

Vague Element Selection and Query Rewriting



for XML Retrieval



Vojkan Mihajlović
Djoerd Hiemstra
Henk Ernst Blok

Outline

- # Motivation
- # Modeling vague search in TIJAH
- # Experimental setup
- # Experimental Results
- # Conclusions & future work

Outline

- # Motivation
- # Modeling vague search in TIJAH
- # Experimental setup
- # Experimental Results
- # Conclusions & future work

Why vague search on structure?

- # Users do not know the exact document structure (DTD, Schema)
- # Users are not familiar with the content of different document elements (semantics)
 - Users are not sure where they want to search for info (search elements)
 - Users are not sure what they want to get as an answer (answer elements)
- # Different users exists

Why do we need vague search?

- # Users can make a distinction between
 - Strict element selection
 - Vague element selection
- # Users can express better their information need
- # Users should utilize IR systems' intelligence for vague search
 - Knowledge of related (similar) elements in the collection
 - Knowledge of the degree of elements' similarity

Vague search types

- # Vague search on terms
 - Synonyms, hypernyms, hyponyms
- # Vague search on element names & terms
 - Vague element name matching
 - Terms crossing structural boundaries
- # Vague search on element relations (schema matching)
 - Path matching
 - Graph (tree) matching

Vague element name selection

- # Variants: based on INEX content-and-structure (CAS) task:
 - Strict selection - SSCAS
 - Vague selection of search elements - VSCAS
 - Vague selection of answer elements - SVCAS
 - Vague selection of search and answer elements - VVCAS
- # Vague selection modeling
 - Expansion lists based on INEX 2004 assessments

Query rewriting techniques

- # Interchange the terms that are in the same search context (element)
- # Interchange the terms that are in different contexts (elements)

TIJAH architecture

- ✦ TIJAH - a prototype three-level database system for structured IR
 - Conceptual level: NEXI parsing, preprocessing, rewriting
 - Logical level: Score region algebra (SRA) query plan
 - Physical level: MonetDB interpreter language (MIL) procedures and binary tables



NEXI++



Query pre-
processing

**SRA Query
Plan
Generator**



MIL

MonetDB
kernel

Outline

- # Motivation
- # Modeling vague search in TIJAH
- # Experimental setup
- # Experimental Results
- # Conclusions & future work

Narrowed Extended XPath - NEXI

From XPath

- Descendant step: //
- Element and attribute selection: e.g., article, @datum
- Predicates: [path {<, >, =, ...} value]
- Combination in predicates: [predicate_1 or/and predicate_2]

Extension

- IR-like search: [about(path, terms) and/or ...]

Example:

- **//article[about(./sec//p, "digital libraries")]**

Vague element search in NEXI

- # NEXI + vague element selection
 - ~element_name, e.g., ~**article**, ~**sec**
- # Vague selection modeling
 - Element name expansion lists: e.g.,
expansion(exp_class, sec) := {(sec, 0.68), (abs, 0.128), ..., (app, 0.05)}
- # Why like this?
 - Various expansion classes can be used for different users and different tasks
 - Makes the system extensible and modular

Expansion lists used

- # Manual expansion lists with default down-weight $w=0.55$
 - ▣ Based on 2004 experiments
- # Seven automatically generated sets of lists
 - ▣ Based on 2004 assessments
 - ▣ Using graded assessments on two dimensions: exhaustivity (E) and specificity (S): high(3), fair(2), marginal(1), not relevant (0)
 - ▣ hh (E=3, S=3), hf (E=3, S>1), fh (E>1, S=3), ff (E>1, S>1), fm (E>1, S>0), mf (E>0, S>1), ff (E>0, S>0)

Manual element name expansion lists

Element name	Expanded element name	Down-weight
abs	abs, fm, kwd, vt, p, sec, article, bdy, ref	0.55
article	article, bdy, sec, abs, fm, bm, bib, bibl, bb, p, ref	0.55
atl	atl, st, fgc	0.55
bb	bb, bm, bibl, bib, atl, art	0.55
bdy	bdy, article, sec, abs, p, ref	0.55
bib	bib, bm, bb, atl, art	0.55
fig	fig, sec, st, p, fgc, st, atl	0.55
fm	fm, sec, abs, kwd, vt, p, article, bdy, ref	0.55
kwd	kwd, abs, fm, st, fgc, atl	0.55
p	p, vt, abs, sec, fm, article, bdy, st	0.55
sec	sec, abs, fm, vt, p, article, bdy, bm, app	0.55
st	st, atl, fgc	0.55
tig	tig, bb	0.55
vt	vt, p, sec, bm, fig	0.55

