Probe, Cluster, and Discover: Focused Extraction of QA-Pagelets from the Deep Web

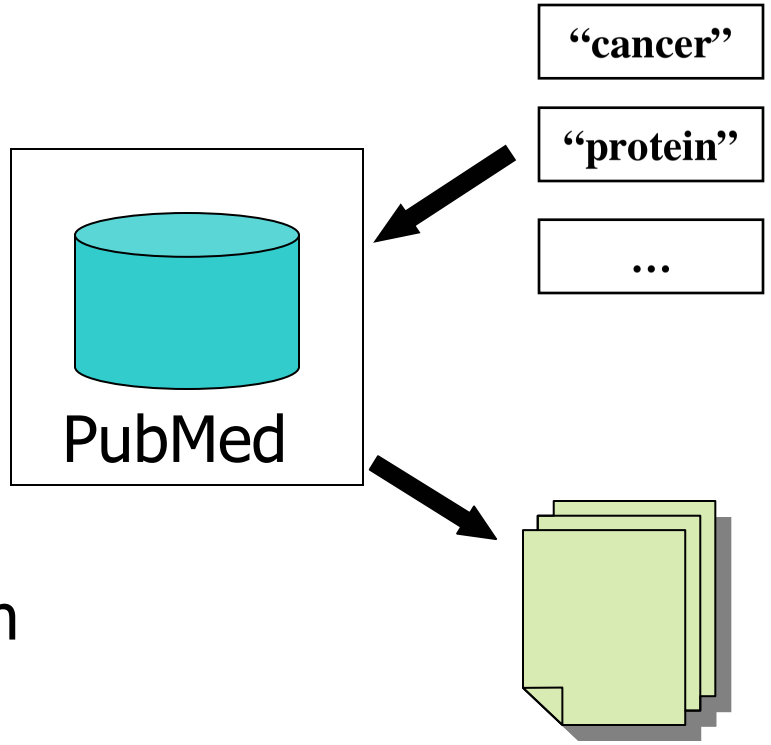
James Caverlee, Ling Liu, and David Buttler
Georgia Institute of Technology
College of Computing

Outline of the talk

- Motivation
  - Why the Deep Web?
  - What are QA-Pagelets?
- System Architecture
- QA-Pagelet Extraction Methods
- Experiments
- Conclusions and Future Work

Our Context: The Deep Web

- Composed of dynamic web pages
  - Typically generated by a user query
  - E.g. Amazon, PubMed, Google, etc.
- Orders of magnitude larger than the so-called static web
  - 500 billion pages vs. 5 billion [2000]
  - Could consider size to be infinite
- Underrepresented (or not represented at all) on most search engines



Opportunity: Making the Deep Web Data Useful

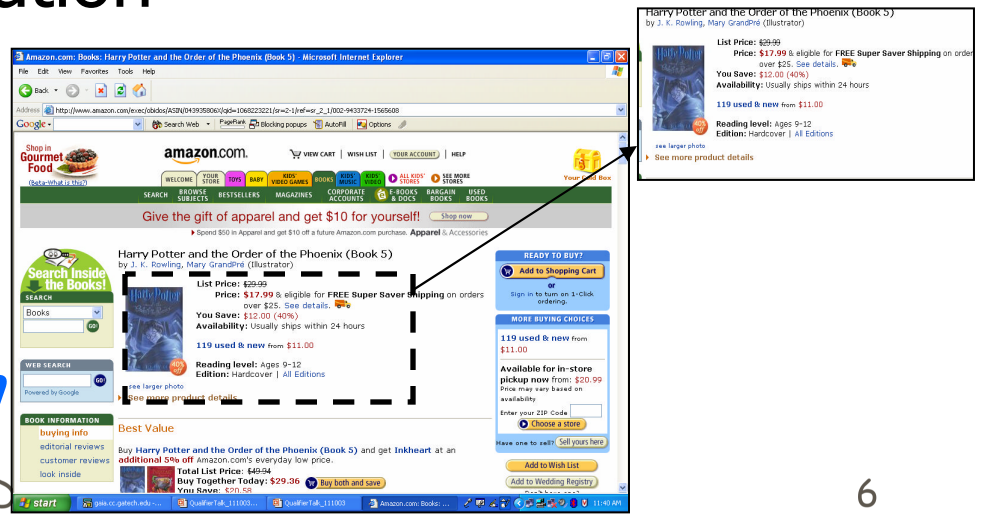
- A number of opportunities exist:
  - Data integration, Data mediation, Composition over multiple sources, Deep Web data source discovery, Change detection, Deep Web search, Etc.
- Main Technical Challenges
  - The number of Deep Web sources continues to grow
  - Most Deep Web data sources are autonomous and heterogeneous
  - There are few search or catalog services for Deep Web data sources

State of the Art:

- Wrapper-based Approaches
  - Manual or Semi-Automated approaches to generate site-specific wrappers
    - XWRAP [Liu+ICDE '00], RoadRunner [Crescenzi VLDB '01], [Adelberg SIGMOD '98], and many others
- Machine-learning based information extraction
  - Manual or semi-automated learning of information extraction rules
    - [Kushmerick IJCAI '97], [Adelberg SIGMOD '98], WHIRL [Cohen AAAI '99], and many others
- Problems:
  - Robustness and Scalability; Vulnerable to changes in text or content structure

THOR - Our Approach

- Key idea: probe, cluster, and discover Deep Web data via multiple steps at increasing levels of granularity:
  - Page identification
  - QA-Pagelet identification
  - Object identification
  - Element extraction



What is a QA-Pagelet?

