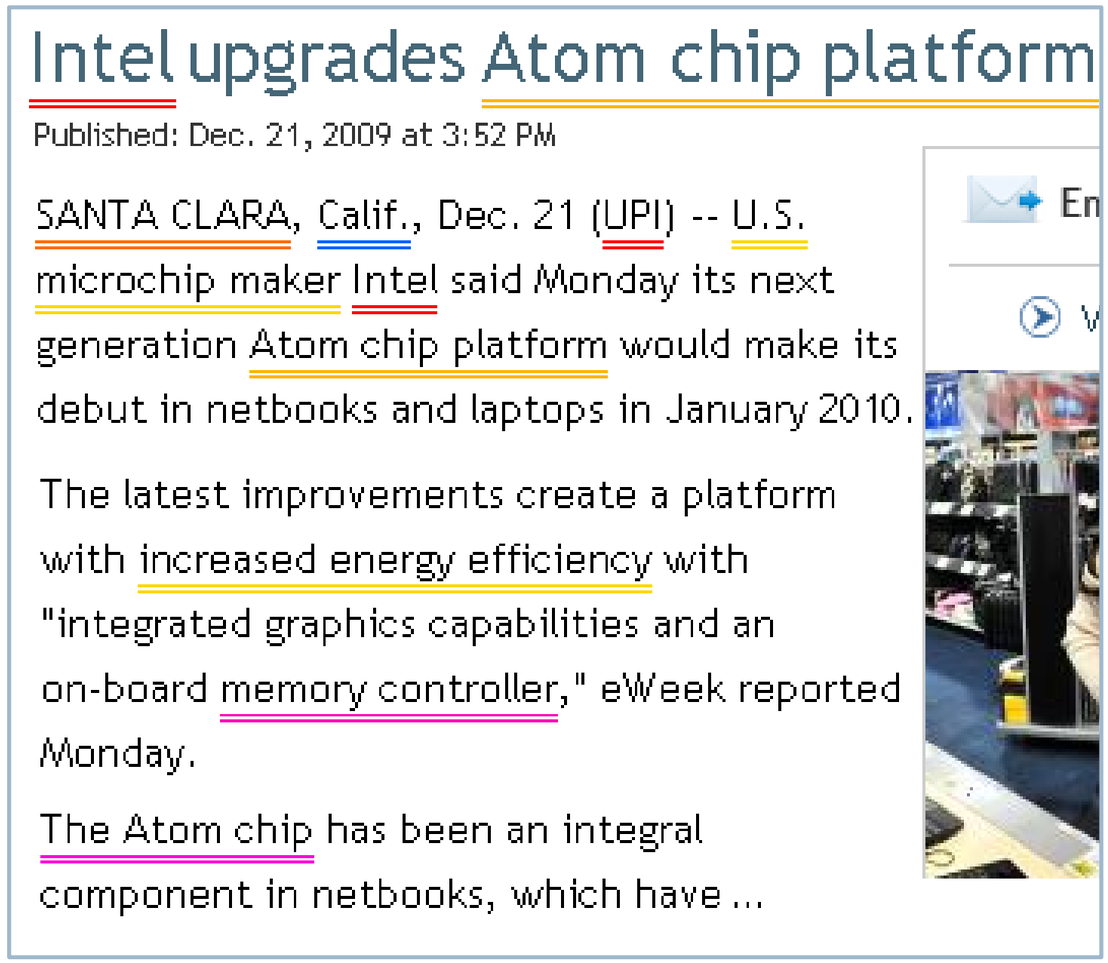
- Plethora of current Semantic and Social Web applications that integrate data from various sources
- BUT data is overlapping or complementary

➔ Detect near duplicate data:

- group, merge, remove resources
- avoid repetition and redundancy

Motivation - News Aggregation Service

- Aggregate articles from a large number of news agencies
- Republish same articles, include slight changes, spelling mistakes, an additional image, or some new information
- RDF data from extractors → entities, relationships, …



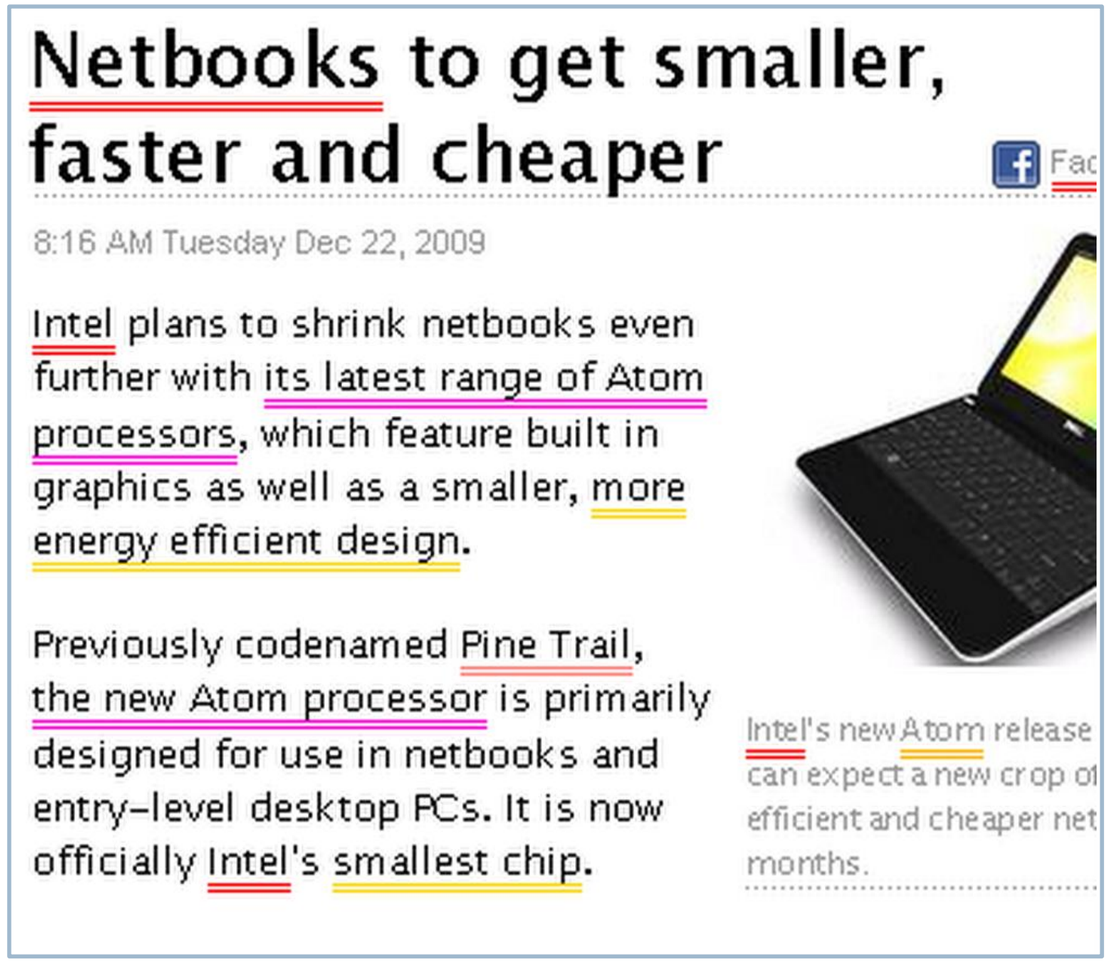
Intel upgrades Atom chip platform

Published: Dec. 21, 2009 at 3:52 PM

SANTA CLARA, Calif., Dec. 21 (UPI) -- U.S. microchip maker Intel said Monday its next generation Atom chip platform would make its debut in netbooks and laptops in January 2010.