Rewriting techniques

```
//article[about(./fm/at1, "digital libraries")]  
//sec[about(., "information retrieval")]
```

Rewriting inside the same predicate - rw1

- `//article[about(./fm/at1, "digital libraries") and about(., "digital libraries")]//sec[about(., "information retrieval")]`

Rewriting on the query level - rw2

- `//article[about(./fm/at1, "digital libraries") and about(., "digital libraries " "information retrieval")]
//sec[about(., "information retrieval") and about(., "digital libraries")]`

Score region algebra - SRA

Data model: set of region tuples ($r \in R$),

$$r := (s, e, n, t, p)$$

▀ Region bounds: s - start, e - end

▀ Region info: n - name, t - type

▀ Region score: p

Operators:

▀ Selection $\sigma_{n=name, t=type}(R)$, $\sigma_{\diamond num}(R)$, $R_1 \supseteq R_2$, $R_1 \sqsubset R_2$

▀ Element relevance score computation $R_1 \supseteq_p R_2$

▀ Element score combination $R_1 \sqcap_p R_2$, $R_1 \sqcup_p R_2$

▀ Element score propagation $R_1 \blacktriangleleft R_2$, $R_1 \blacktriangleright R_2$

Vague selection in SRA

Operator	Operator definition
$\sigma_{n=name, t=type}(R)$	$\{r \mid r \in R \wedge r.n = name \wedge r.t = type\}$
$\sigma_{onum}(R_1)$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in C \wedge r_2.t = term \wedge r_2 \prec r_1 \wedge r_2.m \circ num\}$, where $\circ \in \{=, <, >, \leq, \geq\}$
$R_1 \supset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}$
$R_1 \sqsubset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_1 \prec r_2\}$
$R_1 \supset_p R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\supset}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$
$R_1 \triangleright R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\triangleright}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$
$R_1 \triangleleft R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\triangleleft}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$
$R_1 \sqcap_p R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, p_1 \otimes p_2) \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge (r_1.s, r_1.e, r_1.m, r_1.t) = (r_2.s, r_2.e, r_2.m, r_2.t)\}$
$R_1 \sqcup_p R_2$	$\{(r.s, r.e, r.n, r.t, p_1 \oplus p_2) \mid r \in R_1 \vee r \in R_2\}$
$\sigma_{n=name, t=type}^{expansion(class)}(R_1)$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, r.p) \mid r_1 \in R_1 \wedge r_1.t = type \wedge (r_1.n, r.p) \in expansion(class, name)\}$

$$r_1 \in R_1 \wedge r_1.t = type \wedge (r_1.n, r.p) \in expansion(class, name)$$

Retrieval models

Element score computation

- Language models (we can use others Okapi, tf.idf, ...)

$$f_{\square}(r_1, R_2) = r_1 \cdot p \left(\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2 \cdot p}{\text{size}(r_1)} + (1 - \lambda) \frac{|R_2|}{\text{size}(\text{Root})} \right)$$

Score combination

- AND like (\otimes) = product (also min, sum, ...)
- OR like (\oplus) = sum (also max, prob, ...)

Score propagation

- Sum

$$f_{\blacktriangleright}(r_1, R_2) = r_1 \cdot p \cdot \sum_{r_2 \in R_2 | r_1 \prec r_2} r_2 \cdot p$$

$$f_{\blacktriangleleft}(r_1, R_2) = r_1 \cdot p \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2 \cdot p$$

Outline

- # Motivation
- # Modeling vague search in TIJAH
- # Experimental setup
- # Experimental Results
- # Conclusions & future work

Collection

The Initiative for Evaluation of XML retrieval (INEX)

- Data: IEEE articles in XML format
 - 2004: ~500MB
 - 2005: ~750MB
- Topics: with structural conditions
 - 2004: content-and-structure (CAS)
 - 2005: CAS & content-only + structure (COS)
- Assessments: graded, two dimensional
- Metrics -> next slide

Evaluation: Metrics

For 2004 runs

- Precision at recall points
 - @10, @25, @50
- Official INEX 2004 metric - `inex_eval` MAP
 - Strict ($E=3, S=3$) and **generalized** ($S>0, E>0$) quantization

