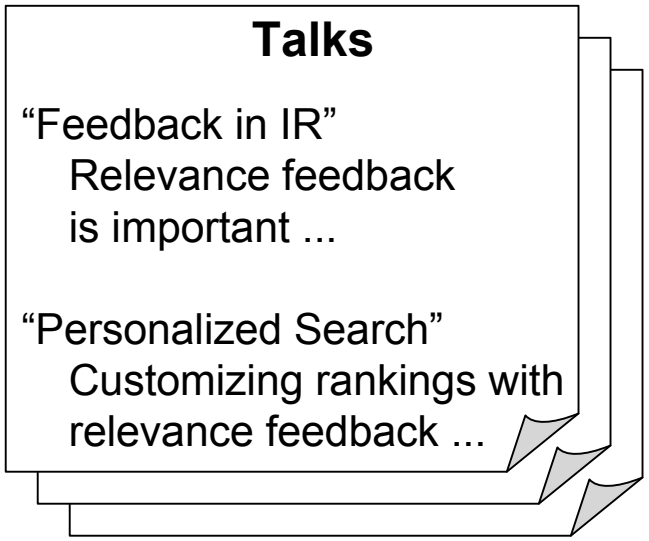
Declarative Information Extraction Using Datalog with Embedded Extraction Predicates

Warren Shen, AnHai Doan, Jeffrey Naughton
University of Wisconsin, Madison

Raghu Ramakrishnan
Yahoo! Research

Information Extraction

Extracting structured information from unstructured data



talks

title	abstract
“Feedback in IR”	“Relevance feedback is important...”
“Personalized Search”	“Customizing rankings with relevance feedback...”

IE Plays a Crucial Role in Many Applications

- **Examples**
  - Business intelligence
  - Enterprise search
  - Personal information management
  - Community information management
  - Scientific data management
  - Web search and advertising
  - Many more…

- **Increasing attention in the DB community**
  - Columbia, Google, IBM Almaden, IBM T.J. Watson, IIT-Bombay, MIT, MSR, Stanford, UIUC, UMass Amherst, U. Washington, U. Wisconsin, Yahoo! Research
  - Recent tutorials in SIGMOD-06, KDD-06, KDD-03

Previous Solutions Unsatisfactory

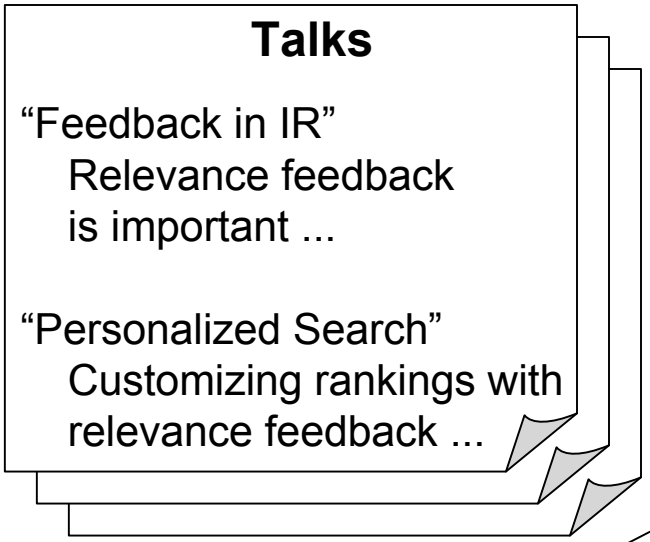
- **Employ an off-the-shelf monolithic "blackbox"**
  - Limited expressiveness
- **Stitch together blackboxes, e.g. with Perl or Java**
  - Example: DBlife
  - Difficult to understand, debug, modify, reuse, optimize
- **Compositional frameworks, e.g. UIMA, GATE**
  - Easier to develop IE programs
  - Still difficult to optimize because no formal semantics of interactions between blackboxes

Optimization However is Critical

- **Many real-world systems run complex IE programs on large data sets**
  - DBlife: Unoptimized IE program takes more than a day to process 10,000 documents
  - Avatar: IE program to extract band reviews from blogs takes 8 hours to process 4.5 million blogs

- **Optimization is also critical for debugging and development**

Proposed Solution:
Datalog with Embedded Procedural Predicates



docs
d_1
d_2
d_3

title	abstract
"Feedback in IR"	"Relevance feedback is important..."
"Personalized Search"	"Customizing rankings with relevance feedback..."

