Phrase Matching in XML

Divesh Srivastava
AT&T Labs–Research
http://www.research.att.com/~divesh/

Joint work with S. Amer-Yahia, M. Fernández and Y. Xu
Motivation

- **Phrase matching is common in information retrieval**
  - complements keyword matching
  - example: “To be or not to be”
  - word matches: contiguity or proximity

- **XML used for document markup**
  - XML tagging destroys contiguity of words
  - heavy tagging (e.g., Treebank) upsets proximity

- **Goal: Understand phrase matching in XML documents**
Motivation: Various XML Examples

- Jon Bosak’s Shakespeare’s plays in XML
  - identifies play elements: scenes, speeches
- The U Penn Treebank
  - identifies grammatical elements: nouns, verbs
- Library of Congress legislative bills
  - identifies bill elements: sponsor, committee-name
Motivation: Example

- **Annotated Shakespeare**
  
  ○ **problem:** text interleaves with arbitrary XML markup

  ```
  <SPEECH>
    <SPEAKER>HAMLET</SPEAKER>
    <LINE>To be, or not to be:
      <COMMENT> The line <QUOTE> To be, or not to be: that is the question </QUOTE> is one of the most quoted phrases in the English language. </COMMENT>
    </LINE>
  </SPEECH>
  ```

- **Solution:** specify tags, entire XML annotations to ignore
  
  ○ identify structural, semantic markup, commentaries on original text
  ○ **dynamic specification permits flexibility**
Problem Statement

- Given (pre-processed) XML database, and
- Proximity query specified by:
  - context node tags $C$
  - list of phrase words $W = [w_1, \ldots, w_q]$
  - ignore-tag tags $T$
  - ignore-annot tags $A$
  - proximity threshold $K$
- Identify all (context node, witness list) pairs in DB
Overview

- Motivation
- Problem statement
- Exact phrase matching
- Proximity phrase matching
- Experimental Results
- Related Work
- Open Problems
Exact Phrase Matching: Interval Encoding

- **(Start, End) numbering of XML elements and text**

```xml
<LINE(5,13)>
    To be, or not to be:
</LINE>

<SPEECH(1,12)>
    HAMLET)
</SPEECH>

<COMMENT(12,27)>
    The line
</COMMENT>

<QUOTE(15,26)>
    To be, or not to be:
</QUOTE>

<QUOTE(5,26)>
    To be, or not to be:
</QUOTE>

Properties

- **(anc, desc) ≡ interval containment**
- **word adjacency ≡ interval contiguity**
Exact Phrase Matching: Inverted Indexes

- \{(Tag/word, list of intervals)\} in Start order

```
<SPEECH(1,44)>
  <SPEAKER(2,4)>
    HAMLET\textsubscript{3}</SPEAKER>
  <LINE(5,43)>
    To\textsubscript{6} be\textsubscript{7}, or\textsubscript{8} not\textsubscript{9} to\textsubscript{10} be\textsubscript{11}:
  <COMMENT(12,38)>
    The\textsubscript{13} line\textsubscript{14} <QUOTE(15,26)>
      To\textsubscript{16} be\textsubscript{17}, or\textsubscript{18} not\textsubscript{19} to\textsubscript{20} be\textsubscript{21}:
      that\textsubscript{22} is\textsubscript{23} the\textsubscript{24} question\textsubscript{25}</QUOTE>
    is\textsubscript{26} one\textsubscript{27} of\textsubscript{28} the\textsubscript{29} most\textsubscript{30} quoted\textsubscript{31} phrases\textsubscript{32} in\textsubscript{33} the\textsubscript{34} English\textsubscript{35} language\textsubscript{36}.
  </COMMENT>
    that\textsubscript{39} is\textsubscript{40} the\textsubscript{41} question\textsubscript{42}:
  </LINE>
</SPEECH>
```

<table>
<thead>
<tr>
<th>\textit{L}_{\text{SPEECH}}</th>
<th>\textit{L}_{\text{LINE}}</th>
<th>\textit{L}_{\text{COMMENT}}</th>
<th>\textit{L}_{\text{to}}</th>
<th>\textit{L}_{\text{be}}</th>
<th>\textit{L}_{\text{question}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,44)</td>
<td>(5,43)</td>
<td>(12,38)</td>
<td>(6,6)</td>
<td>(7,7)</td>
<td>(25,25)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(10,10)</td>
<td>(11,11)</td>
<td>(42,42)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(16,16)</td>
<td>(17,17)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(20,20)</td>
<td>(21,21)</td>
<td></td>
</tr>
</tbody>
</table>

- Build B-tree on Start and End positions for probing

Divesh Srivastava

Phrase Matching in XML
Exact Phrase Matching: Output

- (context node, witness list) pairs in DB