Query-Related, Information-Rich → QA-Pagelet

Query-Answer Pagelet

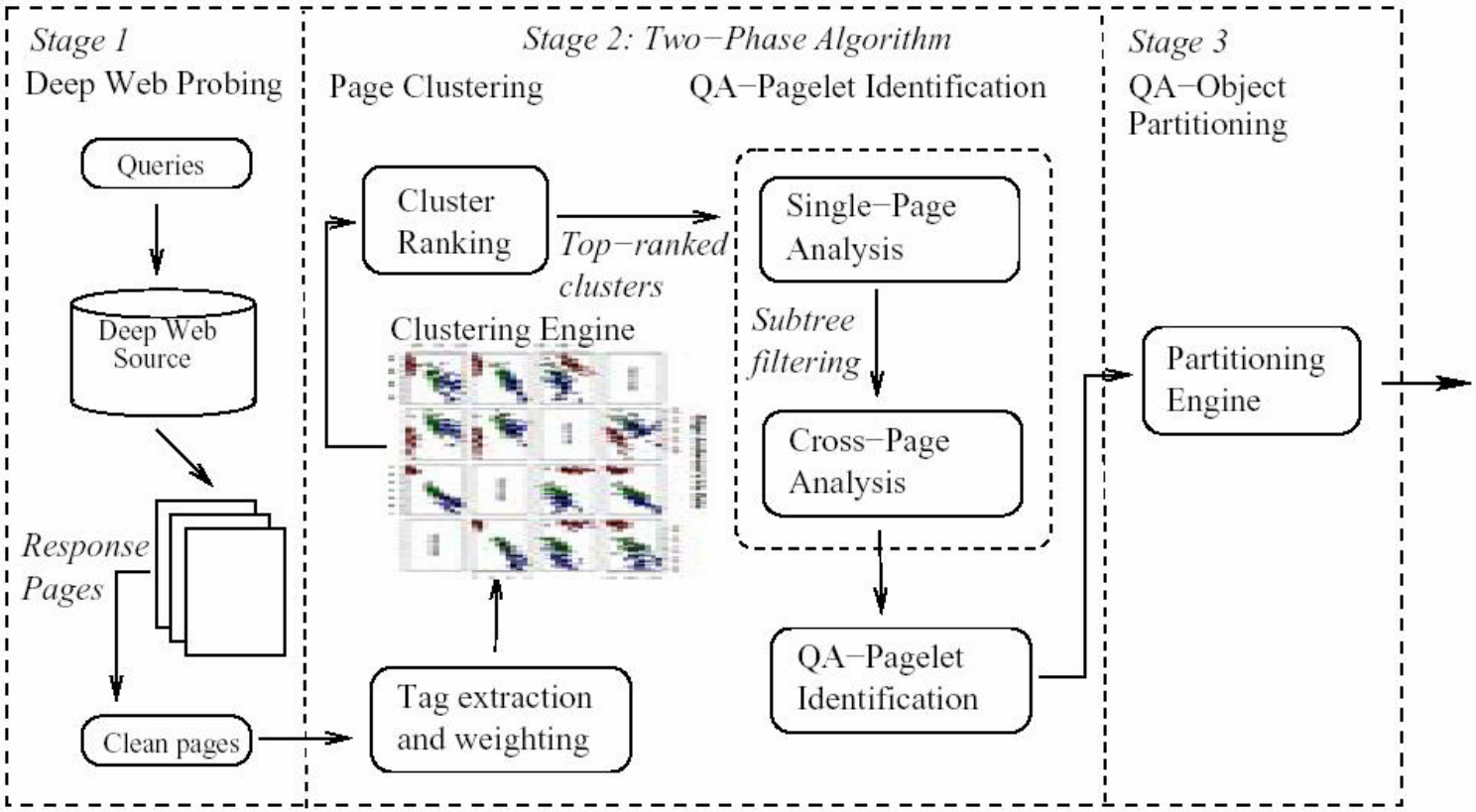
(1) A QA-Pagelet is dynamically-generated in response to a query

(2) It is a page fragment that serves as the primary query-answer content on the page

QA Pagelet Extraction: 3 Stages

- Sample Page Collection by Query Probing
- **Two-Phase QA-Pagelet Extraction**
  - **Identification of interesting pages**
  - **Identification of QA-Pagelets**
- QA-Object Partitioning
  - Identification of object boundaries within each QA-Pagelet

THOR System Architecture



QA-Pagelets: Main Technical Challenges

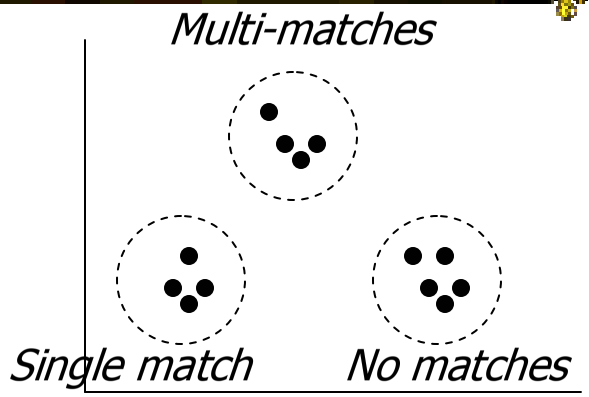
- Problem 1: Each resource may produce many different types of pages.

(a) Multi-matches (b) Single match (c) No matches

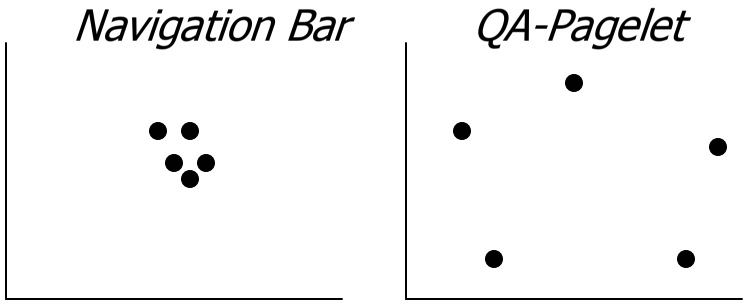
  - Which page types are important?
  - How do we design algorithms independent of page type?
- Problem 2: How can we extract just the query-answer portions?