For 2005 runs

- Normalized extended cumulative gain `nxCG`
 - @10, @25, @50
- Effort precision - gain recall `ep-gr` **MAep**
 - Strict and **generalized** quantization

Topics for XXCAS and XXCOS

SSCAS & SSCOS

- `//article[about(./fm/atl, "digital libraries")]//sec[about(., "information retrieval")]`

SVCAS & SVCOS

- `//article[about(./~fm/~atl, "digital libraries")]//sec[about(., "information retrieval")]`

VSCAS & VSCOS

- `//article[about(./fm/atl, "digital libraries")]//~sec[about(., "information retrieval")]`

VVCAS & VVCOS

- `//article[about(./~fm/~atl, "digital libraries")]//~sec[about(., "information retrieval")]`

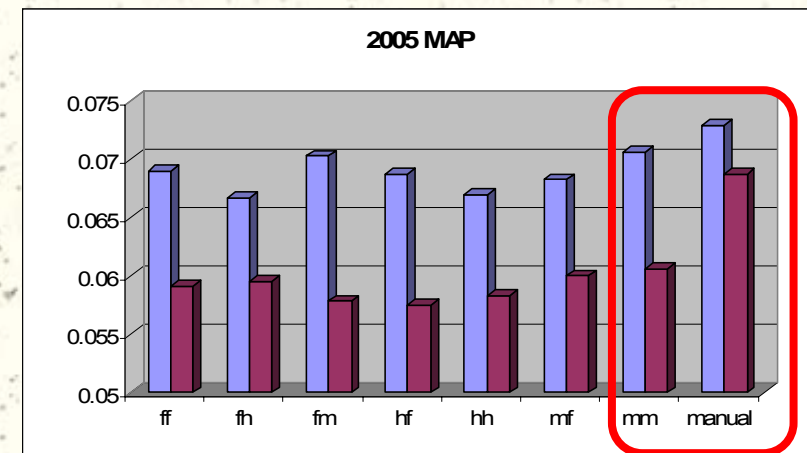
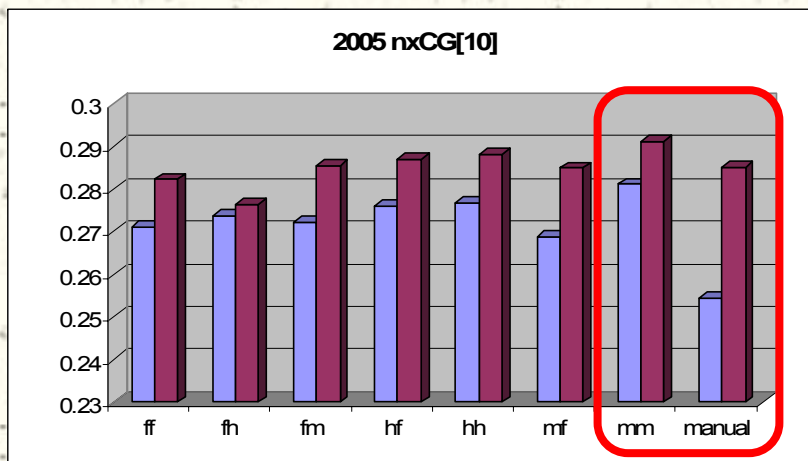
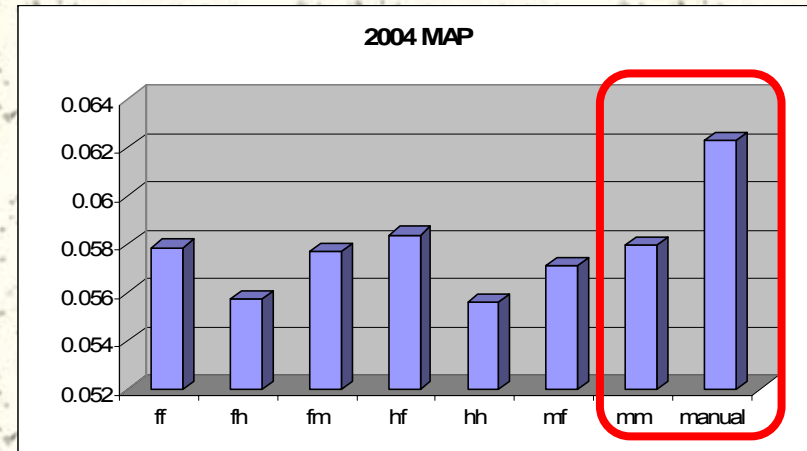
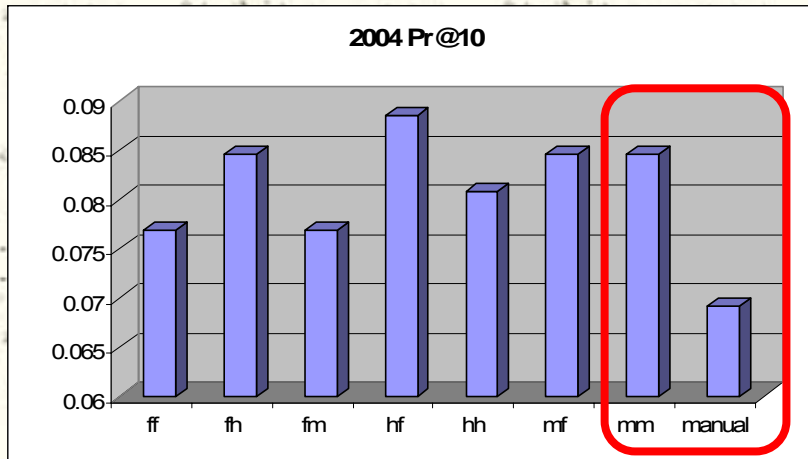
Rewriting topics