The latest improvements create a platform with increased energy efficiency with "integrated graphics capabilities and an on-board memory controller," eWeek reported Monday.

The Atom chip has been an integral component in netbooks, which have ...



Netbooks to get smaller, faster and cheaper

8:16 AM Tuesday Dec 22, 2009

Intel plans to shrink netbooks even further with its latest range of Atom processors, which feature built in graphics as well as a smaller, more energy efficient design.

Previously codenamed Pine Trail, the new Atom processor is primarily designed for use in netbooks and entry-level desktop PCs. It is now officially Intel's smallest chip.

Intel's new Atom release can expect a new crop of efficient and cheaper net months.

Motivation - News Aggregation Service

Detecting Near Duplicate RDF Resources:

→ compute similarity and select based on requirements

Two main issues:

a) How to compute the similarity between a pair of RDF Resources ?

b) How to efficiently compare resources ?

- Avoid pairwise comparisons
- Allow on-the-fly operation

Outline

1. Motivation
2. **RDFsim Approach**
   - Resource representation
   - Indexing structure
   - Querying for near duplicate resources
3. Experimental Evaluation
4. Conclusions

RDFsim Approach

- Each resource R is an RDF graph
  - Set of RDF triples

- $\mathcal{R}$ is the set of all available resources
- Function computing similarity *sim*: $\mathcal{R}$ x $\mathcal{R}$ → [0,1]

R_1 & R_2 are near duplicates: $sim(R_1, R_2) \geq minSim$

Resource Representation

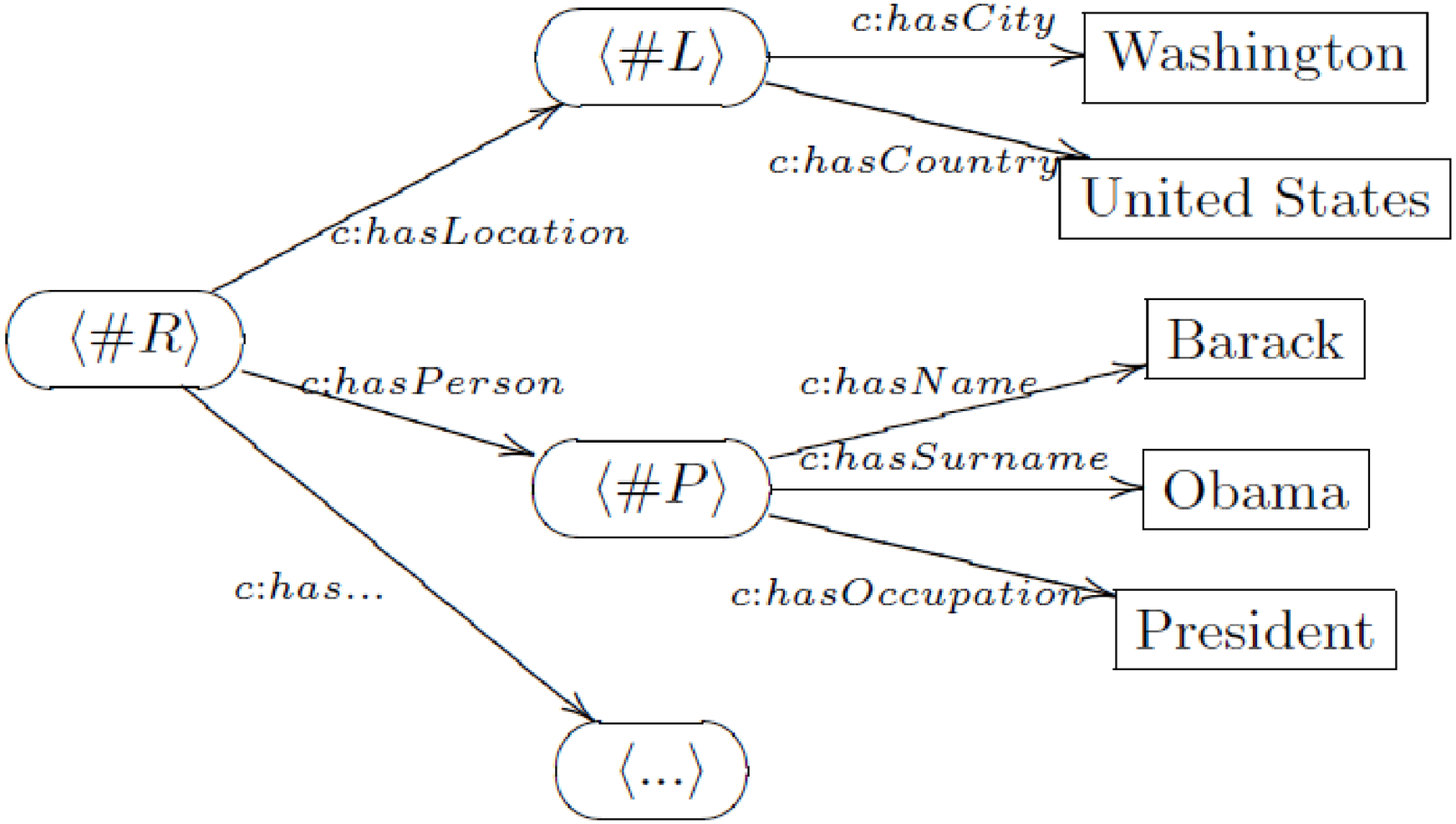
Representation is denoted with *rep(R)*

RDFsim applies a transformation of the RDF graph:

- for each triple
  → concatenate predicate with object

- if object is another RDF triple Ry
  → union with *rep*(Ry)

Resource Representation - Example



rep(L) = { "c:hasCity, Washington", "c:hasCountry, United States" }

rep(P) = { "c:hasName, Barack", "c:hasSurname, Obama",
"c:hasOccupation, President" }

rep(R) = { "c:hasLocation, L", "c:hasPerson, P", . . . } U *rep*(L) U *rep*(P)

Indexing structure

- Based on the Locality Sensitive Hashing (LSH)
- Indexing structure $\mathcal{I}$ that consists of l binary trees:
  - $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_l$
- Each tree is bound to k hash function:
  - $\mathcal{T}_i \rightarrow \mathrm{h}_{1,i}, \mathrm{h}_{2,i}, \ldots, \mathrm{h}_{k,i}$

Adding new resource

A. Extract *rep*(Rx)

B. Compute l labels of length k for each binary tree

B.1. Hash all terms in *rep*(Rx) using each hash function hi,j(.)