Two Phase Solution for QA-Pagelet Extraction

- Phase 1: Page Clustering
  - To Group Pages of the Same Type
  - To Rank Promising Clusters

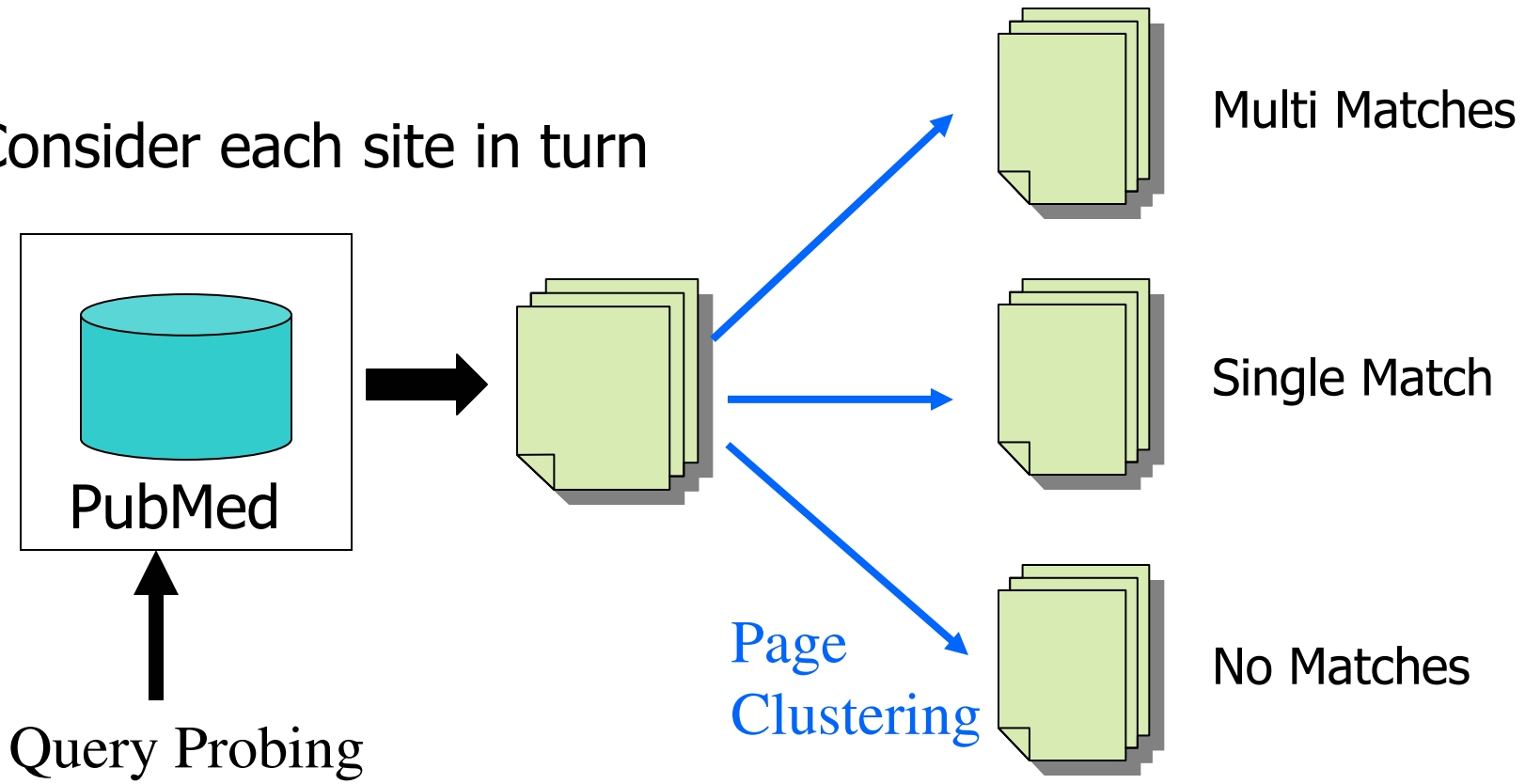


- Phase 2: QA-Pagelet Identification
  - For each page type, identify candidate QA-Pagelets
  - Rank candidates by content clustering (dissimilarity – counterintuitive)



Phase 1: Page Clustering

- Consider each site in turn



The Clustering Problem

- Given a set of *n* pages, $P = \{p_1, p_2, \ldots, p_n\}$, segment these *n* pages into a clustering *C* of *k* clusters: $C = \{Cluster_1, \ldots, Cluster_i, \ldots, Cluster_k \mid \cup Cluster_i = \{p_1, \ldots, p_n\}$ *and* $Cluster_i \cap Cluster_j = \emptyset\}$
- Typical approaches:
  - Random, URLs, Sizes, Raw content, TFIDF content
- Our approach
  - Raw tags, TFIDF tags
- Three fundamental issues:
  - (a) Data model/representation
  - (b) Similarity/Distance metric
  - (c) Clustering algorithm

(a) Model Pages with Tag Signature

- Vector-space representation of page i
- $p_i = \{(tag_1, w_{i1}), (tag_2, w_{i2}), \ldots (tag_N, w_{iN})\}$
  - Where there are N distinct tags
- For example:
  - $p_1 = \{(html, 1), (h1, 1), (br, 4), (p, 3)\}$
  - $p_2 = \{(html, 1), (h1, 1), (table, 1), (tr, 3), (td, 3)\}$
- Orders of magnitude faster than the more complicated tree-edit distance