titles(d,t) :- docs(d), extractTitle(d,t). → perl module

abstracts(d,a) :- docs(d), extractAbstract(d,a). → C++ module

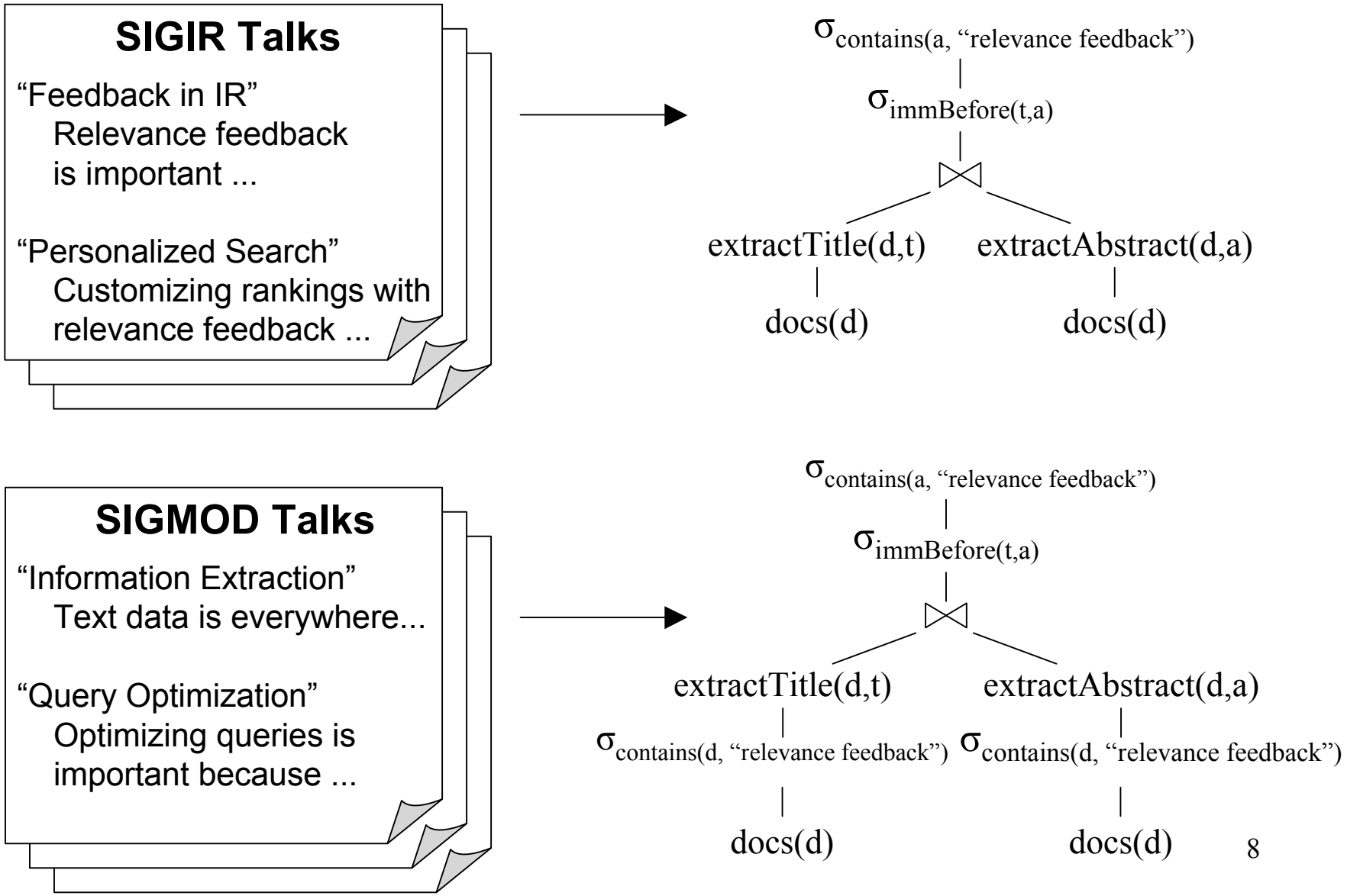
talks(d,t,a) :- titles(d,t), abstracts(d,a),
immBefore(t,a), → perl module
contains(a,"relevance feedback"). → perl module

Benefits of Our Solution

- **Easier to understand, debug, modify, reuse**
  - People already write IE programs by stitching blackboxes together
  - Stitching them together using Datalog is a more natural way

- **Can optimize IE programs effectively**
  - based on data set characteristics
  - automatically

Example 1


SIGIR Talks
"Feedback in IR"
Relevance feedback
is important ...
"Personalized Search"
Customizing rankings with
relevance feedback ...
σcontains(a, "relevance feedback")
σimmBefore(t,a)
extractTitle(d,t)
extractAbstract(d,a)
docs(d)
docs(d)
SIGMOD Talks
"Information Extraction"
Text data is everywhere...
"Query Optimization"
Optimizing queries is
important because ...
σcontains(a, "relevance feedback")
σimmBefore(t,a)
extractTitle(d,t)
extractAbstract(d,a)
σcontains(d, "relevance feedback")
σcontains(d, "relevance feedback")
docs(d)
docs(d)

Example 2

- **Tested our framework on an IE program in DBlife**
  - Originally took 7+ hours on one snapshot (9572 pages, 116 MB)
  - Manually optimized by 2 grad students over 3 days in 2005 to 24 minutes

- **Converted this IE program to our language**
  - Automatically optimized in 1 minute after a conversion cost of 3 hours by 1 student to 61 minutes