B.2. Detect the minimum hash value produced by hi,j(.)

B.3. Map *min*(hi,j(.)) to a bit

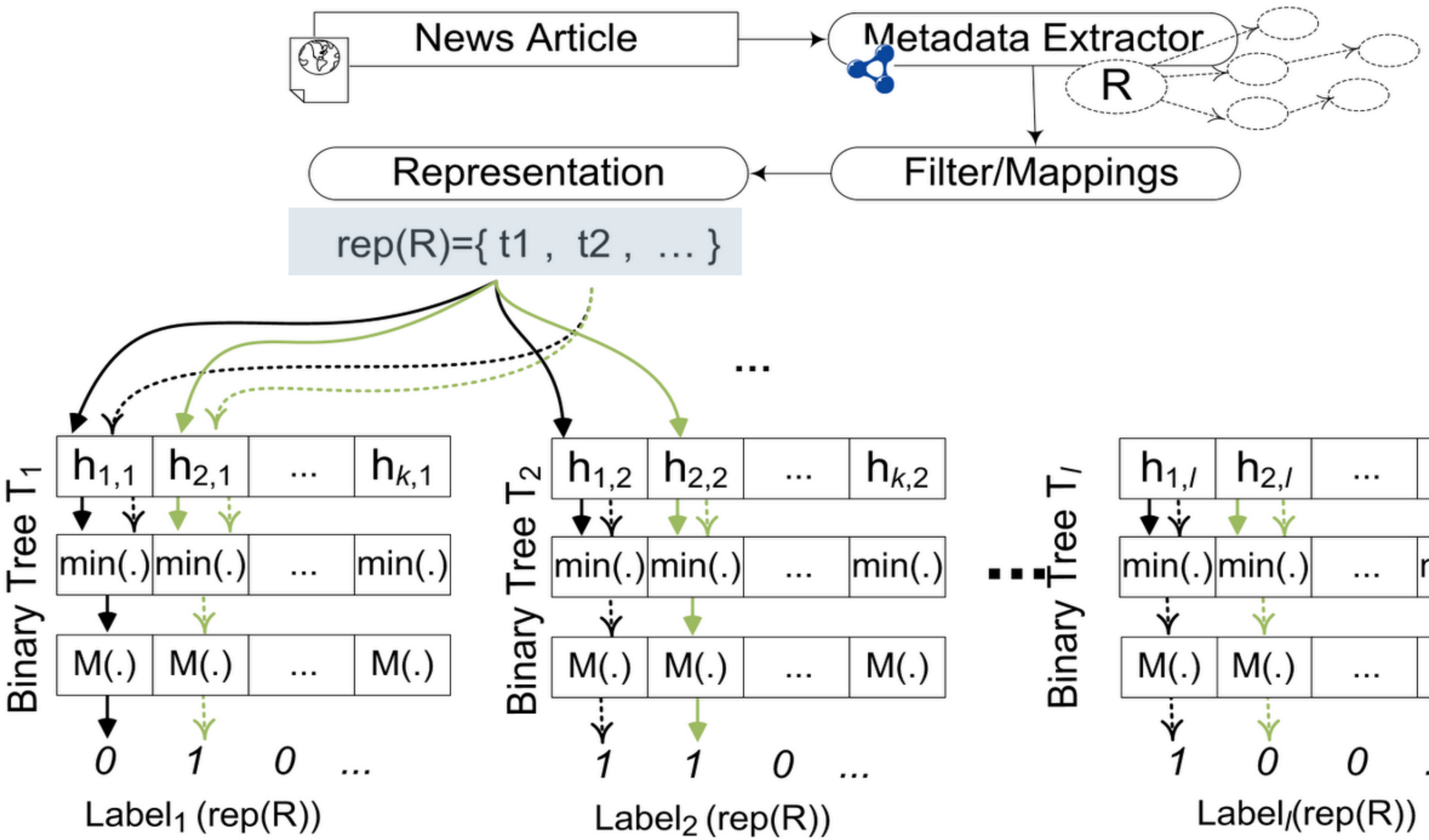
B.4. Use result as the i'th bit of the label of *rep*(Rx)