(b) Measure Similarity with Cosine

$$sim_{Cos}(p_i, p_j) = \left(\frac{\sum_{k=1}^{N} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{N} w_{ik}^2} \cdot \sqrt{\sum_{k=1}^{N} w_{jk}^2}} \right)$$

- Cosine (or normalized inner product) measures the angle between two vectors, independent of vector length
- Ranges from
  - 0 (not similar at all) to
  - 1 (completely similar)

Problem: Simple Tag Weights

- [No Results] p_1 = {(html, 1), (table, 1), (tr, 3), (td, 3)}
- [Single Result] p_2 = {(html, 1), (table, 1), (tr, 3), (td, 3), **(b, 1)**}
- $sim_{Cos}(p_1,p_2) = 0.98$

- TFIDF –
  - Term Frequency – Tags that occur frequently within a page should have *high weight*
  - Inverse Document Frequency – Tags that occur frequently across the space of pages should have *low weight*
- $w_{ik} = \underbrace{\log(tf_{ik} + 1)}_{\textbf{TF}} \underbrace{\log((n+1)/ n_k)}_{\textbf{IDF}}$

(c) Which Clustering Algorithm?

- Simple KMeans
  - (1) Randomly generate *k* cluster centers
  - (2) Assign each page's tag signature to the nearest cluster center
  - (3) Recalculate the cluster center for each of the *k* clusters
  - (4) Goto 2
- Guide cluster formation by random restarts; evaluate each cluster by internal similarity

$$centroid_C = \left\{ \begin{array}{l} (tag_1, \frac{1}{|C|}\sum_{i \in C} w_{i1}) \\ (tag_2, \frac{1}{|C|}\sum_{i \in C} w_{i2}) \\ \cdots \\ (tag_1, \frac{1}{|C|}\sum_{i \in C} w_{iN}) \end{array} \right\}$$