```xml
<SPEECH(1,44)>
  <SPEAKER(2,4)>
    HAMLET
  </SPEAKER>
  <LINE(5,43)>
    To  be, or not to be:
  </LINE>
  <COMMENT(12,38)>
    The line
    "To be, or not to be:
    that is the question"
    is one of the most quoted phrases in the English language.
  </COMMENT>
</SPEECH>

{(1,44),
  { [ 6, 7, 8, 9, 10, 11, (12, 38), 39, 40, 41, 42 ],
    [ 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 ] }
}

- Each witness can be compacted to a single interval
Exact Phrase Matching: Notation

- $L_C$: index of all intervals of context nodes
- $L_{w_j}$: index of all intervals of word $w_j$
- $L_M$: index of all intervals of ignored markup
  - $L_{a_j}$: index of all ignore-annot intervals
  - $L E_{t_j}$: $\{(s, s), (e, e) \mid (s, e) \text{ in ignore-tag index } L_{t_j}\}$
Exact Phrase Matching: Indexed Nested Loops

- Generalizes relational indexed nested loops

  for each context interval $i_c$ in $L_C$ {
    witnessSet = { };
    index probe $L_{W_1}$ to find first interval $i_1$ such that descendant($i_1$, $i_c$);
    repeat {
      matchPos = 1; $m = [ i_1 ]$;
      repeat {
        probe ($L_{W_{(matchPos+1)}} \cup L_M$) to find $i_2$ with $i_2$.start = last($m$).end+1;
        if (no match found) break;
        if ($i_2 \in L_{W_{(matchPos+1)}}$) matchPos++; $m = append(m, i_2)$;
      } until (matchPos = $q$)
      if (matchPos = $q$) witnessSet = witnessSet $\cup \{ m \}$; /* complete witness */
      $i_1 = next(L_{W_1})$
    } until not(descendant($i_1$, $i_c$))
    output ($i_c$, witnessSet) }

- nested context nodes: repeated computation
- arbitrarily many probes of $L_M$
Exact Phrase Matching: PIX

while (not(empty(L))) {
    i = remove-first(L);
    if (i ∈ L_C) {
        /* i is context interval */
        if (not(empty(S)) && not(descendant(i, top(S).interval)))
            output-and-clean(i);
        new-interval(i);
    } else {
        /* i is word or ignored markup */
        if (empty(S)) break;
        if (not(descendant(i, top(S).interval))) output-and-clean(i);
        if (i ∈ L_M) {
            extend-with-markup(i);
            if (i ∈ L_{di}) /* i is nested annotation */
                new-interval(i);
        } else if (i ∈ L_{w,pi}) extend-with-word(i, pos);
    }
}
if (not(empty(S))) output-and-clean((0,0));

output-and-clean(i) {
    repeat {
        c = pop(S);
        if (c.interval ∈ L_C) /* context interval */
            output(c.interval, c.witnessSet);
        /* Propagate nested witnesses up stack */
        top(S).witnessSet ∪ = c.witnessSet;
    } until (empty(S) or descendant(i, top(S).interval));
}

extend-with-markup(i) {
    for each m ∈ top(S).matchSet {
        if (i.start = last(m.partialWitness).end + 1)
            m.partialWitness = append(m.partialWitness, i);
        else
            discard-partial-match(m)
    }
}

extend-with-word(i, pos) {
    if (pos = 1) {
        top(S).matchSet = top(S).matchSet ∪ ([i], 1);
    } else {
        for each m ∈ top(S).matchSet {
            if (m.matchPos + 1 = pos and i.start =
                last(m.partialWitness).end + 1)
                m.partialWitness = append(m.partialWitness, i);
            m.matchPos++;
            /* Once matched complete phrase */
            if (m.matchPos = q) {
                top(S).witnessSet ∪ = { m.partialWitness };
                discard-partial-match(m)
            }
        }
    } else discard-partial-match(m)
} } }

Divesh Srivastava

Phrase Matching in XML
Exact Phrase Matching: PIX Analysis

- Generalizes stack-based structural join algorithm
  - stack to identify (anc, desc) pairs
  - key reliance on interval numbering with contiguity
  - take order of phrase words into account
  - use markup intervals between phrase words

- Analysis
  - traverse each interval list once
  - bounded-size in-memory stack

- Asymptotically optimal (linear) I/O complexity
Proximity Phrase Matching: Key Ideas

- Only modify `extend-with-word()`
  - keep track of positions skipped, up to $K$
- Permits skipping over unknown words and tags
  - in addition to ignore-tags, ignore-annots
- Can also generalize INL using B-tree indexes
Experimental Results: INL vs PIX