C. Insert labels in the trees

example in next slides …

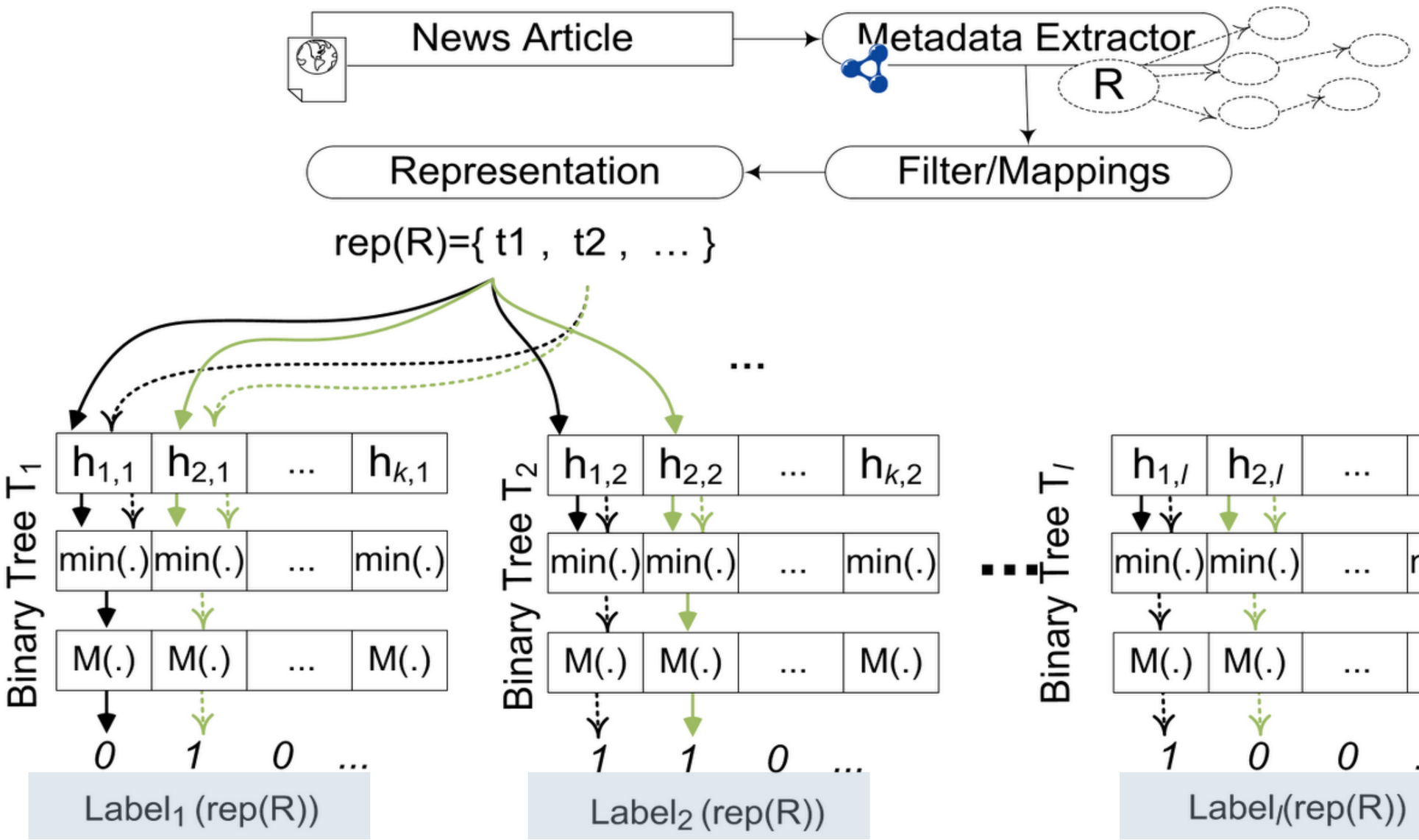
Adding new resource

A. Extract *rep*(Rx)



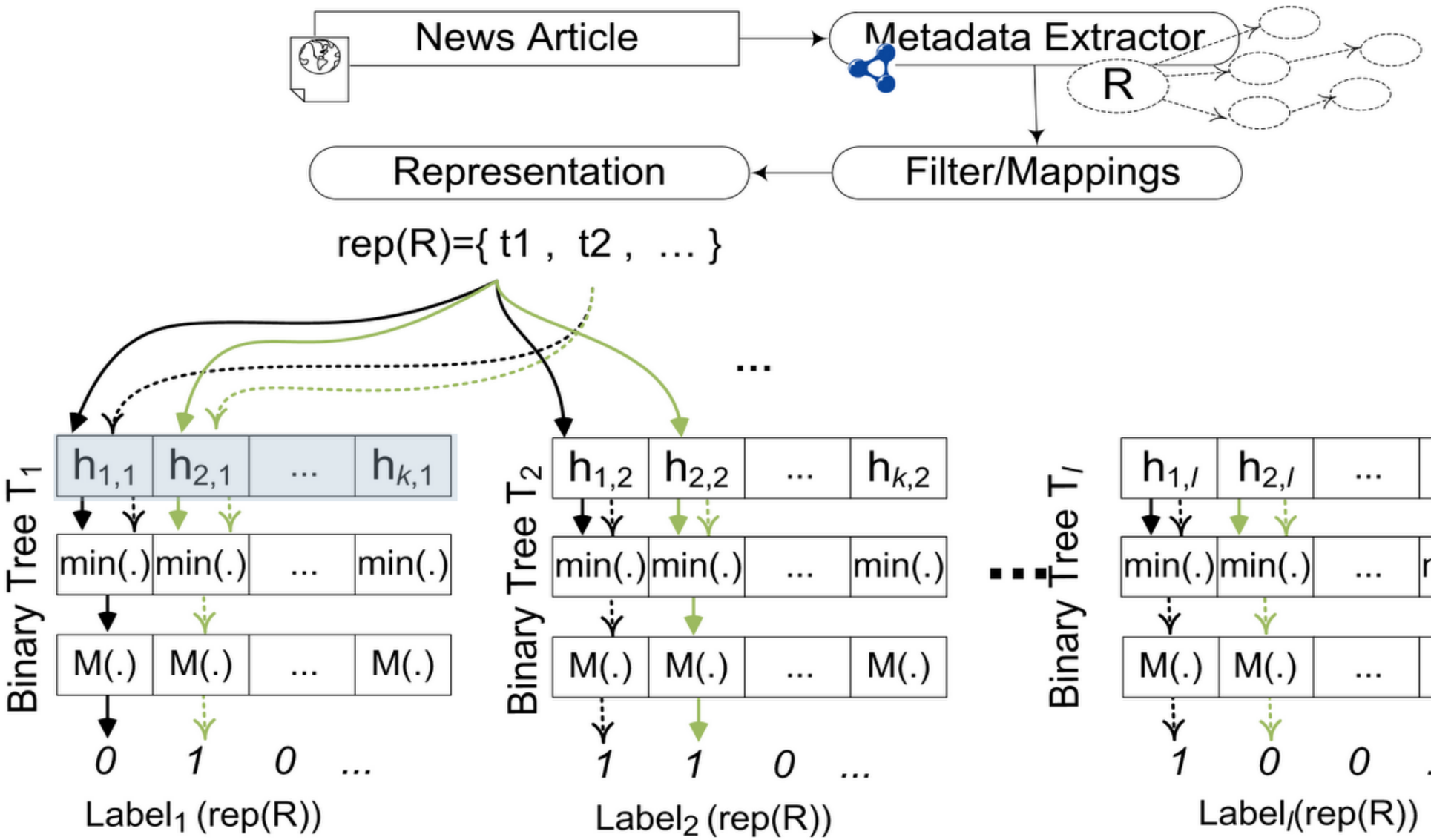
Adding new resource

B. Compute l labels of length k for each binary tree