$$Similarity(Cluster_i) = \sum_{p_j \in Cluster_i} sim_{Cos}(p_j, centroid_i)$$

Identifying Interesting Page Clusters – the Ranking Algorithm

- By Several Metrics
  - Average Distinct Terms

$$\frac{1}{|Cluster_i|}\sum_{p\in Cluster_i} distinctTermsCount(p)$$

  - Average Fanout

$$\frac{1}{|Cluster_i|}\sum_{p\in Cluster_i} \max_{u\in p.V}\{fanout(u)\}$$

  - Average Page Size

$$\frac{1}{|Cluster_i|}\sum_{p\in Cluster_c} Size(p)$$

- The Best Phase 1 Clusters are Passed to Phase 2

Phase 2: Identify QA-Pagelets Based on Subtree Analysis

QA-Pagelet corresponds to a subtree

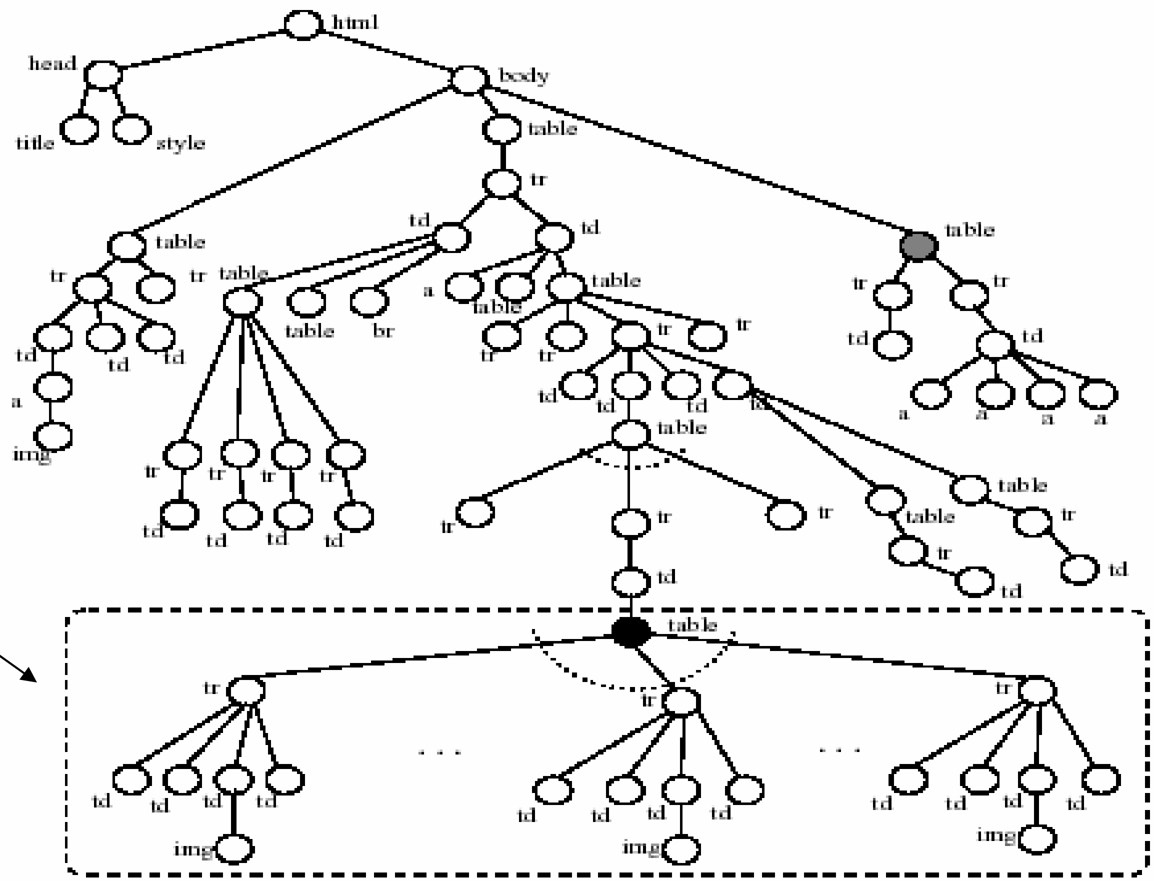
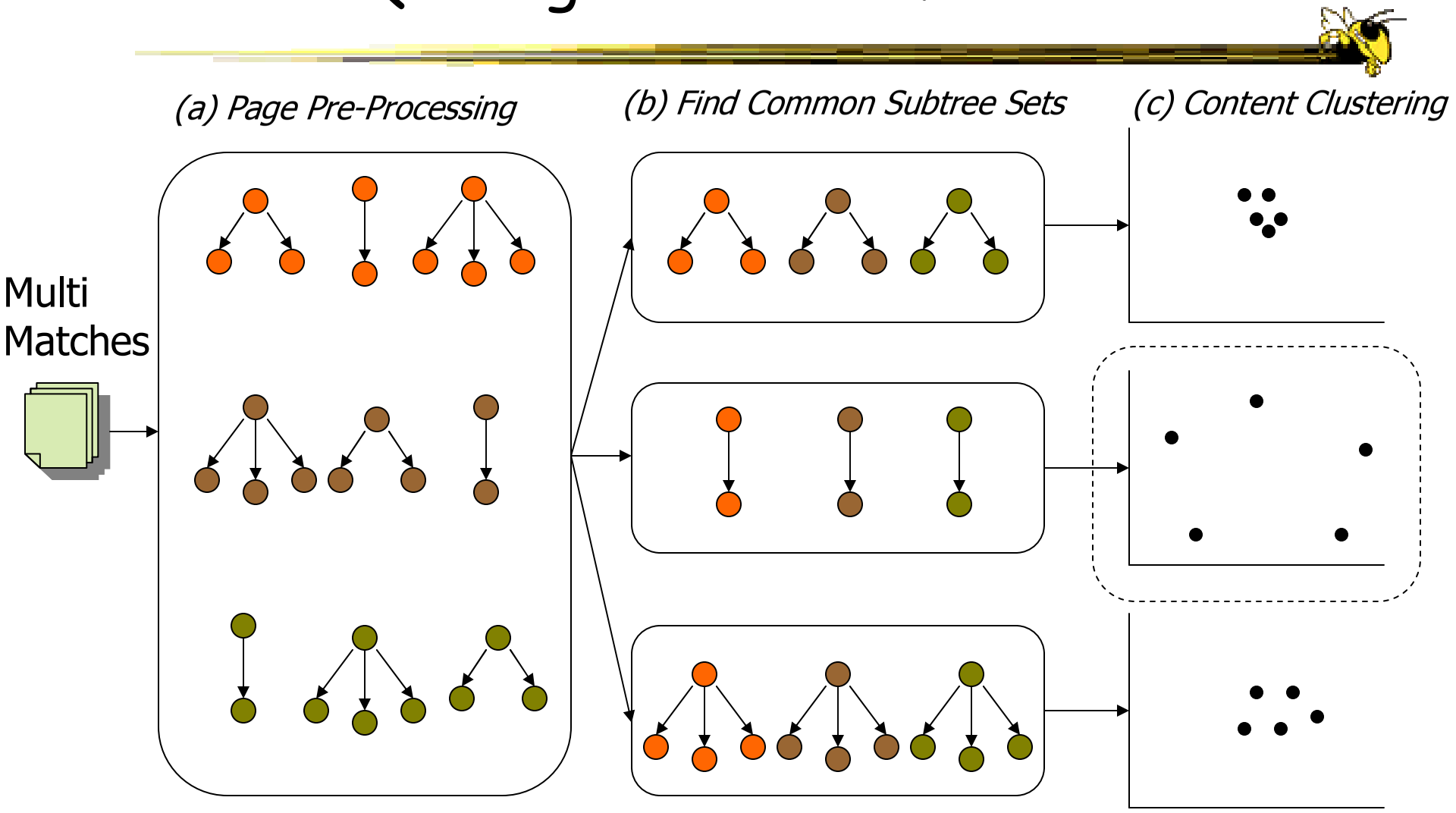


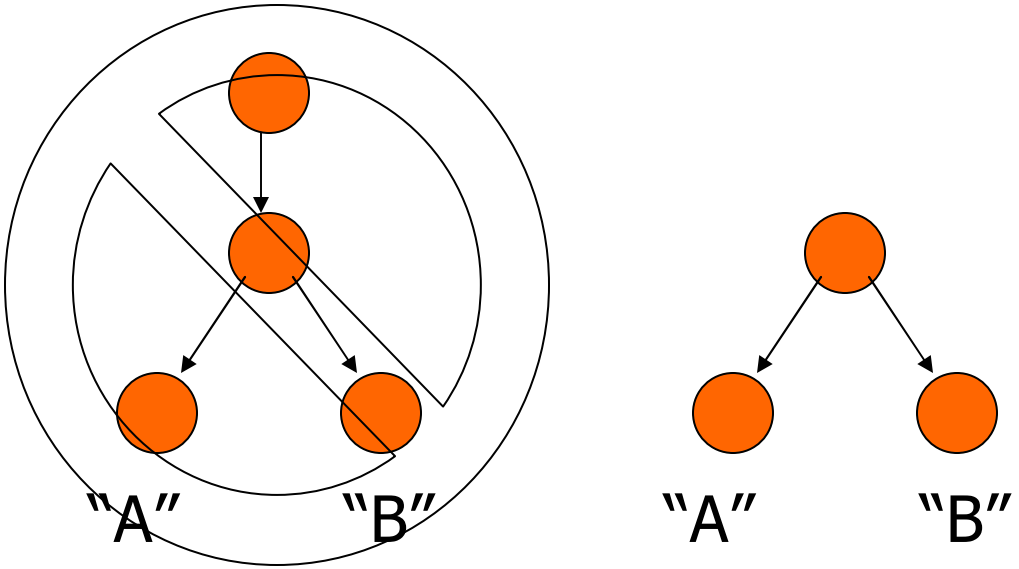
Figure 1: Sample Tag Tree from IBM.com

Phase 2: QA-Pagelet Identification



Step 1: Page Pre-Processing

- Convert pages to subtrees
- Remove content-free subtrees
- Remove non-minimal subtrees