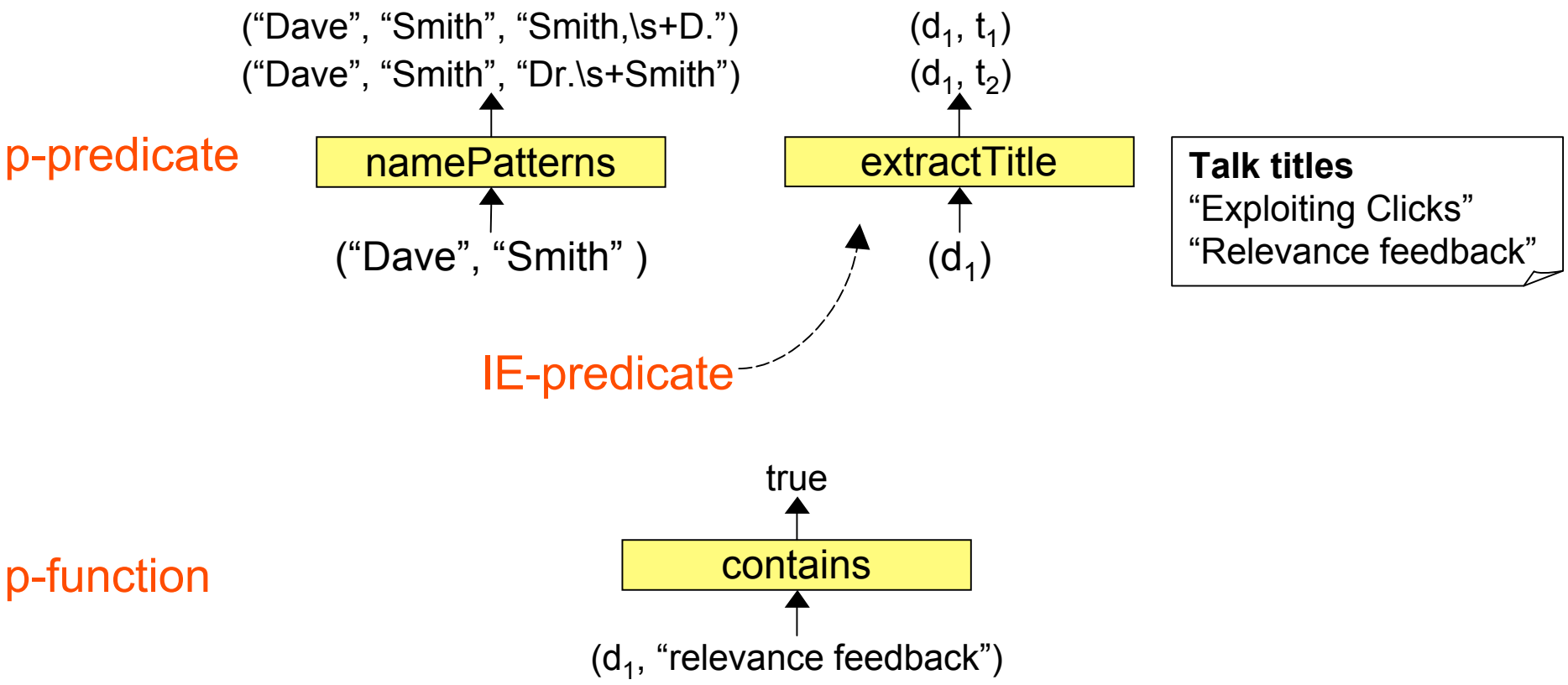
- **Our framework can drastically speed up development time by eliminating labor-intensive manual optimization**

Challenges and Contributions

- **How do we formally define the Datalog extension?**
  - Xlog language
- **How do we optimize IE programs?**
  - Three text-centric optimization techniques
  - Cost-based plan selection
- **Extensive experiments on real-world data**