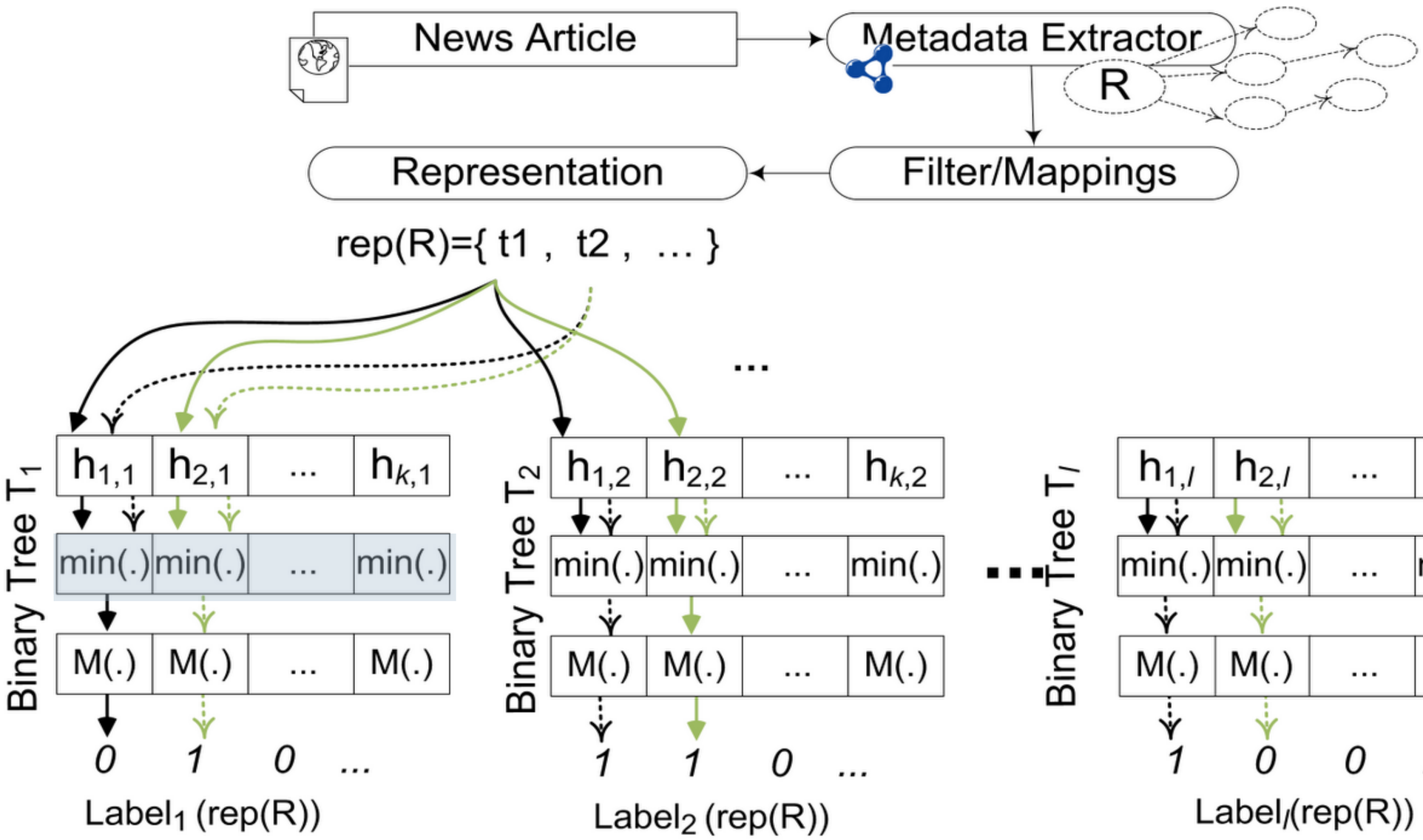
Adding new resource

B.1. Hash all terms in *rep*(Rx) using each hash function hi,j(.)



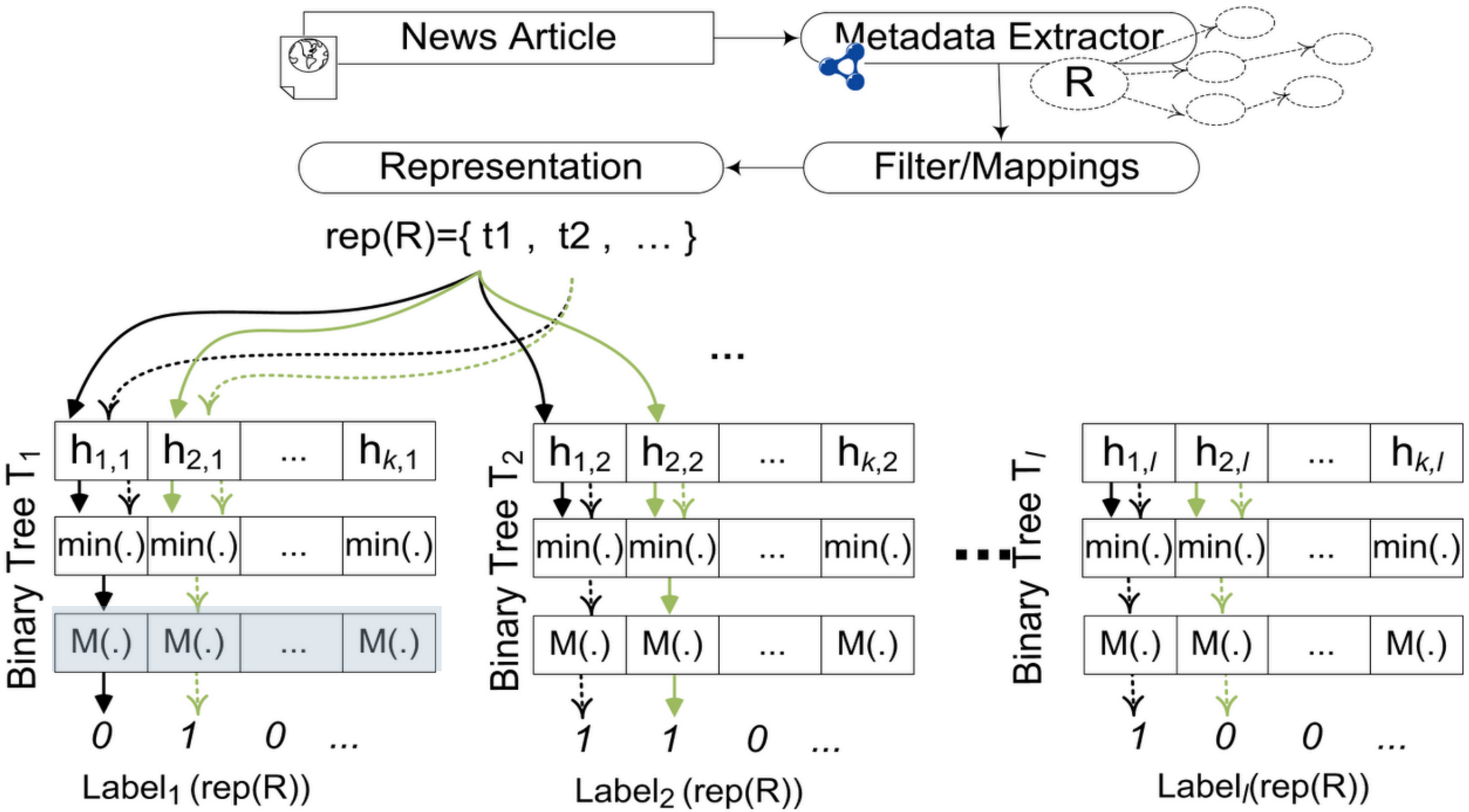
Adding new resource

B.2. Detect the minimum hash value produced by {hi,j(.)}
for all i=1…k, j=1…l → *min*(hi,j(.))