Step 2: Find Common Subtree Sets

- CommonSubtreeSet$_i$ = {subtree$^1_{i,1}$, …, subtree$^{nc}_{i,nc}$}
- Each set should correspond to a functional fragment of the page, e.g.:
  - The navigational bar
  - The QA-Pagelet
  - And so on
- Using structural characteristics:

$$distance(subtree_i, subtree_j) = w_1 \frac{EditDist(P_i, P_j)}{max(len(P_i), len(P_j))}$$
$$+ w_2 \frac{|F_i - F_j|}{max(F_i, F_j)} + w_3 \frac{|D_i - D_j|}{max(D_i, D_j)} + w_4 \frac{|N_i - N_j|}{max(N_i, N_j)}$$

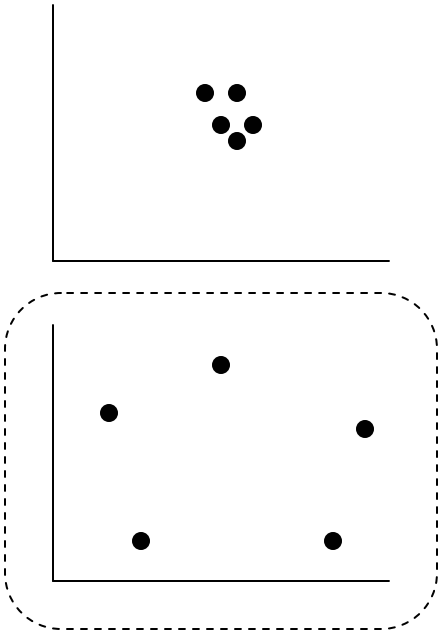
Step 3: Content Clustering

- Examine cross-page subtree content to find "interesting" common subtrees
- Vector-space representation of *the content* of each $subtree_{ij}$ (the *ith* subtree in the *jth* common subtree set)
- $subtree_{ij} = \{(term_1, w_{i1}), (term_2, w_{i2}), \ldots (term_{Nj}, w_{iNj})\}$
  - Where there are N_j distinct terms in the jth common subtree set
- $w_{iq} = \log(tf_{iq} + 1) \log((n_j+1)/ n_{qj})$

Step 3: Content Clustering (cont'd)

- Rank the common subtree sets
  - According to the average intra-subtree set similarity

$$IntraSubtreeSetSim_i = \sum_{j=1}^{n} \sum_{l \neq j}^{n} sim(subtree_j, subtree_l)$$



- Intuition
  - Static subtrees should be very similar
  - Dynamic (QA-Pagelets) should be very dissimilar

Experiments and Evaluation

- Objectives:
  - Evaluating the Page Clusters
    - Entropy + Time
  - Evaluating QA-Pagelet Extraction
    - Distance Measure and TFIDF Weighting
  - Evaluating the Two-Phase Approach
    - Precision/Recall

Experimental Setup

- Web data:
  - Randomly selected 50 sources from a crawl that had identified 3,000 unique search forms
  - Probed each source with 100 random terms from the Unix dictionary and 10 nonsense words for 5,500 total pages
- Synthetic data:
  - 5.5 million pages generated by randomly generating tag and content signatures for each synthetic page based on the overall distribution of Web data

Evaluating the Page Clusters

- Entropy:

$$Entropy(Cluster_i) = \frac{-1}{\log(c)} \sum_{j=1}^{c} (p(z) \log p(z))$$

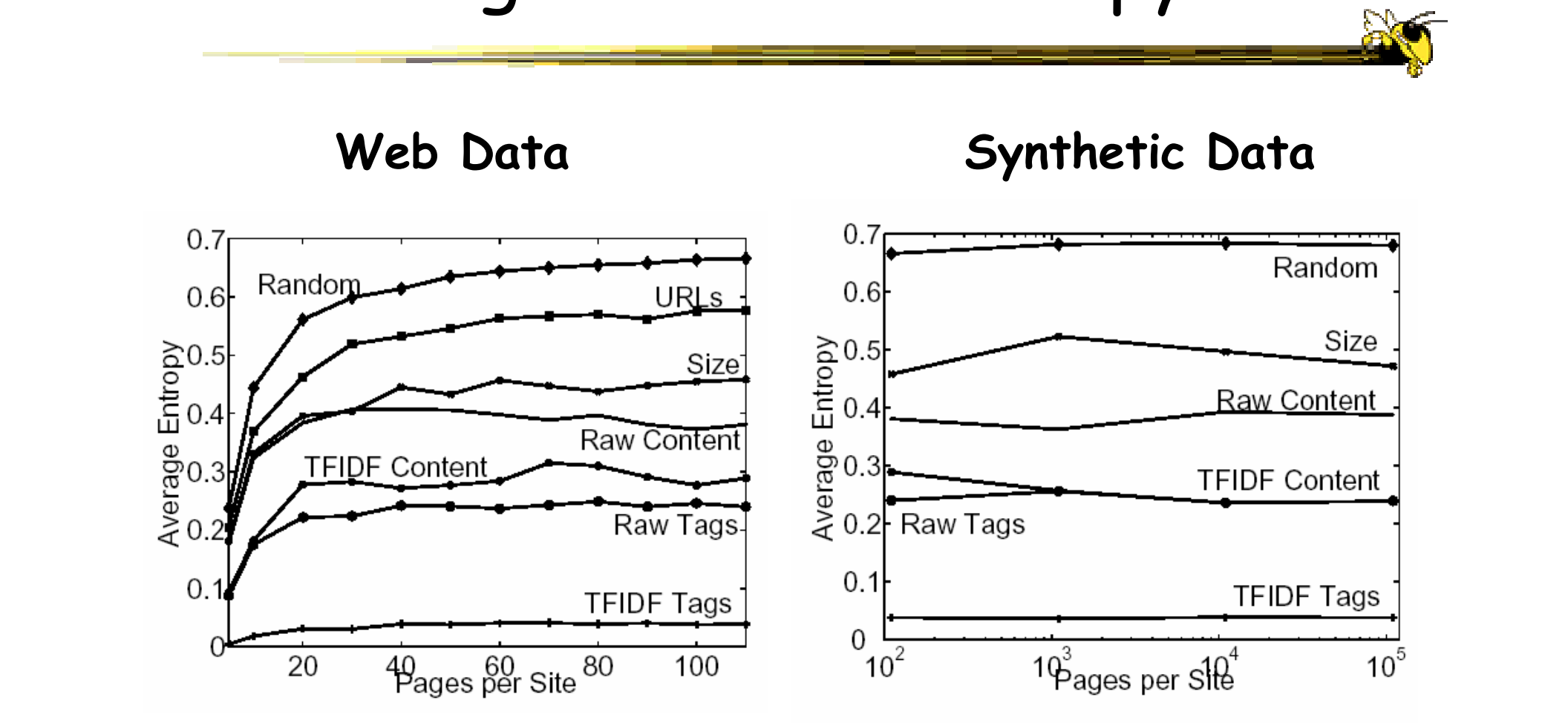
- Where

  p(z) = Prob(page p belongs to Class(j) | page p is in cluster(i))

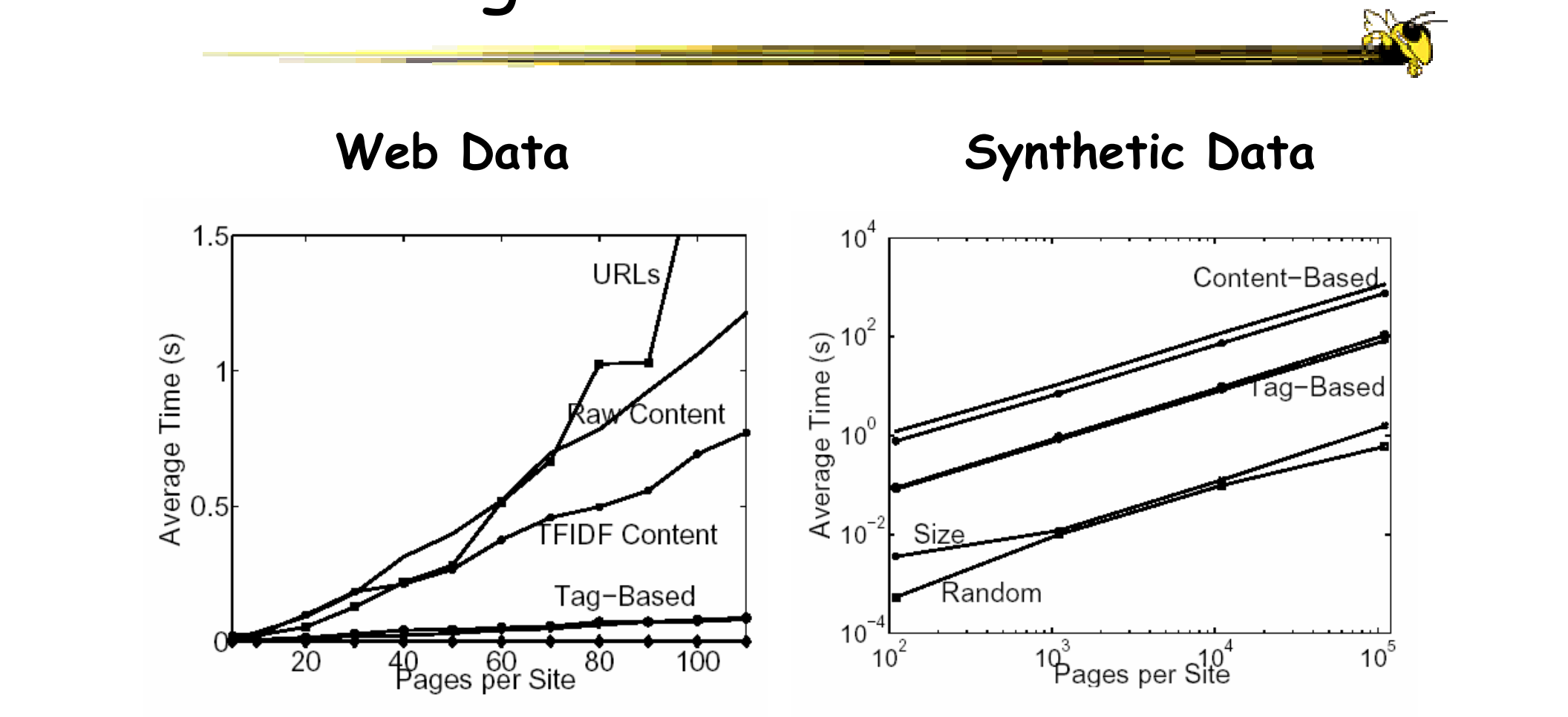
- Estimated as:

  p(z) = number of pages in Cluster(i) that belong to Class(j) / number of pages in Cluster (i)