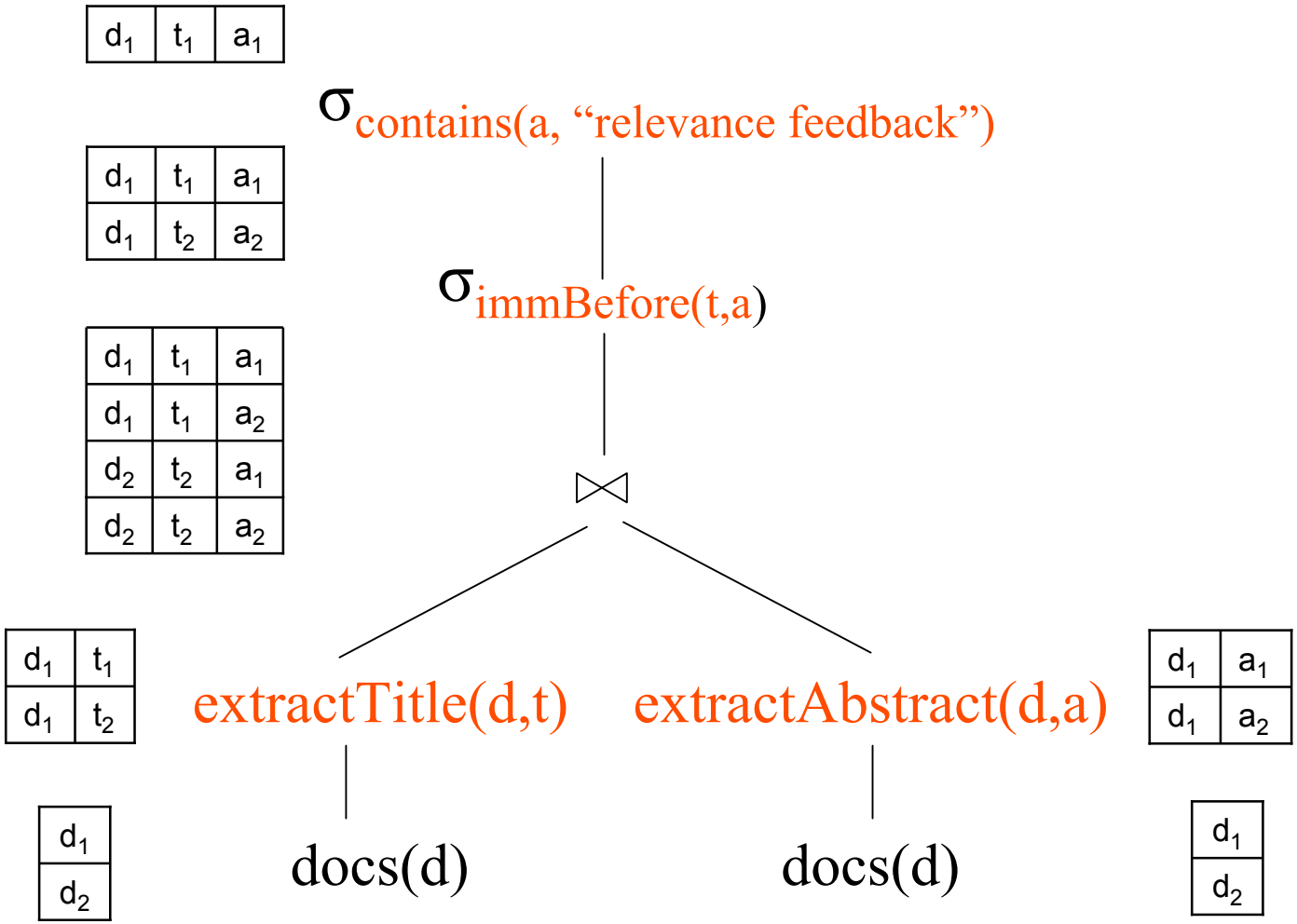
Xlog: Syntax

titles(d,t) :- docs(d), extractTitle(d,t).
abstracts(d,t) :- docs(d), extractAbstract(d,t).
talks(d,t,a) :- titles(d,t), abstracts(d,a), immBefore(t,a), contains(a,“relevance feedback”).

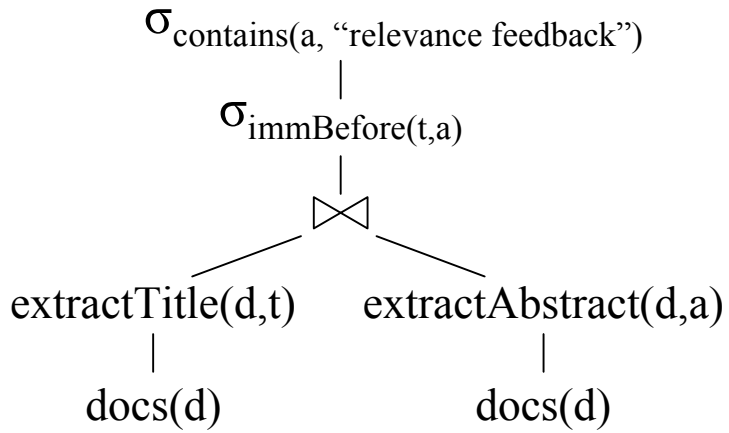


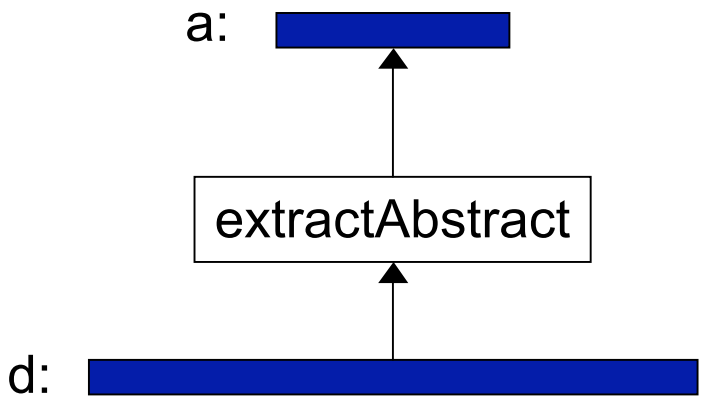
Xlog: Semantics

titles(d,t) :- docs(d), extractTitle(d,t).
abstracts(d,t) :- docs(d), extractAbstract(d,t).
talks(d,t,a) :- titles(d,t), abstracts(d,a), immBefore(t,a), contains(a,“relevance feedback”).