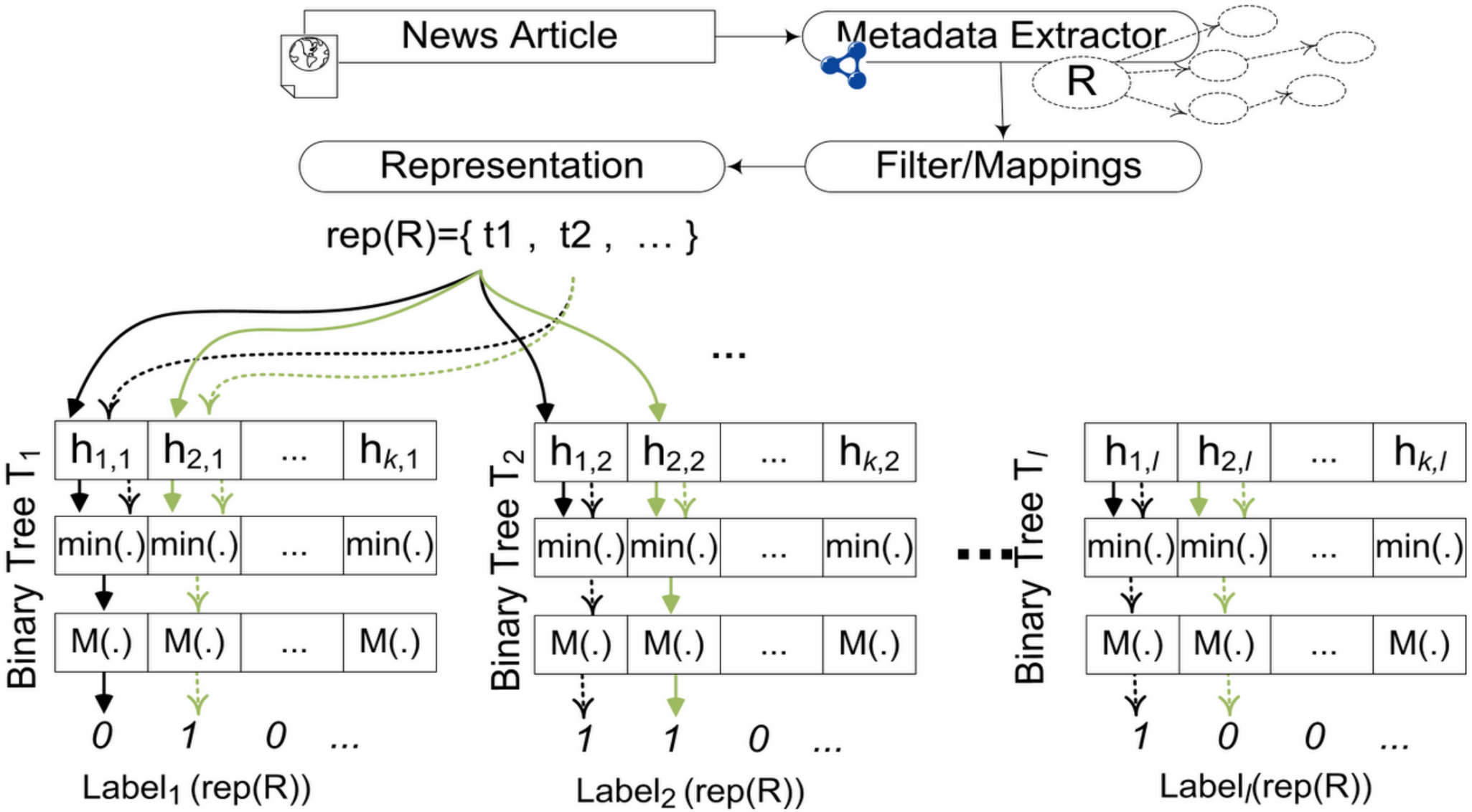
Adding new resource

B.3. Map *min*(hi,j(.)) to a bit 0 or 1



Adding new resource

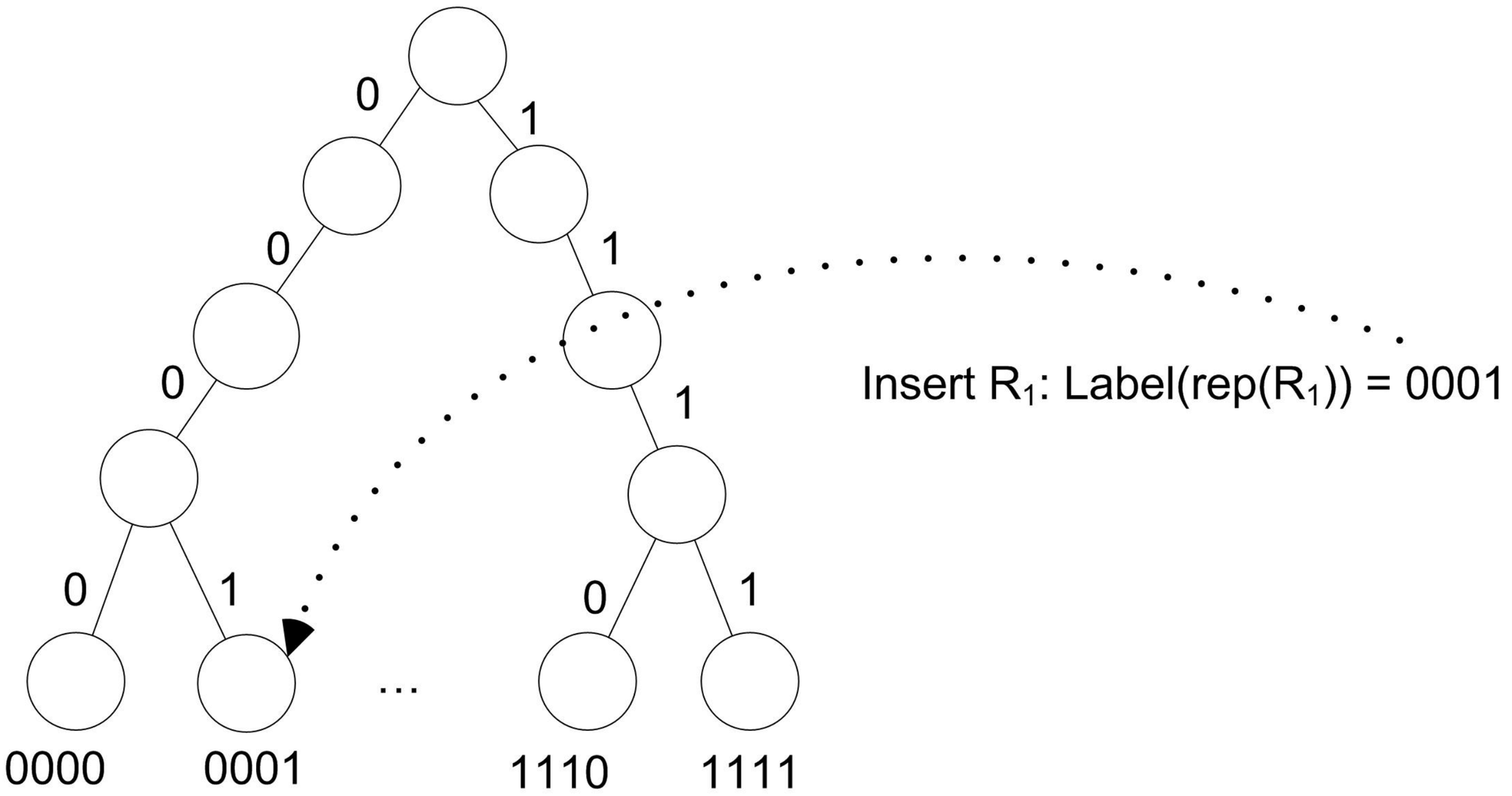
B.4. Use result as the i'th bit of the label of *rep*(Rx)



