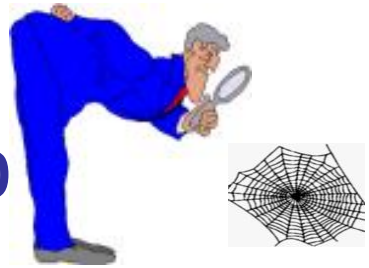
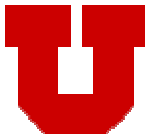
Searching for Hidden-Web Databases



Luciano Barbosa Juliana Freire

University of Utah



The Hidden Web

- Web content *hidden* behind form interfaces
  - Not accessible through traditional search engines
- Databases on the Web
  - Lots of data (7,500--91,000 TB)
  - High-quality content: flight schedules, library catalogs, sensor readings, patent filings, genetic research data, ...
- Lots of *academic and commercial* interest in accessing and integrating hidden-Web data
  - MetaQuerier (UIUC), Metasearch (UIC), Tatu (U. Utah), ...
  - Yahoo!, Transformic, Dipsie, CompletePlanet...
- Issues considered: Crawling, form clustering, form matching
- Problem: How to find hidden-Web databases?

Searching for Hidden-Web Databases

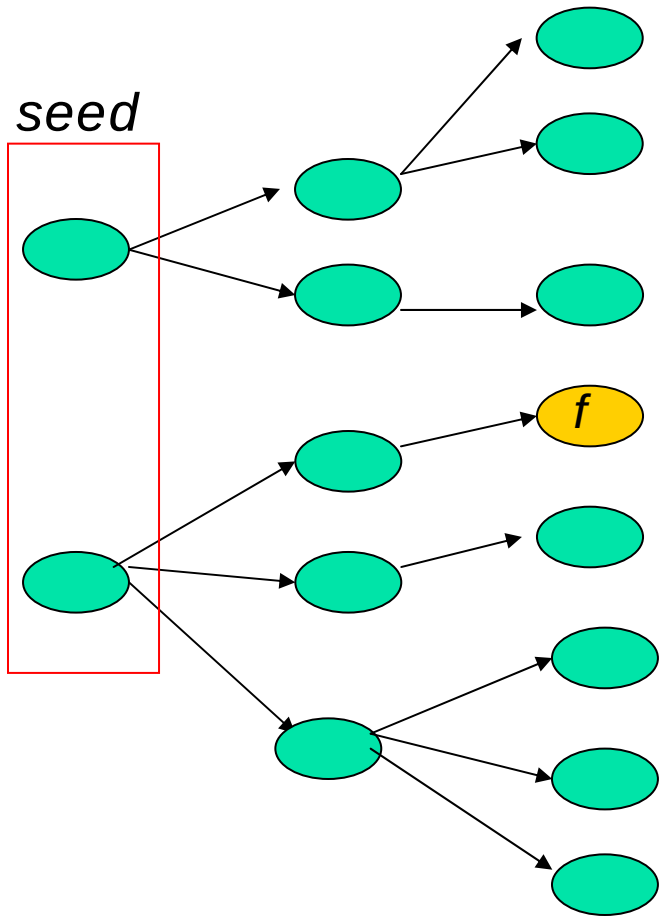
- Find and identify the entry points to the hidden Web
  - Given the set of all Web pages W, find $W_f \subset W$ st W_f contains searchable forms
- Very sparse space: the Web is huge, there are relatively few databases -- $|W_f| << |W|$
  - Google indexes 8 billion pages
  - ~300,000 hidden databases [Chang et al, 2004]

Look for a few needles in a haystack

- Forms not precisely defined
- Requirements
  - Perform a broad search
  - Avoid visiting pages unnecessarily