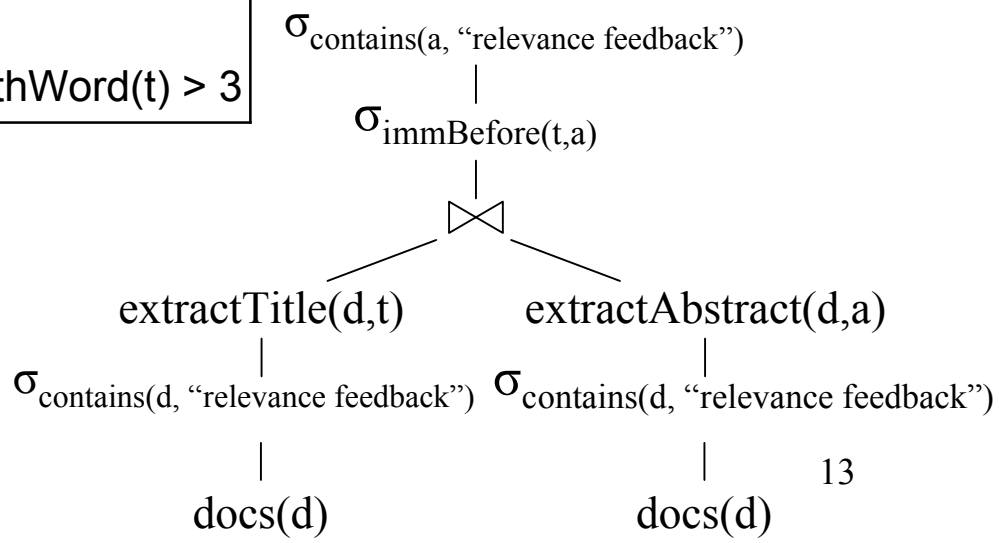


Optimization 1: Pushing Down Text Properties



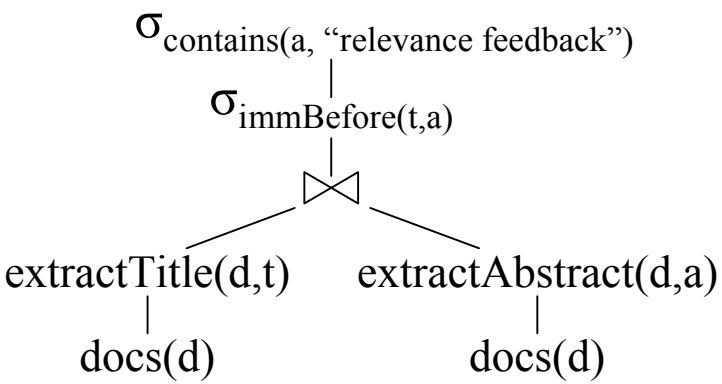


contains(a,w) Λ comes-from(a,d) → contains(d,w)
italics(s) Λ overlaps(s,t) → containsItalics(t)
(lengthWord(s) = 3) Λ comes-from(s,t) → lengthWord(t) > 3



Optimization 2: Scoping Extractions

- **Narrow the text regions that an IE predicate must operate over**
  - Exploit location conditions used to prune span pairs



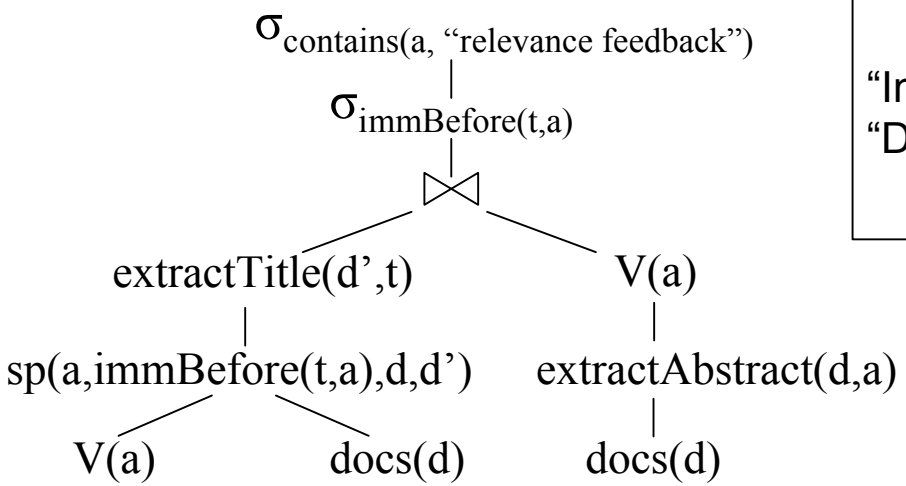
Talks

"Feedback in IR"
Relevance feedback
is important ...