Adding new resource

C. Insert labels in the trees

i.e., Label$i$(*rep*(Rx)) → binary label for Ti



Querying for near duplicate resources

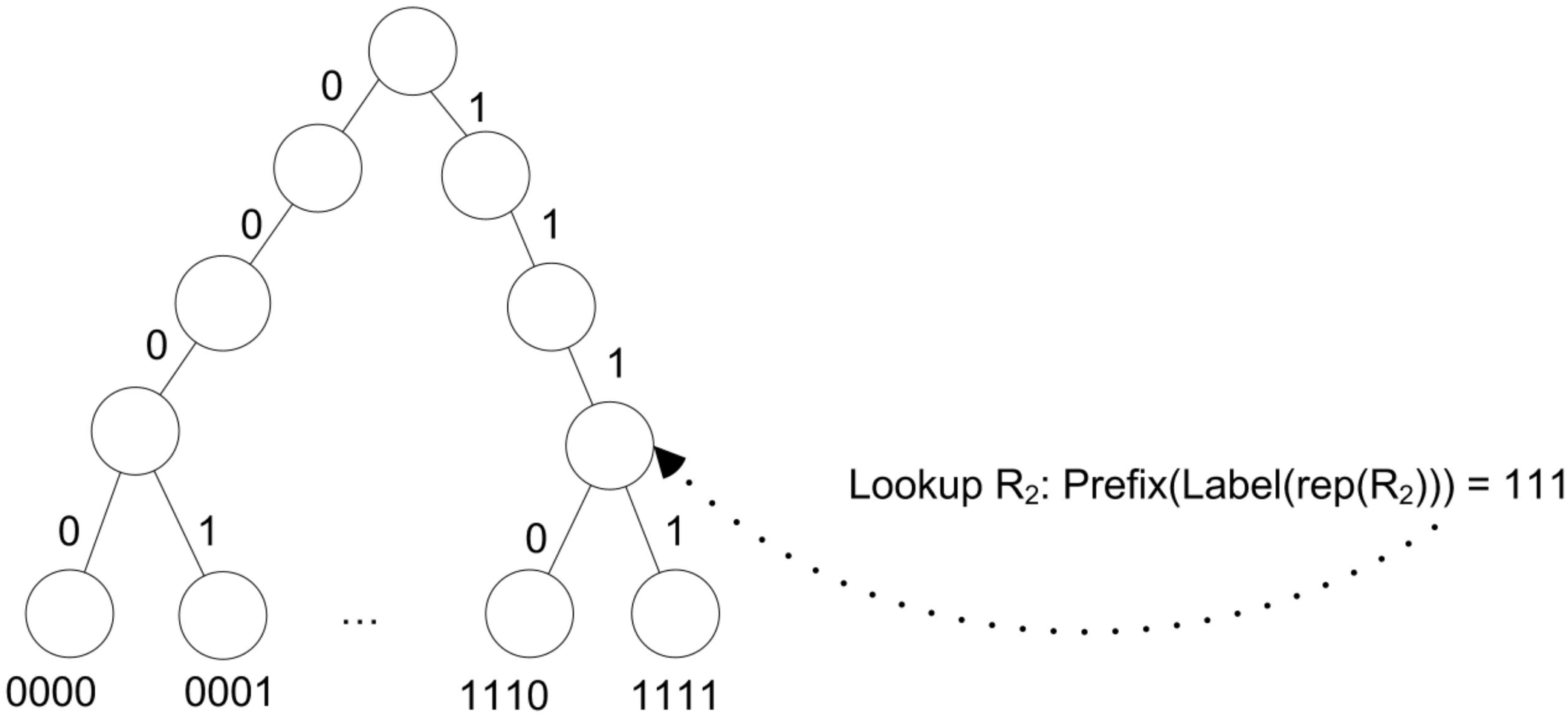
- Create the labels for each tree $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_l$
- Similar resources are indexed at nearby nodes in the tree with high probability

  → selection criterion can be relaxed

  i.e., prexfix lookup with length k'
- We set k' that allows detection with probability equal or higher to the requested *minProb* (see paper)
- We retrieve from each tree the resources
- Return the union

Querying for near duplicate resources

Example: retrieve resource from tree



Outline

1. Motivation
2. RDFsim Approach
   - Resource representation
   - Indexing structure
   - Querying for near duplicate resources
3. **Experimental Evaluation**
4. Conclusions

Experimental Evaluation

Dataset (available online) :

- Crawled news articles from the Google News Web site (e.g., BBC, Reuters, and CNN)
- RDF statements using the Open Calais Web service
- 94.829 news articles with 2.711.217 entities (RDF data)