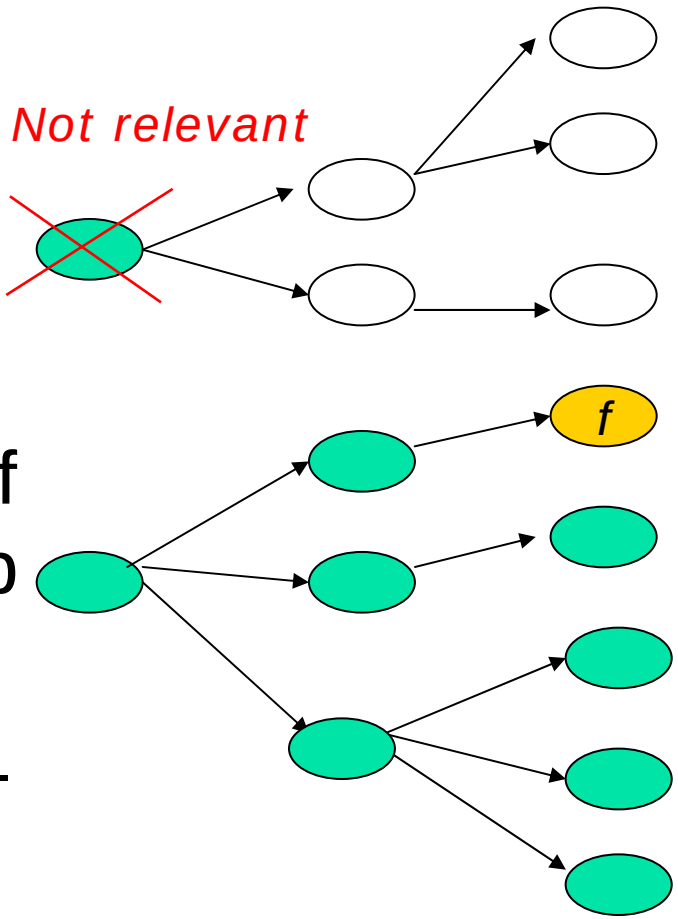
Traditional Crawlers

- Start from a *seed* set of urls
- Recursively follow links to other documents
- Problems:
  - Too many documents must be retrieved before hitting the target
  - Inefficient: it takes too long to crawl the *whole* Web



Focused Crawlers

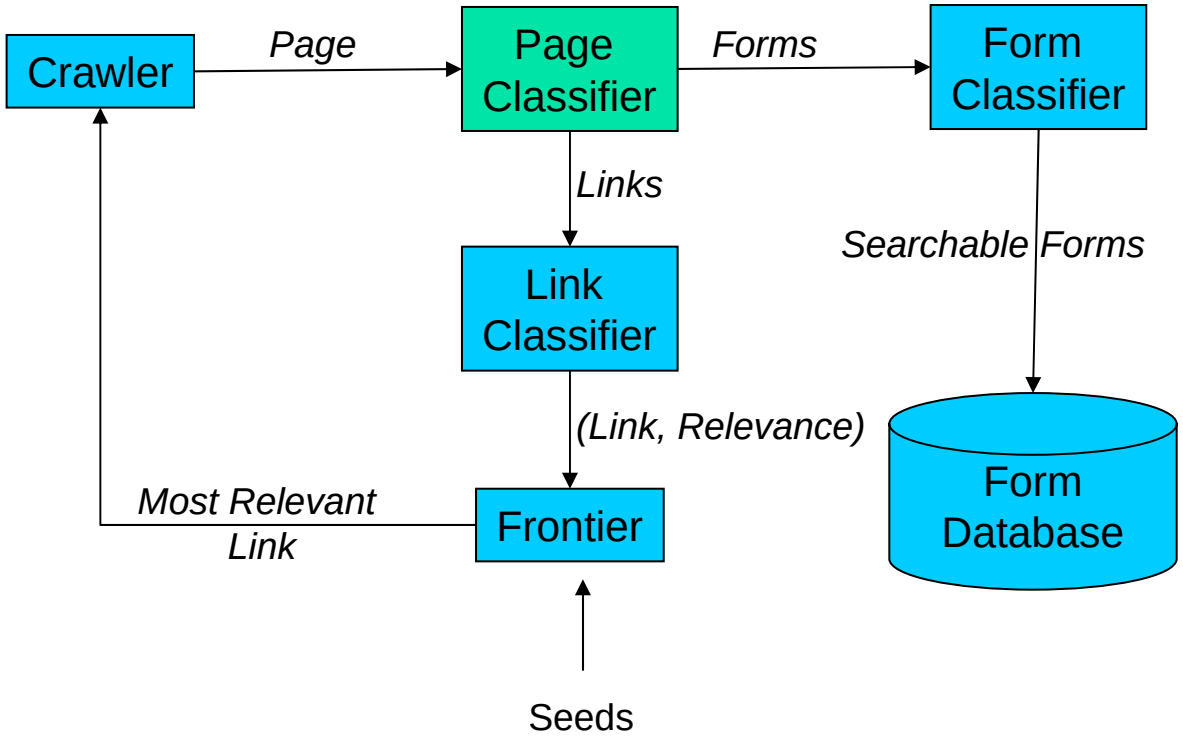
- Search and retrieve only subset of Web that pertains to a specific topic of relevance
  - consider the contents
- Retrieves a small subset of the documents on the Web
- Problem: still inefficient
  - Too many pages retrieved -- few forms even within a restricted domain



Form-Focused Crawler

- Focus on topic – just like a focused crawler
- Also focus on finding forms: better guide the search
- Goal: ensure a high *harvest rate* – fraction of pages fetched that contain forms