- # Performed automatically at the conceptual level

Outline

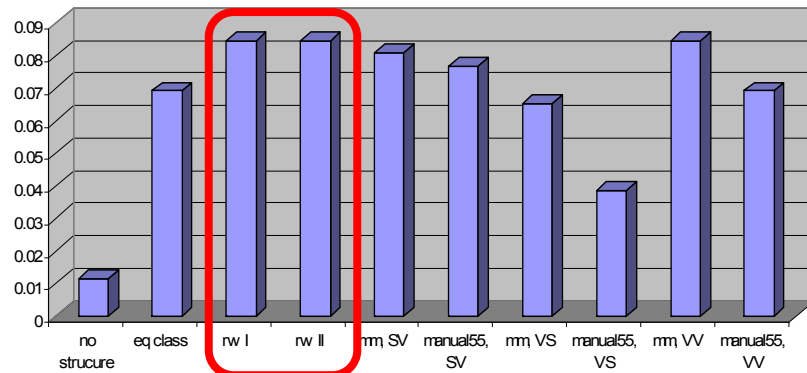
- # Motivation
- # Modeling vague search in TIJAH
- # Experimental setup
- # Experimental Results
- # Conclusions & future work

The best element name expansion

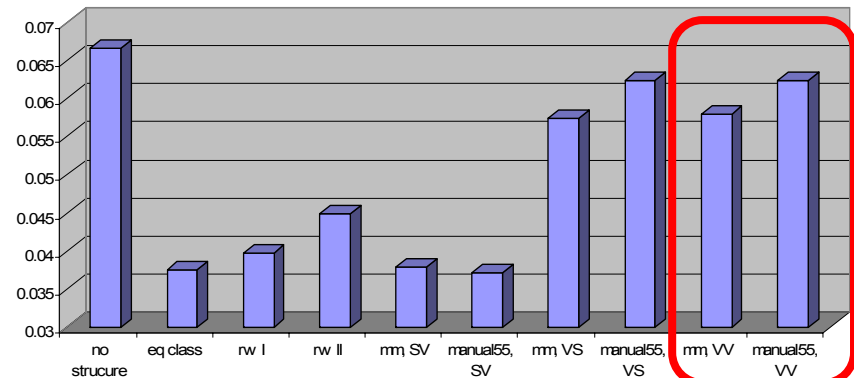


Vague selection vs. query rewriting

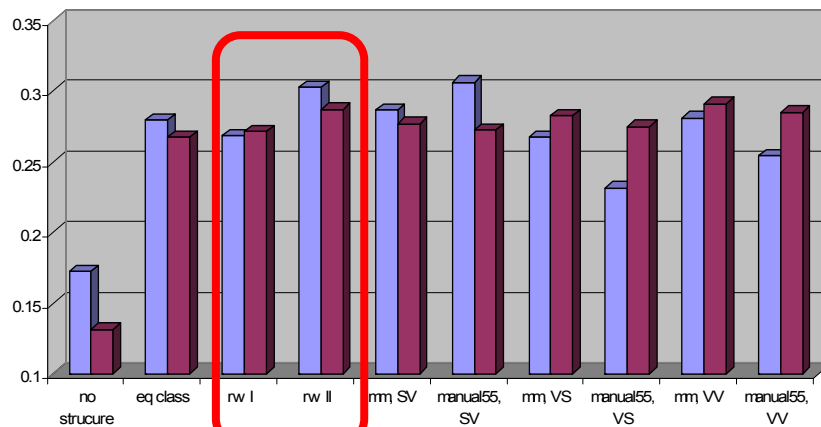
2004 Pr@10



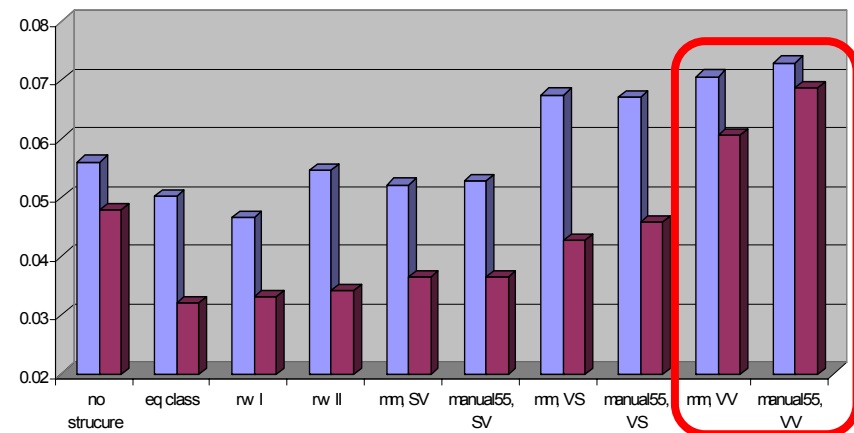
2004 MAP



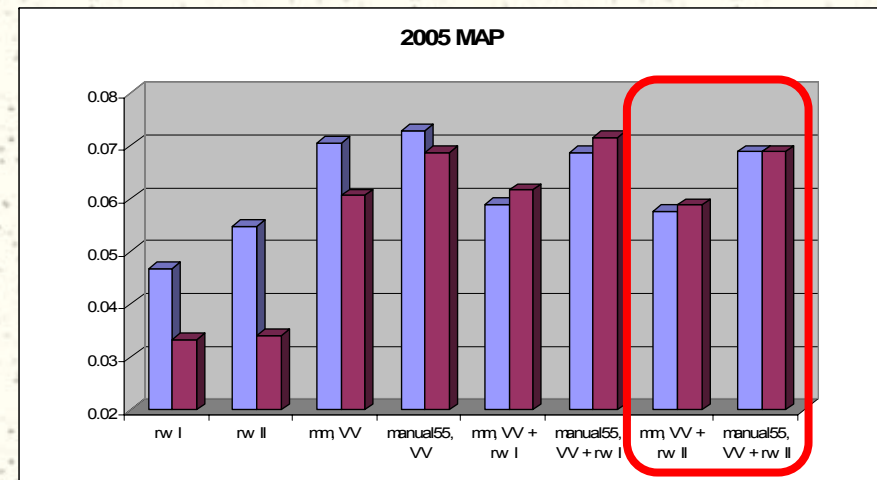
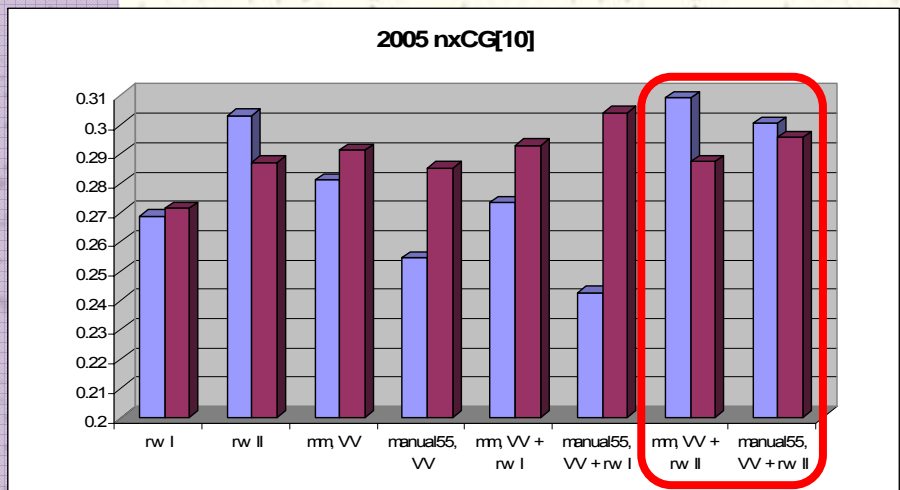