Clustering Results – Entropy



Clustering Results – Time



Precision and Recall Metrics

- Precision

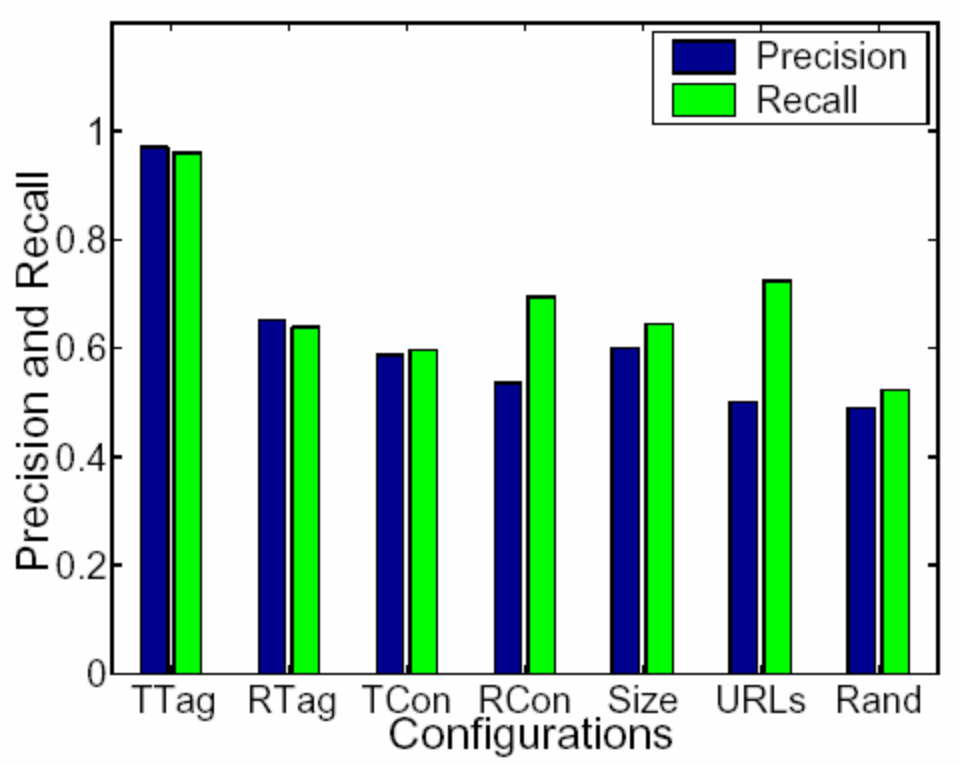
  = Number of QA-Pagelets Correctly Identified / Number of Subtrees Identified as QA-Pagelets

- Recall

  = Number of QA-Pagelets Correctly Identified / Total Number of QA-Pagelets in the Set of Pages

Overall Precision/Recall

- Phase 2 success based on Phase 1 success



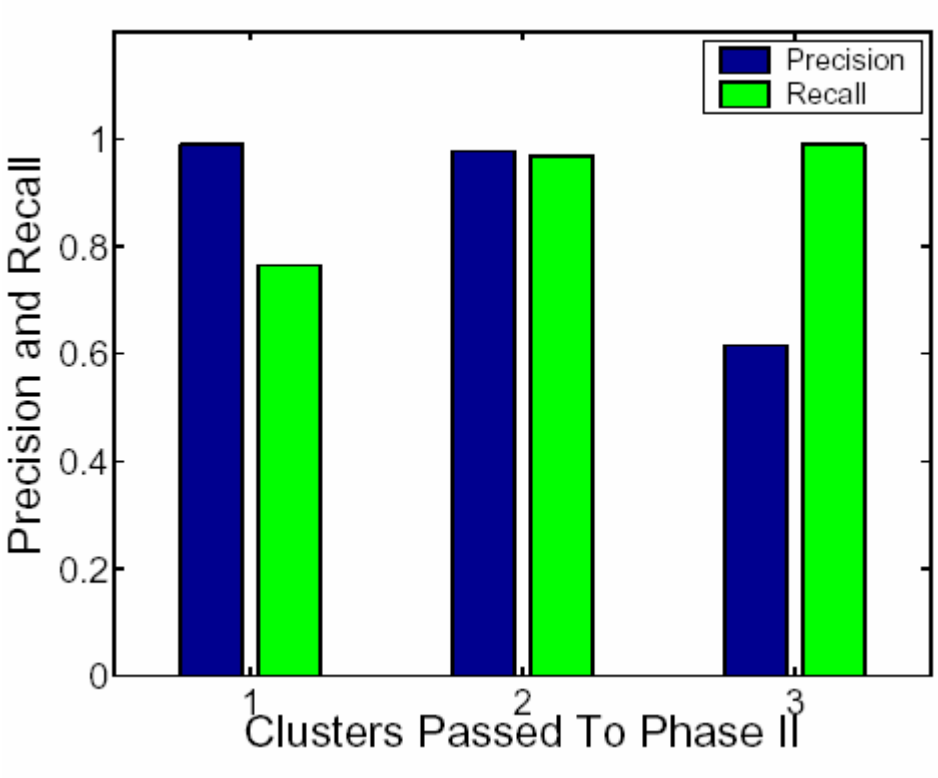
Double impact:

(1) if a normal results page is misclustered into a no-results cluster, it won't advance to phase 2;

(2) any no-results pages that do advance will adversely affect phase 2 performance

Precision/Recall Tradeoff

- Based on Number of Clusters Passed to Phase 2



With one cluster, precision is high, but recall is low

With three clusters, precision is low, but recall is high

Emphasizes necessity of optimizing the number of page clusters to analyze

Conclusion and Future Work

- Summary of the main contributions
  - THOR – Scalable and efficient system for discovering and extracting QA-Pagelets
  - Two-phase extraction approach:
    - Cluster pages by structural similarity
    - Subtree cluster analysis for QA-Pagelet extraction
- Future work
  - Incorporating THOR clustering algorithms into XWRAP systems or other extraction techniques

Questions?