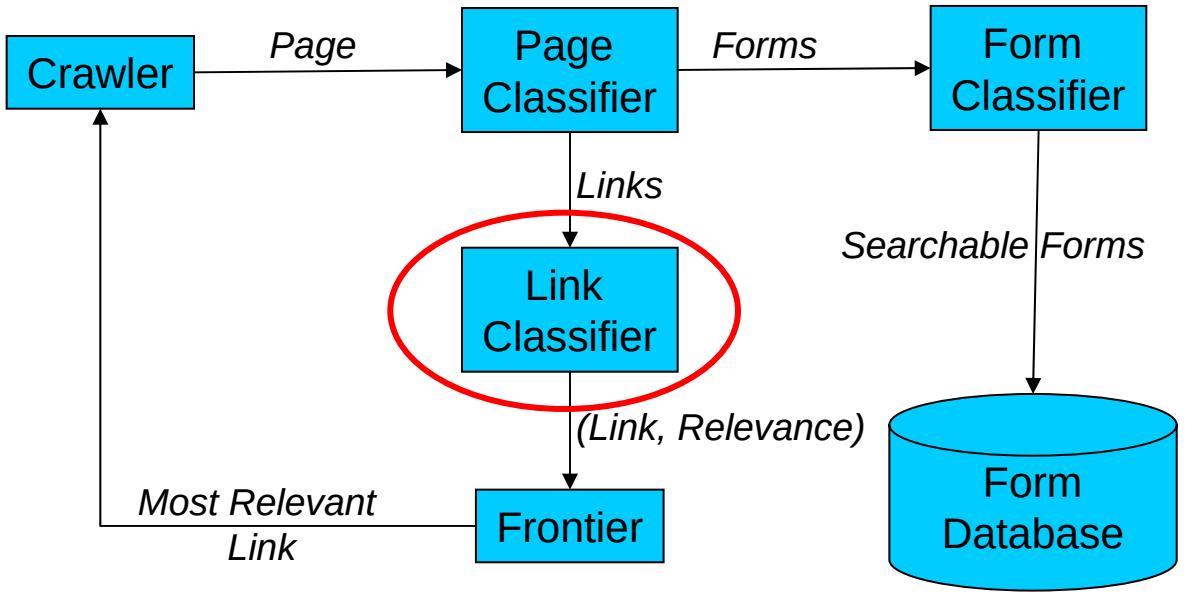
Form-Focused Crawler: Overview



Page Classifier

- Goal: identify pages that belong to a given topic – keep the crawler on a specific topic
- Similar to the best-first focused crawler [Chakrabarti et al, 1999]
- Implementation: used Rainbow to build as Naïve Bayes classifier

Link Classifier



Link Classifier

- Goal: further focus the crawl – prioritize links that are likely to lead (or that are *close*) to pages P_f containing forms
- Problem: Learn patterns of links that lead to P_f in *1 or more steps*
  - Estimate how far is a link *l* from P_f
- Building the classifier:
  - Repeatedly crawl sample sites to learn features of good paths [Rennie and McCallum, 1999]
  - May require crawling a substantial portion of the sites to build a *good* sample of the Web graph:
    - Ok for well-defined search problems
    - Too inefficient for a *broad* search

Learning by Crawling Backwards

- Build a sample of the paths by crawling backwards
  - Find a representative set of pages P_0 that contain searchable forms
  - Use Google or AltaVista “link:” facility

  P_1 = link(P_0); P_2 = link(P_1); ...
- Extract features from links in

level n (P_n, P_{n-1}), level n-1 (P_{n-1}, P_{n-2}), ...,

level 2 (P_2 ,P_1) , level 1 (P_1 ,P_0)

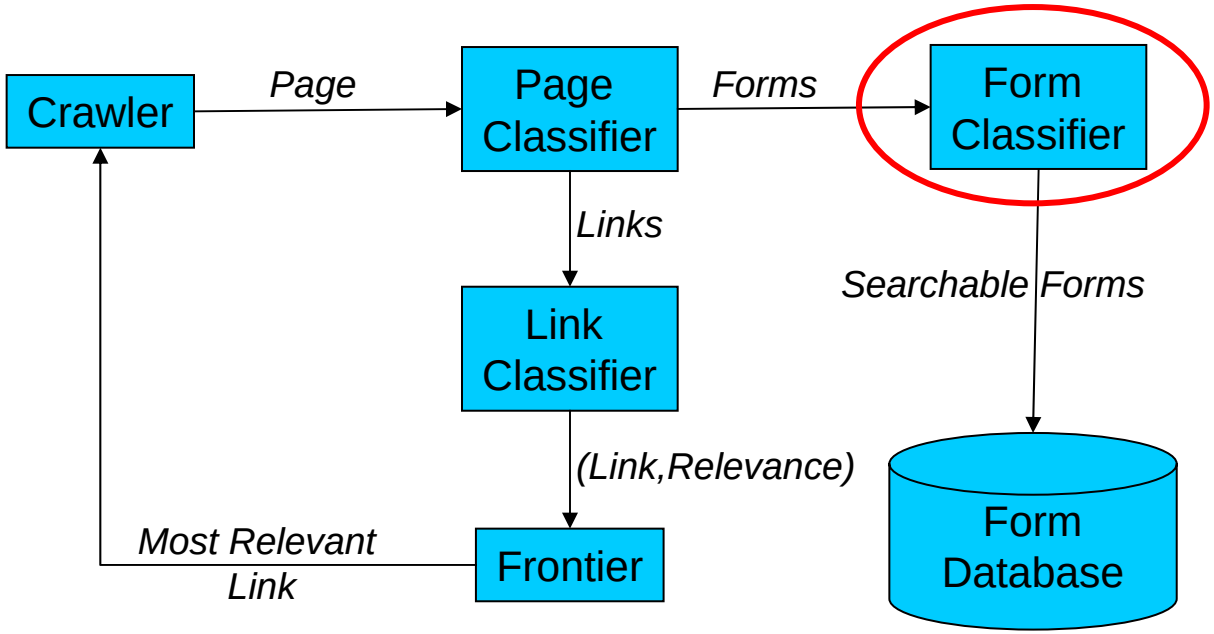
  - Contexts: anchor, URL, text in the proximity of the URL
- Build a Naïve Bayes classifier
  - Obtain the probabilistic class membership for links in a level