- **Implementation**
  - *in Java, uses Berkeley DB package for indexes*

- **Data Sets**
  - *real data from Treebank linguistic corpus (WSJ, Brown corpus)*
  - *synthetic XMach data modified for text*
Experiments: Applicability of Known Results

- No context nesting, no ignored markup
  - akin to relational joins
- $| L_{w_1} |\ll | L_{w_2} |
  - INL is substantially better
- $| L_{w_1} |\sim | L_{w_2} |
  - PIX is superior
Experiments: Exploring Variability

- Vary number of witnesses and context nodes

- PIX is about four times faster than INL for all witnesses
  - repeated index probing for markup is expensive
Experiments: Exploring Variability

- Vary nesting of context nodes and annotations

- PIX is independent of nesting depth, INL increases linearly
  - repeated index probing for nested context nodes
Experiments: Real Data

- Treebank data, no control on dataset

<table>
<thead>
<tr>
<th>Witnesses reported</th>
<th>Witnesses reported</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>All</td>
</tr>
<tr>
<td>INL 0.06s</td>
<td>0.06s</td>
</tr>
<tr>
<td>PIX 25.21s</td>
<td>27.10s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>WSJprd</th>
<th>INL 3.91s</th>
<th>4.03s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PIX</td>
<td>0.45s</td>
<td>0.95s</td>
</tr>
<tr>
<td>Brown</td>
<td>INL</td>
<td>16.12s</td>
<td>16.12s</td>
</tr>
<tr>
<td></td>
<td>PIX</td>
<td>15.12s</td>
<td>15.13s</td>
</tr>
</tbody>
</table>

- Real data supports observed behavior on synthetic data
  - synthetic data was reasonably realistic
Experiments: Summary

- No context nesting, no ignored markup
  - INL/PIX tradeoffs $\sim$ relational joins
- Vary number of witnesses and context nodes
  - INL and PIX grow linearly, PIX is faster for all witnesses
- Vary nesting of context nodes and annotations
  - PIX is independent, INL increases linearly
- Real data supports observed behavior on synthetic data
Related Work

- **IR search engines ignore HTML tags for phrase matching**
  - no prior work on XML and phrase matching
- **Structural join algorithms [AJK+02]**
  - use of stack, asymptotic optimal I/O
- **Keyword matching in XML**
  - dynamic context ([SKW01], XRank [GSB+03], TIX [AYJ03])
  - word approximation (XXL [TW02], XIRQL [FG01])
Open Problems: Cost-based PPM

- A hybrid of INL and PIX is likely to be best
  - which inverted lists to scan
  - which inverted lists to index probe
  - which query plan

- Analogy with relational join expression evaluation
  - need a cost model
  - need cardinality estimates
Open Problems: Scoring and Ranking

- Not all witnesses are equal
  - different numbers of ignore-tags
  - different numbers and sizes of ignore-annots

- Not all context nodes are equal
  - different types of witnesses

- Generalize TF*IDF measure from IR
  - some preliminary ideas in paper
Open Problems: Top-K Answers

- **Partial proximity phrase matching**
  - what if not all phrase words match?
  - what if proximity threshold exceeded?
  - how to assign scores?

- **Dynamic context determination**
  - is specific partial match better than general match?
  - how does one get top-K context nodes?
Conclusions

- **Core contributions: XML phrase matching**
  - proposed useful query primitives
  - developed efficient algorithms
  - performed experimental validation
  - built PIX prototype on top of GALAX
- **Easily extended for phrase matching on streaming XML**
  - no indices, track element nesting
- **XML structure + text research**
  - wealth of challenging problems
Motivation: PIX

● Problem: XML tagging destroys words contiguity
  ○ heavy tagging (e.g., Treebank) upsets proximity

● Approach 1: Ignore all XML tags
  ○ used by IR search engines on HTML

● Why is this not enough?

<LINE>\texttt{To be, or not to be;}</LINE>
<COMMENT>\ldots one of the most quoted phrases \ldots</COMMENT>
\texttt{that is the question}:</LINE>

● Lesson: Need to ignore entire XML annotations
Motivation: PIX

- Approach 2: Specify ignore-annots, ignore other tags
  - identify commentaries on original text
  - identify parenthetical phrases

- Why is this not enough?

  <LINE>Be all my sins remember'd</LINE>
</SPEECH>
<LINE></SPEECH>

- Lesson: Use XML markup to identify boundaries
Motivation: PIX

- **Approach 3:** Specify ignore-annots, ignore-tags
  - data-driven or query-specified?
- **Argument for dynamic (query-time) specification**

  ```xml
  <LINE>The harlot’s cheek</LINE>
  <PP>beautied with plastering art</PP></LINE>
  <LINE>Is not more ugly to the thing that helps it</LINE>
  ```

- **Lesson:** Dynamic specification permits flexibility (usability?)