"Personalized Search"
Customizing rankings with
relevance feedback ...

Papers

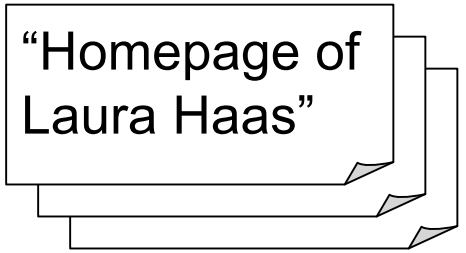
"Information Extraction"
"Data mining"
...



Optimization 3: Pattern Matching

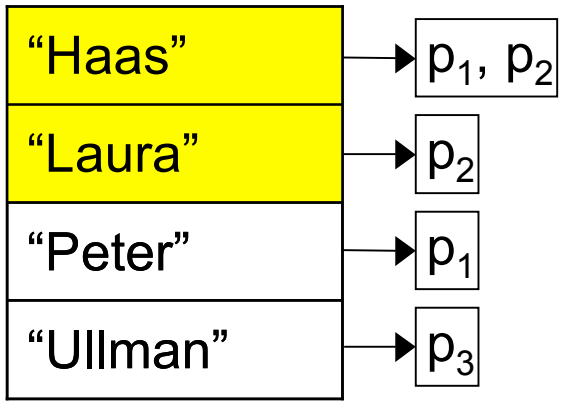
- **IE programs often match many patterns**

  p_1 = "Peter\s\s*Haas"
  p_2 = "Laura\s\s*Haas"
  p_3 = "(Jeff\s|Jeffrey\s)\s*Ullman"



- **Matching all patterns against all documents is expensive**
  - Unoptimized DBlife takes 14 hours to match 148,514 name patterns against 10,000 documents daily

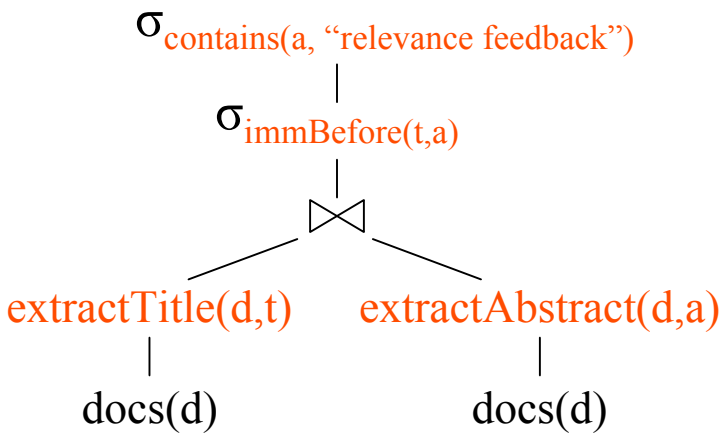
- **Usually only a few patterns occur in a document**
  - Index patterns to consider only promising patterns for each document



Candidate patterns:
p_1 , p_2

Estimating Plan Cost

- **Similar to estimating cost of relational plans with user-defined operators and functions**



- **But, need to adapt cost model to account for text data**
  - Model cost of IE-predicates to account for length of input text spans

Finding the Optimal Plan

- **At start, no statistics about procedural predicates and functions**

- **Adopt reoptimization strategy**
  1. Execute default plan for *k* documents and collect statistics for each procedural predicate and function
     - runtime
     - number of output tuples
     - extracted span lengths
  2. Update cost model with new statistics
  3. Search plan space for the plan with the lowest cost
  4. Finish executing with reoptimized plan

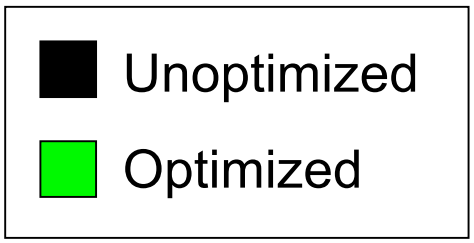
Experimental Setup

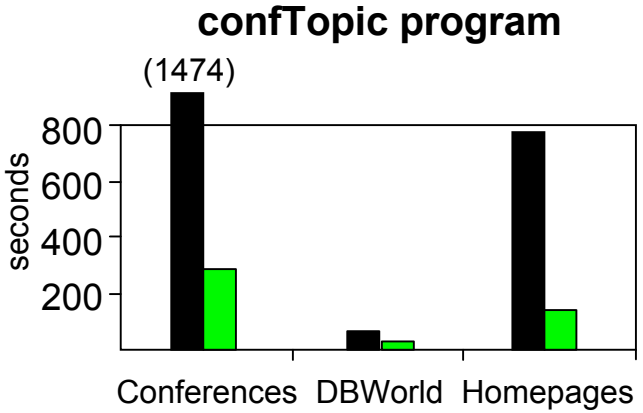
Data Set	Number of Documents	Size
Homepages	294	3.2 MB
DBWorld	90	5.5 KB
Conferences	142	2.5 MB