Feature Space for Jobs

level/field	URL	Anchor	Around the link	Title of page	Text of page	Number of pages
1	**job 111** **search 38** **career 30** opm 10 htdocs 10 roberthalf 10 accountemps 10	**job 39** **search 22** ent 13 **advanced 12** **career 7** width 6 popup 6	**job 66** **search 49** **career 38** **work 25** home 16 keyword 16 help 15	job 77 career 39 work 25 search 23 staffing 15 results 14 accounting 13	job 186 search 71 service 42 new 40 career 35 work 34 site 27	187
2	**job 40** classified 29 news 18 annual 16 links 13 topics 12 default 12 ivillage 12	**job 30** **career 14** today 10 ticket 10 corporate 10 big 8 list 8 find 6	**job 33** home 20 ticket 20 **career 18** program 16 sales 11 sports 11 search 11	job 46 career 28 employment 16 find 13 work 13 search 13 merchandise 13 los 10	job 103 search 57 new 36 career 35 home 32 site 32 resume 26 service 22	212
3	ivillage 18 cosmopolitan 17 ctnow 14 state 10 archive 10 hc-advertise 10 **job 9** poac 9	**job 11** advertise 8 web 5 oak 5 fight 5 **career 5** against 5 military 5	**job 21** new 17 online 11 **career 11** contact 10 web 9 real 9 home 9	job 17 ctnow 8 service 8 links 7 county 7 career 7 employment 7 work 6	font 37 job 33 service 24 cosmo 20 new 19 career 19 color 16 search 16	137

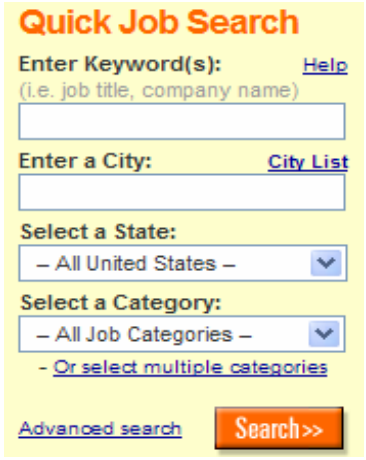
- Words related to topic domain and “search” in all features
- Frequency of *relevant* words decreases as the distance from target page increases, but many relevant words are present in lower levels
- Important to combine focused crawler with link classifier

Form Classifier



Form Classifier

- Not all forms are entry points to hidden-Web databases
  - E.g., login, discussion groups, mailing list subscriptions
- Goal: Identify searchable forms



Searchable form

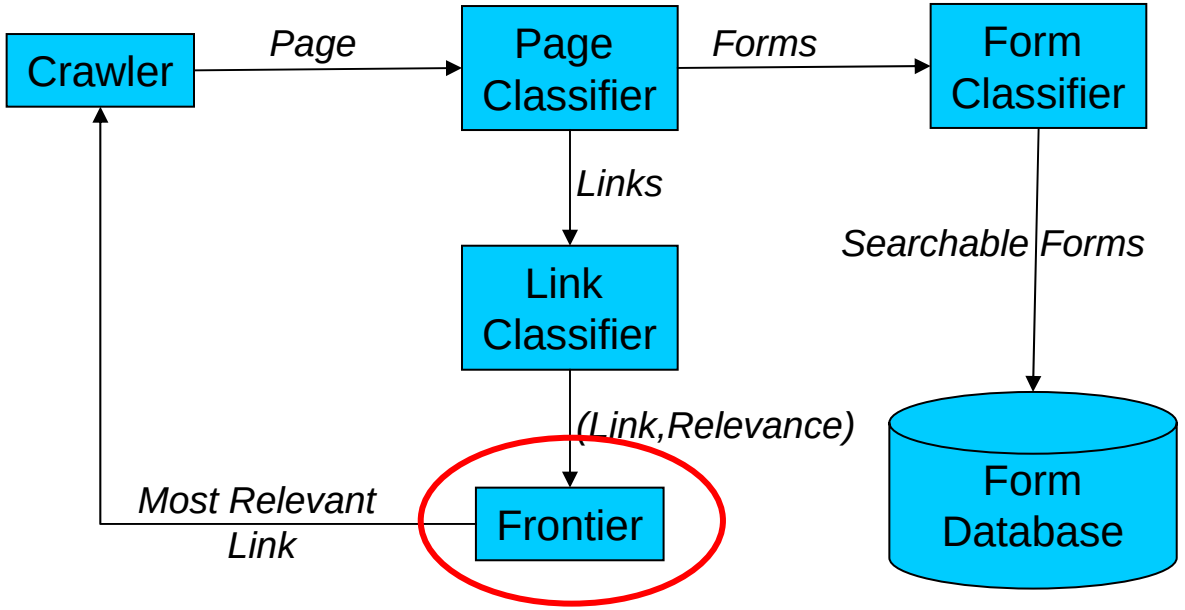
Non-searchable form

Form Classifier

- Positive examples: UIUC repository
- Negative examples: manually collected
- 14 features: number of checkboxes, textboxes etc.
- Tried different classifiers
- Chosen: C4.5 – lowest test error rate