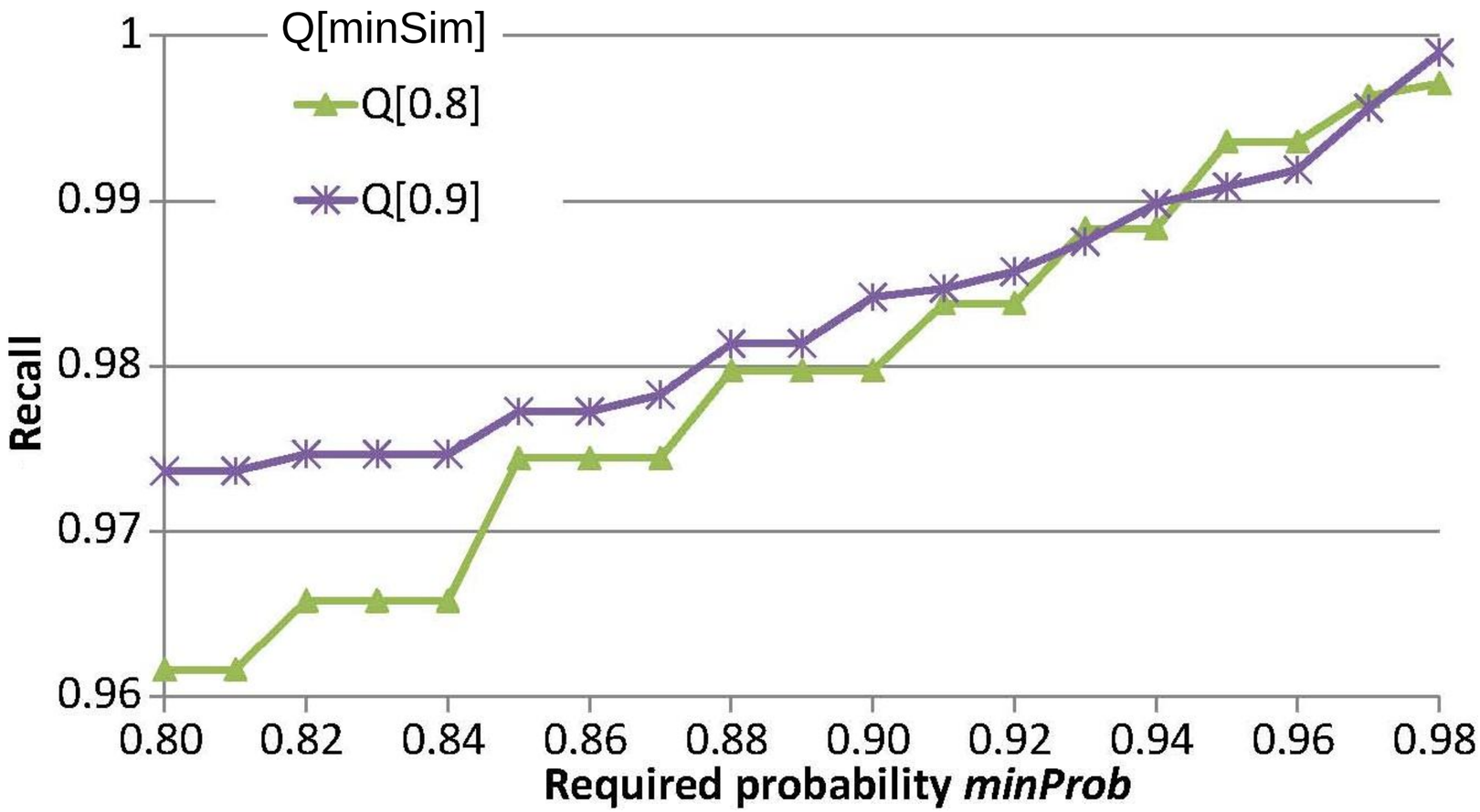
Methodology:

- Detect near duplicate for each articles
- Different required probabilistic guarantees, i.e., minProb
- Two approaches:
  - Searching using the RDFsim approach
  - Detecting near duplicates with pairwise comparison

Experimental Evaluation

Probabilistic guarantees vs. recall:

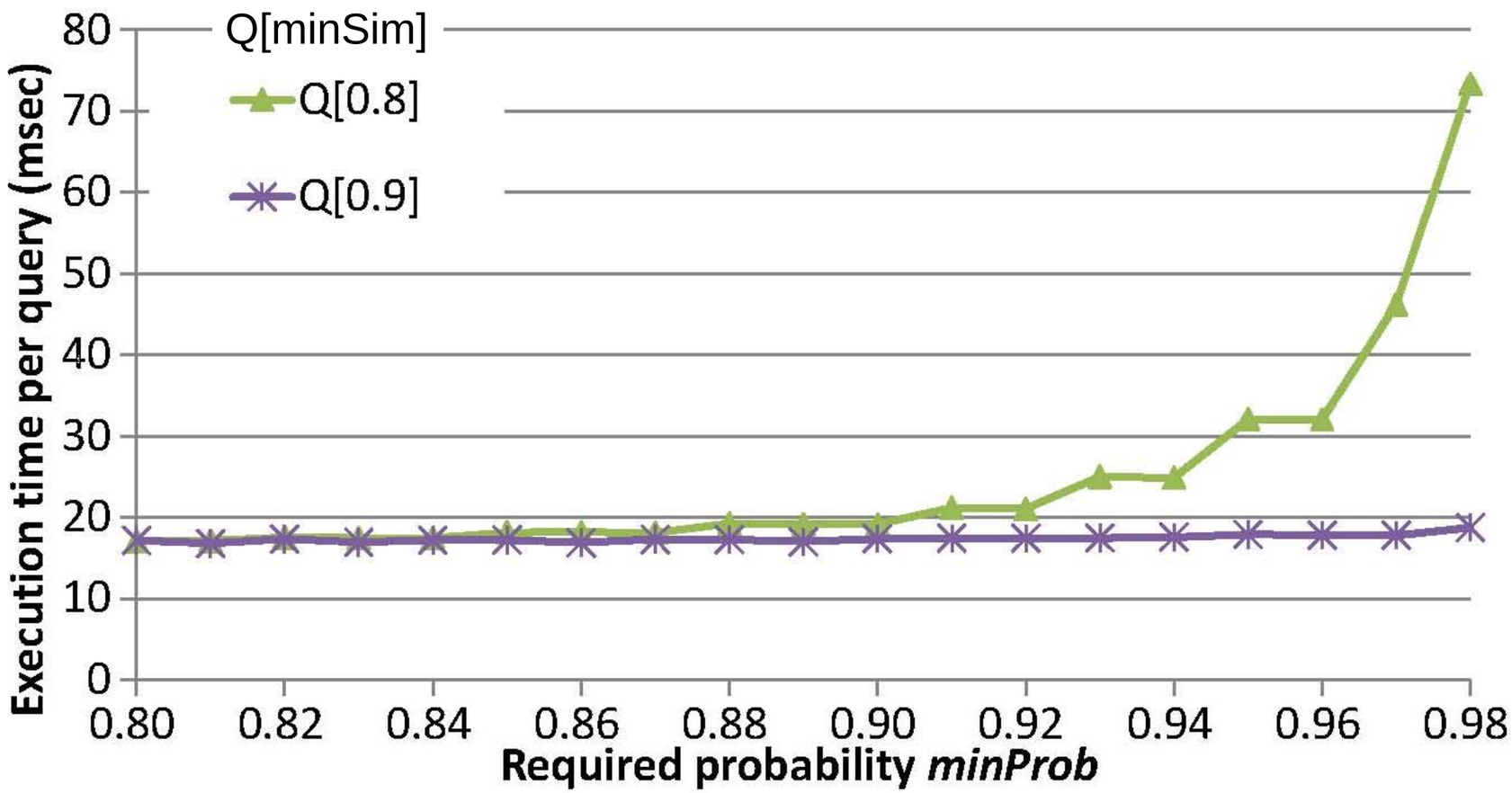
- Recall increases with the required *minProb*
- Recall is always higher than the value of *minProb* (verifies that the probabilistic guarantees are satisfied)



Experimental Evaluation

Probabilistic guarantees vs. average query execution time:

- Small avg execution time for all configurations
- Time increases as the requested *minProb* increases



Outline

1. Motivation
2. RDFsim Approach
   - Resource representation
   - Indexing structure
   - Querying for near duplicate resources
3. Experimental Evaluation
4. **Conclusions**

Conclusions

- Efficiently detect near duplicate resources on the Semantic Web
- Utilize the RDF representations of resources
- Consider semantics and structure of descriptions