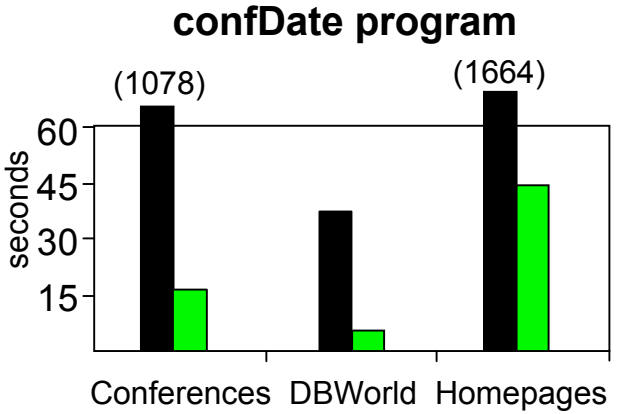
IE Programs	Description
confTopic	Find (X,Y) where topic X is discussed at conference Y.
confDate	Find (X,Y) where conference X is held during date Y.
affiliation	Find (X,Y) where person X is affiliated with organization Y.
advise	Find (X,Y) where person X is advising person Y.
chair	Find (X,Y, Z) where person X is a chair of type Y at conference Z.

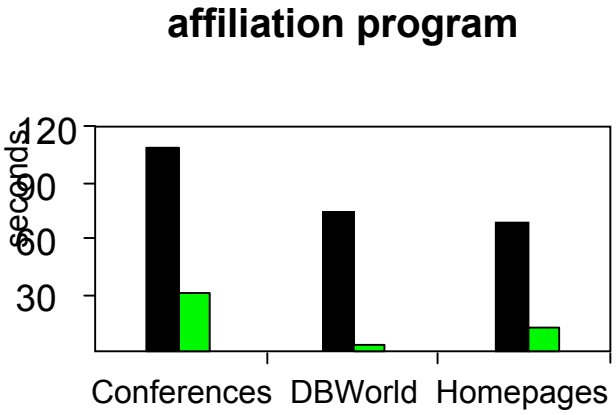
The Need for Optimization

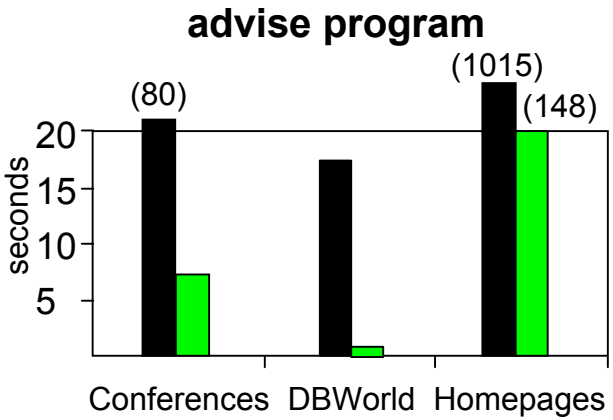
- **Optimization reduces runtime significantly by 52-99%**