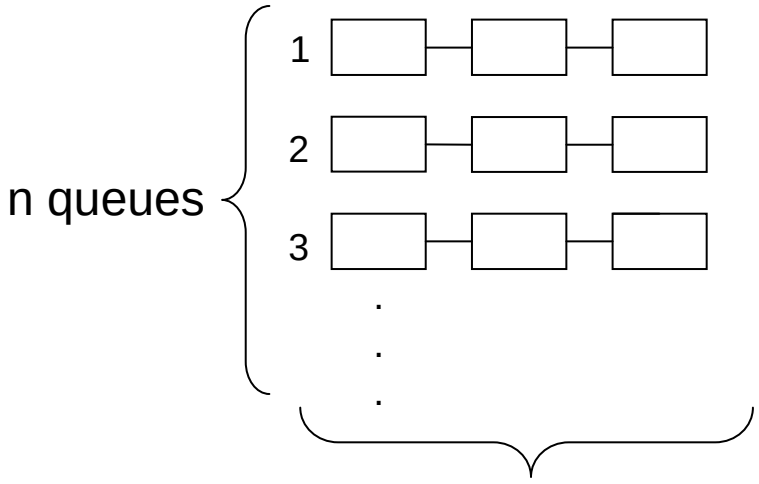
Algorithm	Error test rate
C4.5	8.02%
Support Vector Machine	14.19%
Naive Bayes	10.49%
Multiayer Perceptron	9.87%

Frontier


Crawler
Page
Page Classifier
Forms
Form Classifier
Links
Searchable Forms
Link Classifier
(Link,Relevance)
Form Database
Most Relevant Link
Frontier

Managing the Frontier

- Frontier
  - Links to be visited ordered by *importance*
  - Determines the crawler's steps
- Link classifier sets the priority
  - One queue per level
- Root pages have the highest priority within a queue
  - Searchable forms are close to the root page [Chang et al, 2004]