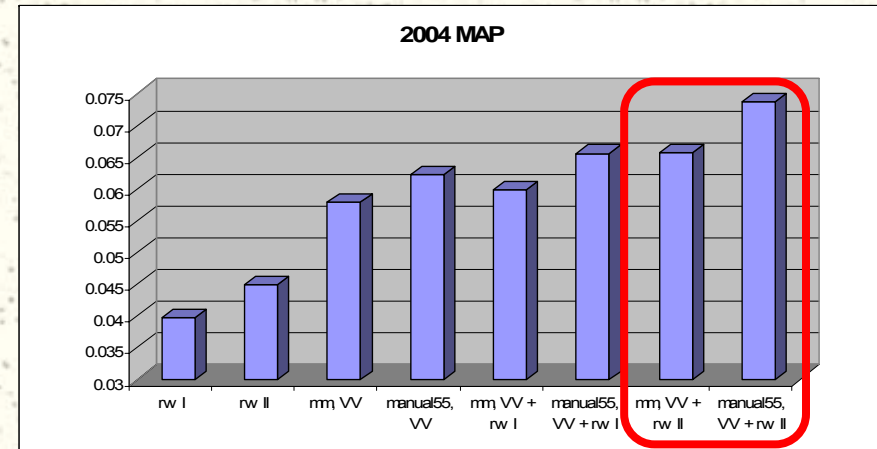
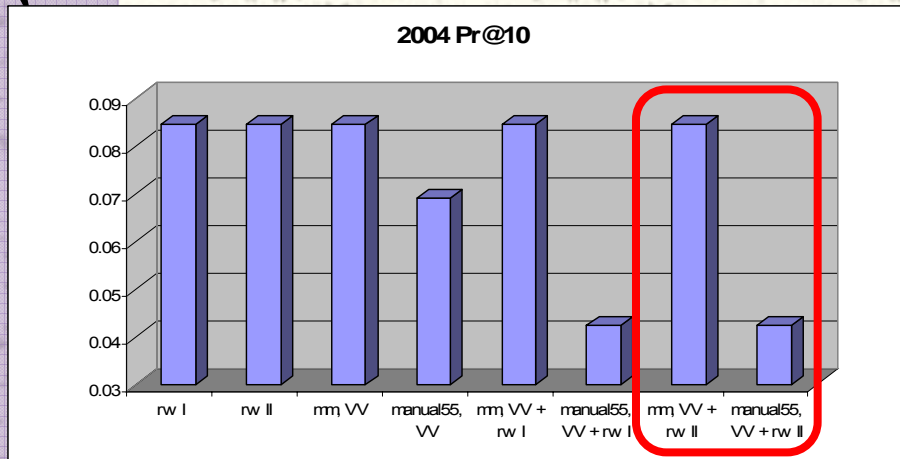
2005 nxCG[10]



2005 MAP



Vague selection & query rewriting



Outline

- # Motivation
- # Modeling vague search in TIJAH
- # Experimental setup
- # Experimental Results
- # Conclusions & future work

Conclusions

- # TIJAH can handle vague search (flexible)
- # Vague element name selection leads to a higher average precision => recall enhancing tool
- # Rewriting techniques result in higher precision @ low recall points => precision enhancing tool
- # Combination improves precision but MAP is not stable => overall good effectiveness

Future work

- # Different assignments of down-weights
 - E.g., using graded assessments
- # Different ways of generating expansion lists
- # More advanced rewriting techniques
- # Utilize some of the schema matching techniques
- # ... we need semantically richer, heterogeneous collection

End of slide show for the ones who are not interested in the discussion, click to exit.

Questions?