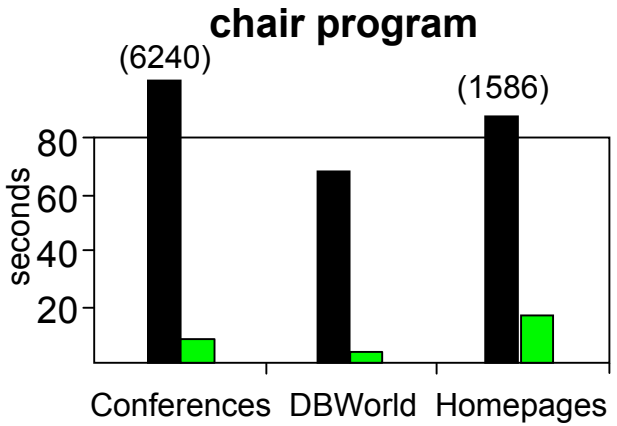






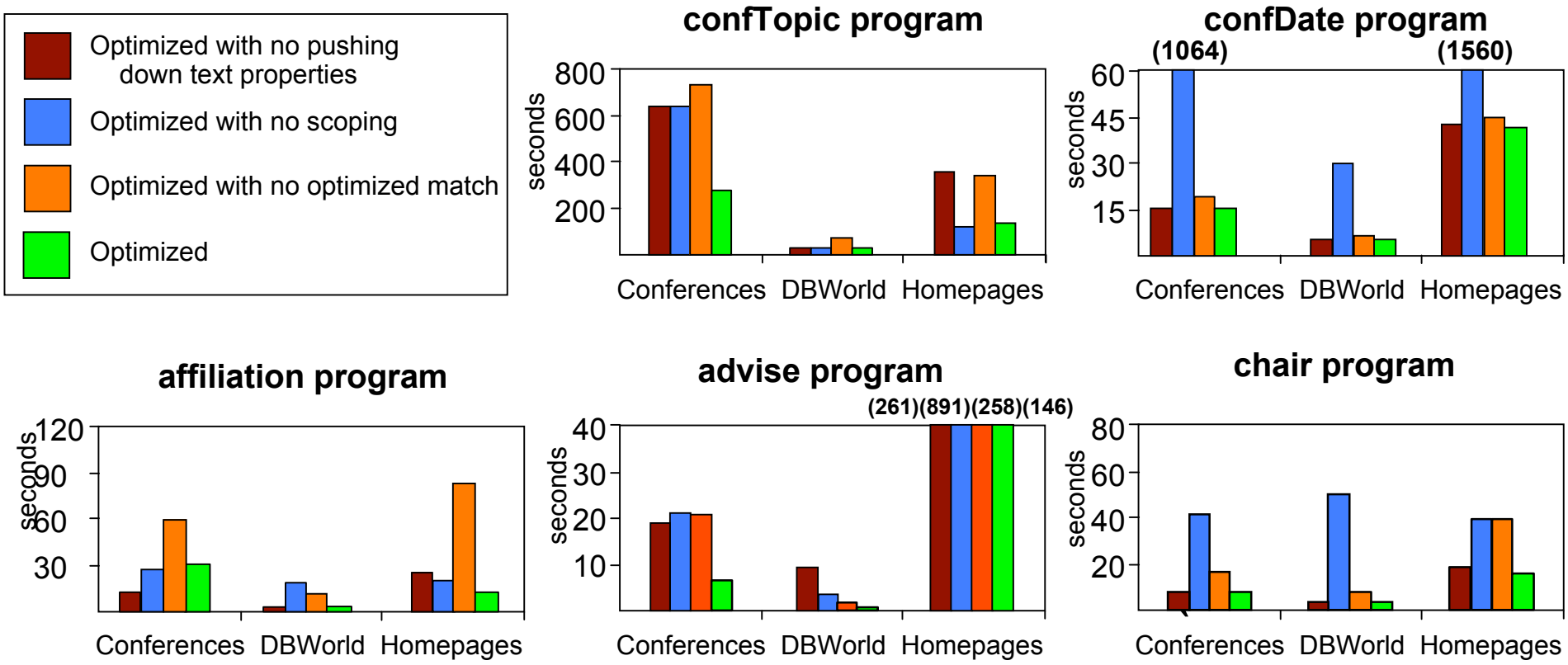






Component Contributions

- **Removing one optimization produces an inferior plan in 35 out of 45 cases**



The Need for Modeling Text Properties

- **Accounting for text span length in cost model results in a superior or comparable plan in 12 out of 15 cases**

Runtime (seconds)	confDate			confTopic			affiliation			advise			chair		
Data set	C	D	H	C	D	H	C	D	H	C	D	H	C	D	H
Optimized w/out modeling text span length	1056.0	33.2	1557.5	240.9	33.2	143.0	10.0	3.7	22.0	9.6	1.0	176.4	8.1	4.5	19.3
Optimized	16.8	5.6	45.5	297.7	33.2	147.0	30.7	3.7	12.6	7.1	0.9	146.0	8.1	3.9	16.1

- **Optimizing for given data set vs. optimizing for a different data set reduces runtime by 3-78% in 8 out of 15 cases**

  (See paper for more experiments)

Related Work

- **Much work on IE in the DB, AI, Web, and KDD communities**
  - Most works have focused on improving IE accuracy
  - See tutorials in KDD-03 and SIGMOD-06

- **Optimizing IE**
  - Pruning useless documents [Ipeirotis, Agichtein, Jain, Gravano, SIGMOD-06]
  - Incorporating large dictionaries [Chandel, Nagesh, Sarawagi, ICDE-06]
  - Indexing documents for pattern matching [Cho, Rajagopalan, ICDE-02]
  - See tutorial on scalable IE [Agichtein, Sarawagi, KDD-06]

- **Datalog extensions**
  - Wrappers in LIXTO [Gottlob, Koch, Baumgartner, Herzog, Flesca, PODS-04]
  - Declarative networking [Loo et al., SIGMOD-06]
  - Diagnosis of distributed systems [Abiteboul, Abrams, Haar, Milo, PODS-05]
  - Software analysis [Whaley, Lam, APLAS-05]

Conclusion and Future Work

- **Datalog with embedded IE predicates provides a natural framework to write IE programs**
  - Easier to understand, debug, modify, reuse
  - Can automatically optimize programs based on the data they are run on
- **Defined challenges and provided initial solutions**
  - Xlog language
  - Three text-centric optimizations
  - Cost-based plan search
- **Promising empirical results**
  - Tested on deployed system and real-world data
- **Future work**
  - Richer data models, recursion and negation
  - "Workbench" to let developers easily build scalable IE programs

The Need for Data-specific Optimization

- **Optimizing for given data set vs. optimizing for a different data set reduces runtime by 3-78% in 8 out of 15 cases**