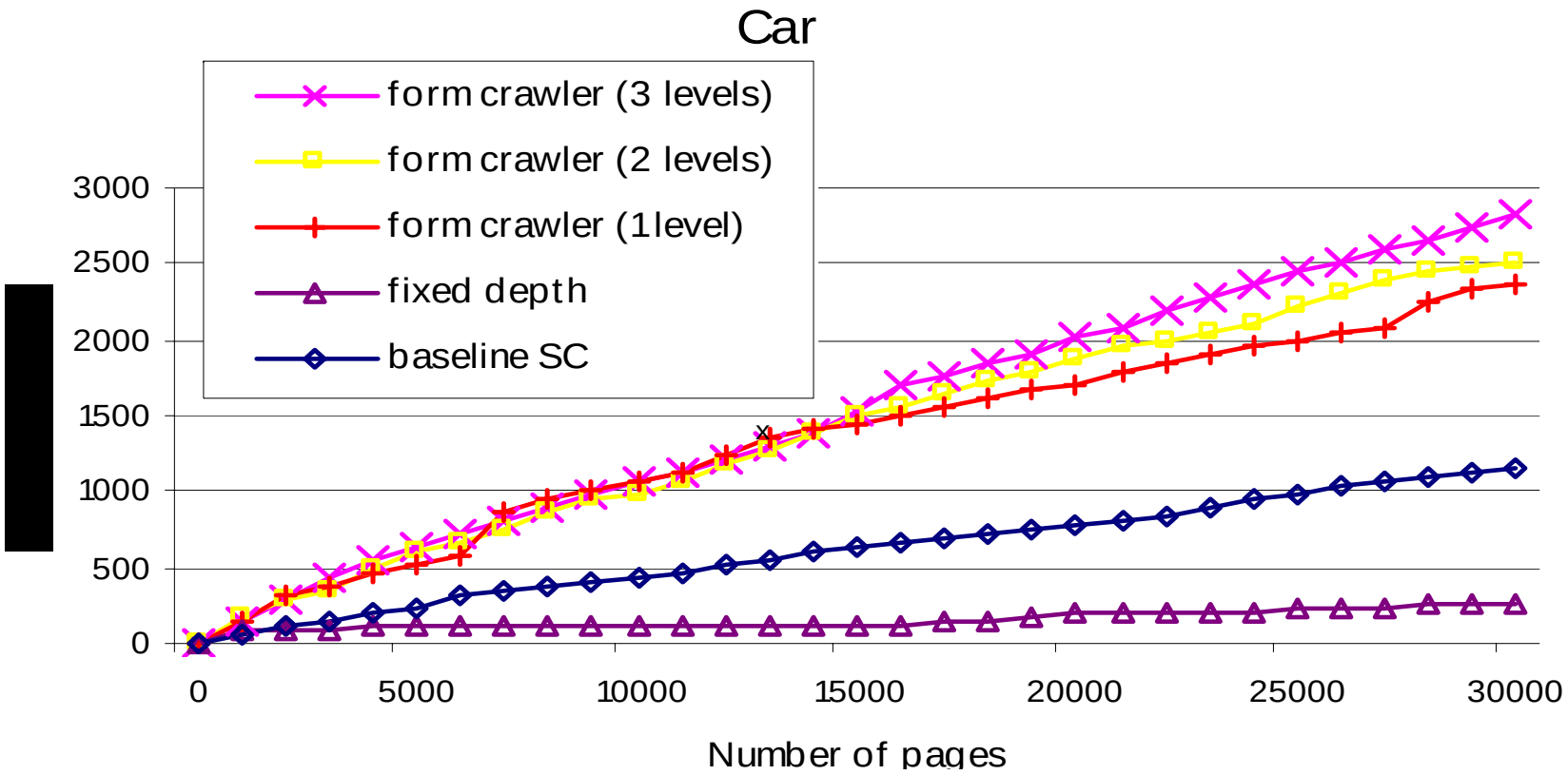
Stopping Criteria

- Some sites may not have searchable forms
- Sites have very few searchable forms
  - Average of 4.2 query interfaces per deep-Web site [Chang et al, 2004]
- Avoid unnecessarily crawling a given site for too long
- Stop crawling a site S if
  - Enough *distinct* forms are retrieved, or
  - A maximum number of pages is visited in S

Experiments

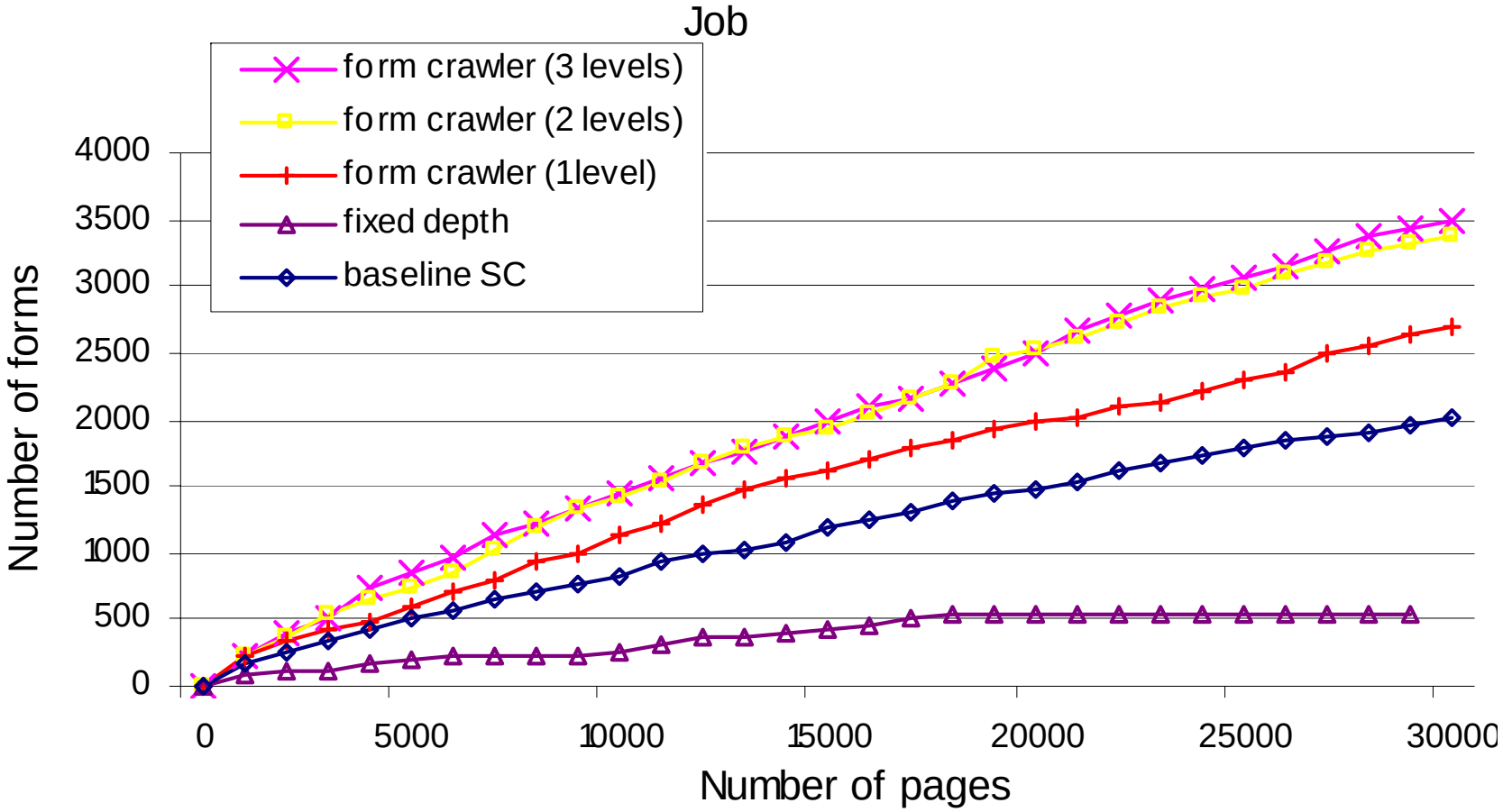
- Crawlers:
  - Baseline: Variation of the best-first crawler [Chakrabarti et al, 1999]
  - Fixed depth [Chang et al, 2005]
  - Baseline SC: Baseline + stop criteria
  - Form Crawler with 1-3 levels
  - *Form Crawler with 1-3 levels without prioritizing root pages*
- Measure the number of *distinct* relevant forms in relation to number pages visited
- Domains: cars, jobs, books

Results: Cars

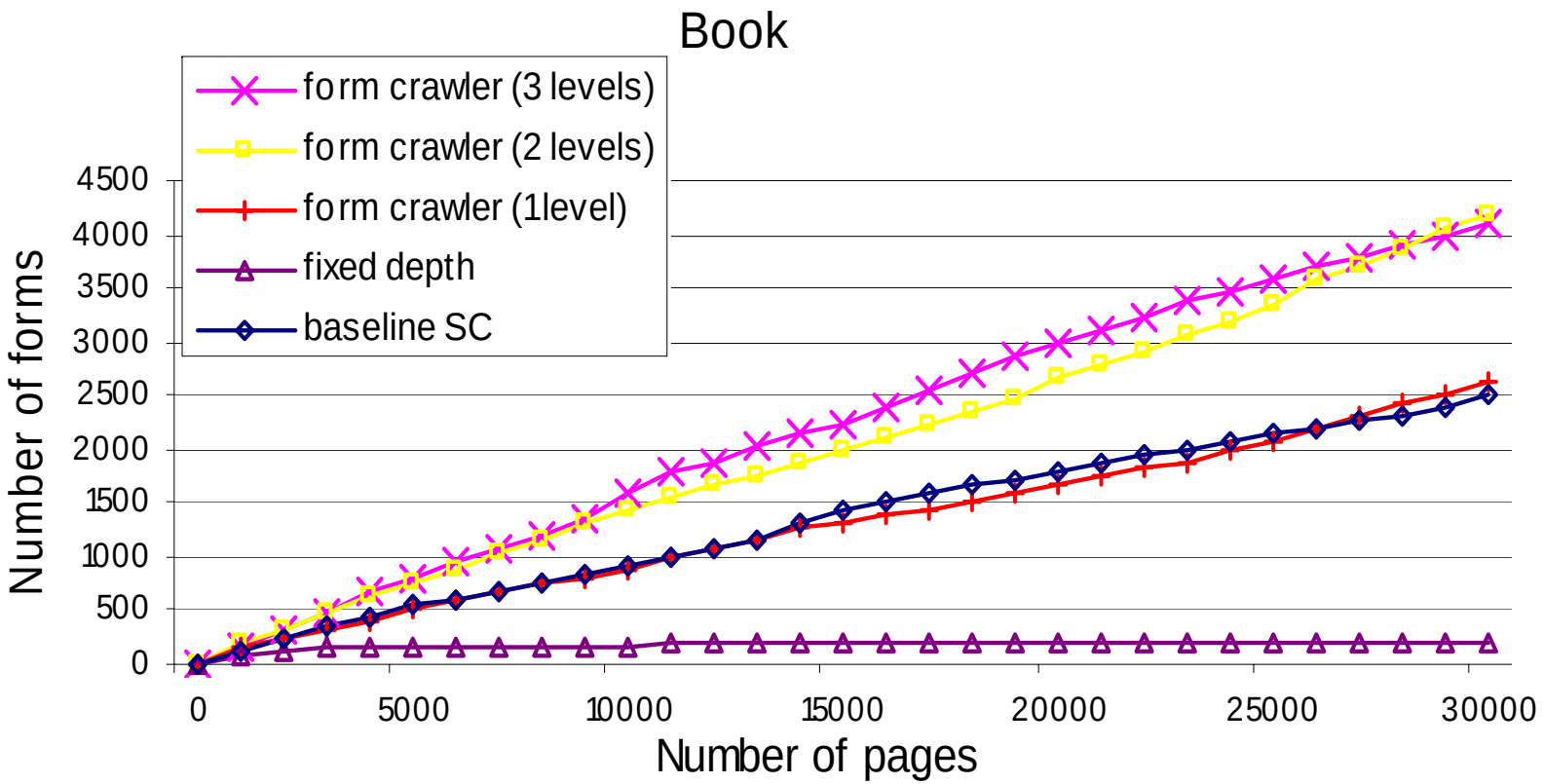


- Baseline SC is much better than baseline: 1,168 forms after crawling 30,000 pages
- Multiple levels are better than baseline SC: 2,883 forms retrieved

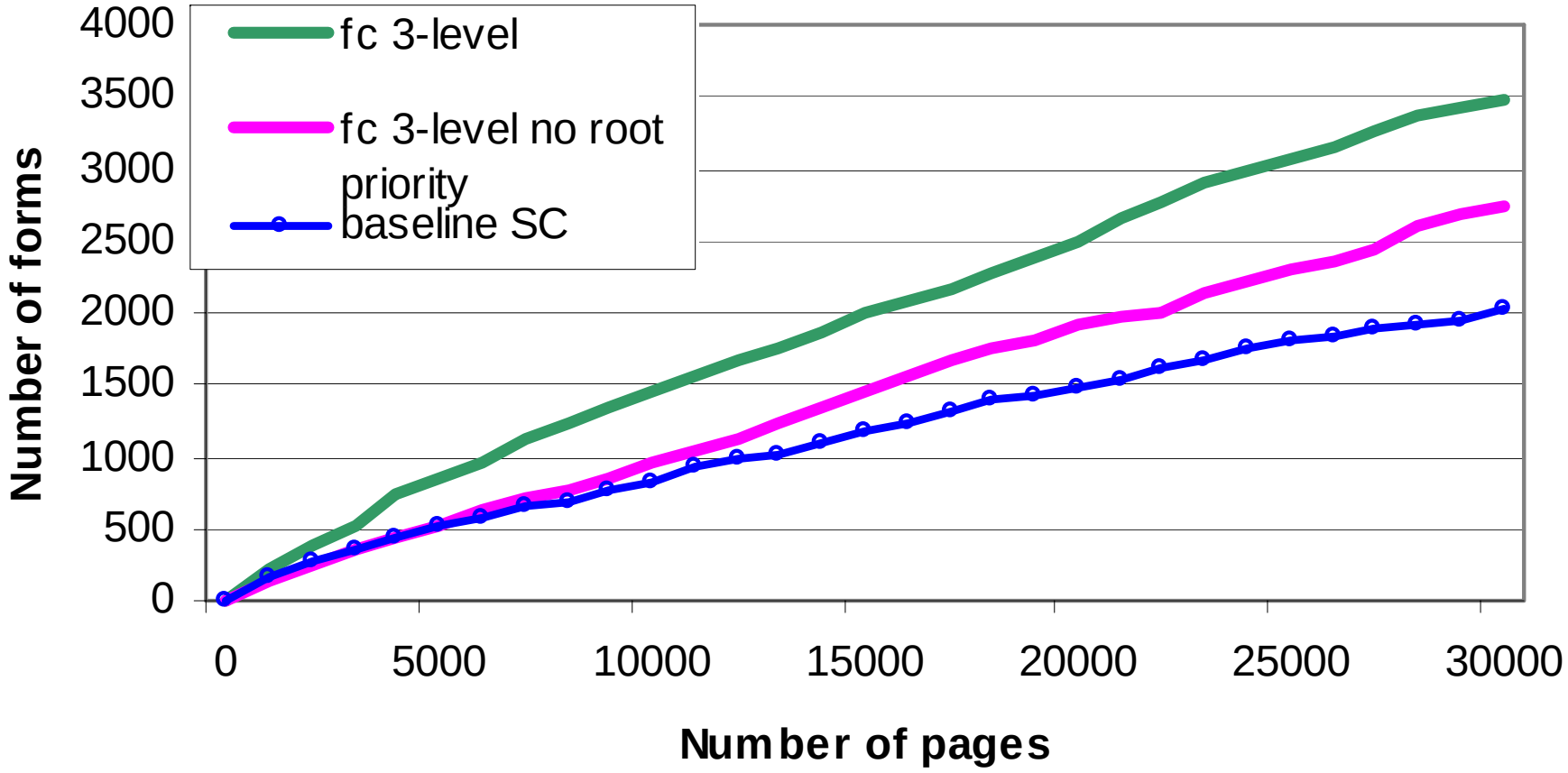
Results: Jobs



Results: Books



Prioritizing Root Pages: Jobs



Experiments: Summary

- Learning link/path features + considering delayed benefit is effective
  - Best performance: multi-level Form Crawler
  - 3 vs 1 level leads to 20-110% improvement
- Too many levels does not help
  - >=4, no improvement
- Stopping criteria are critical
  - Baseline SC is much better than baseline
- Fixing the crawling depth is not enough
- High priority to root pages helps

Related Work

- Focused crawlers:
  - Avoid off-topic pages (Chakrabarti et al, 1999 and 2002)
    - Only consider links that give *immediate benefit*
    - May miss relevant pages linked from irrelevant pages
  - Consider delayed benefit (Rennie and McCallum, 1999 and Diligenti et al, 2000)
- MetaQuerier Database Crawler (Chen et al, 2005)
  - No focus on topic, fixes depth of breadth-first crawl

Conclusion and Future Work

- New *efficient* crawling strategy to automatically discover hidden-Web databases
  - Focus on topic, prioritize promising links, use appropriate stopping criteria
- Issue: are the retrieved forms good?
  - Form classifier is effective, but not perfect
  - Can't check +3 thousand forms manually!
  - Further analyze forms in repository
- Hidden-Web database directory
  - Build form crawlers for different topics
  - Organize them in a hierarchy (a la DMOZ)