LE2I Laboratory - UMR/CNRS

University of Bourgogne
Dijon - France

XML Similarity

Background, Applications and Motivations

Richard CHBEIR, Ph.D.
Associate professor

Research Organization of Information and Systems
国立情報学研究所
National Institute of Informatics

Overview

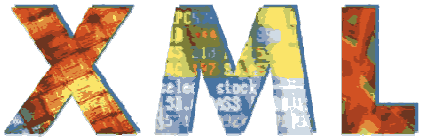
- Introduction
- Background
- Applications
- Our contributions
- Conclusion

What's XML?

Introduction and motivation

- XML (eXtensable Markup Language)
  - Structures and describes the content (and presentation)
  - Major means for efficient data representation and management

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Academy>
  <Department >
      <Laboratory>
            <Professor>John Cramer</Professor>
              <Student>John Takagi</Student>
      </Laboratory>
   </Department>
</Academy>
```



Plan :

Introduction
Background
Applications
Contributions
Conclusion

Why XML?

Introduction and motivation

- XML has become a *de facto* standard
  - Current applications
    - Multimedia Information description, storage and retrieval
    - Data exchange across enterprises and platforms (ontology mapping)
    - Web (services) interactions

Information destined to be shared
is henceforth represented using XML

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Introduction and motivation

How to make these two applications interoperate?

Introduction and motivation

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Emergent need

Information Retrieval and Database systems: XML documents comparison

XMLSearch

- Goal
  - Briefly introduce the XML similarity/comparison topic
    - Background
    - Applications
    - Motivations and personal contributions
    - Future research directions

Research Organization of Information and Systems
国立情報学研究所
National Institute of Informatics

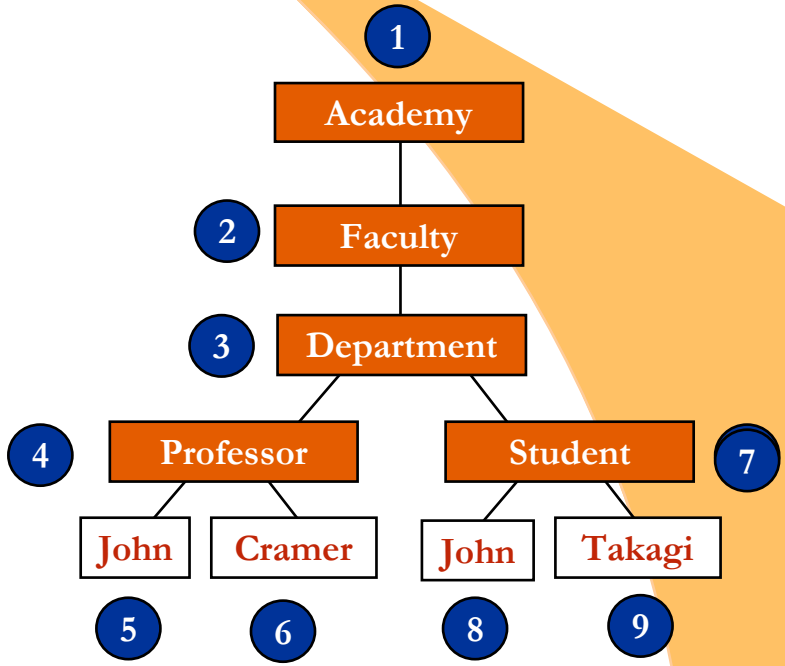
Overview

- Introduction
- Background
- Applications
- Our contributions
- Conclusion

Background

- XML (eXtensable Markup Language)
  - Can be observed as an OLT
    - Node label (x.l)
    - Node depth (x.d)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Academy>
  <Department >
        <Laboratory>
              <Professor>John Cramer</Professor>
              <Student>John Takagi</Student>
        </Laboratory>
  </Department>
</Academy>
```



Ordered Labeled Tree

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- There are two different views of XML
  - *Document-centric* view
    - Involves a liberal use of free-form text that is "marked up" with elements
  - *Data-centric* view
    - More emphasis on the structure (DB like data)
    - Data-centric documents are typically easier to process with computer programs because the data is better organized
    - Typical use of XML

```
<Memo>
  Please make sure you are at the
  <Location>ninth floor</Location>by
  <MeetingTime>10:30 AM </MeetingTime>
  to<Purpose>discuss the budget</Purpose>
</Memo>
```

```
<Memo>
  <MeetingTime>10:30 AM</MeetingTime>
  <Purpose>Discuss Budget</Purpose>
  <Location>ninth floor</Location>
</Memo>
```

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- A range of algorithms for comparing semi-structured data, e.g. XML documents, have been proposed:
  - Most of them exploit the *Artificial Intelligence* concept of *Edit Distance*
    - *Data-centric* view
  - Others focus on extending conventional *Information Retrieval* methods, e.g. the vector space model, so as to provide efficient XML similarity assessment
    - *Document-centric* view

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- *Edit Distance*

Finding the cheapest sequence of edit operations (insert, update, delete...) that can transform one string into another

h	e	l	o	o

h	e	l	l	o

Edit distance: 1 update + 1 insert

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- *Information retrieval*

Identifying important blocs

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- *Information retrieval*

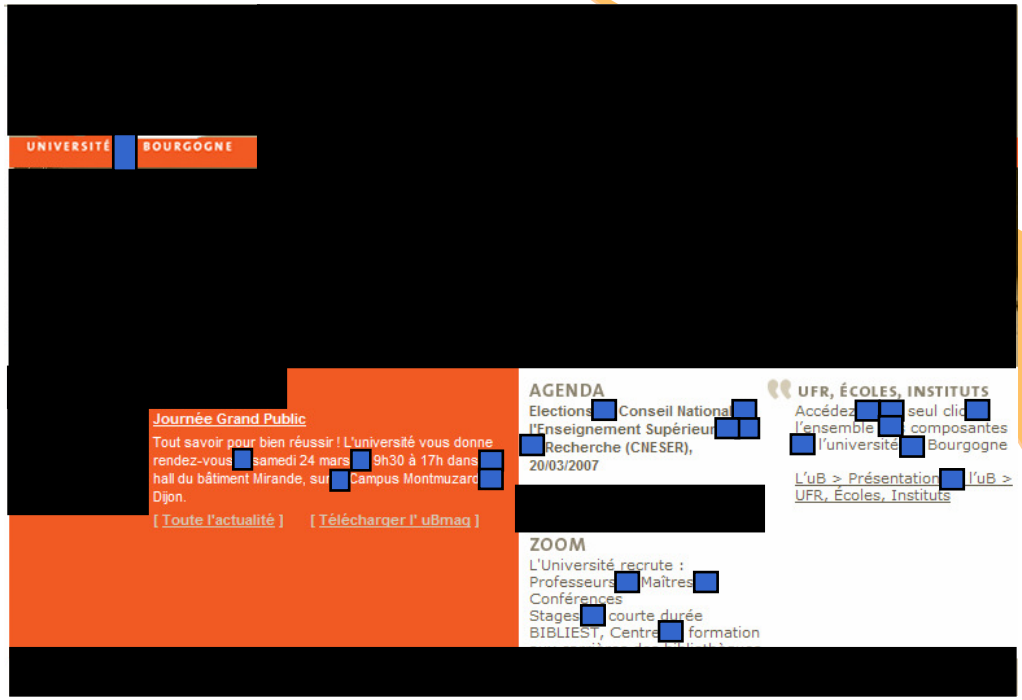
Identifying important blocs

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- *Information retrieval*



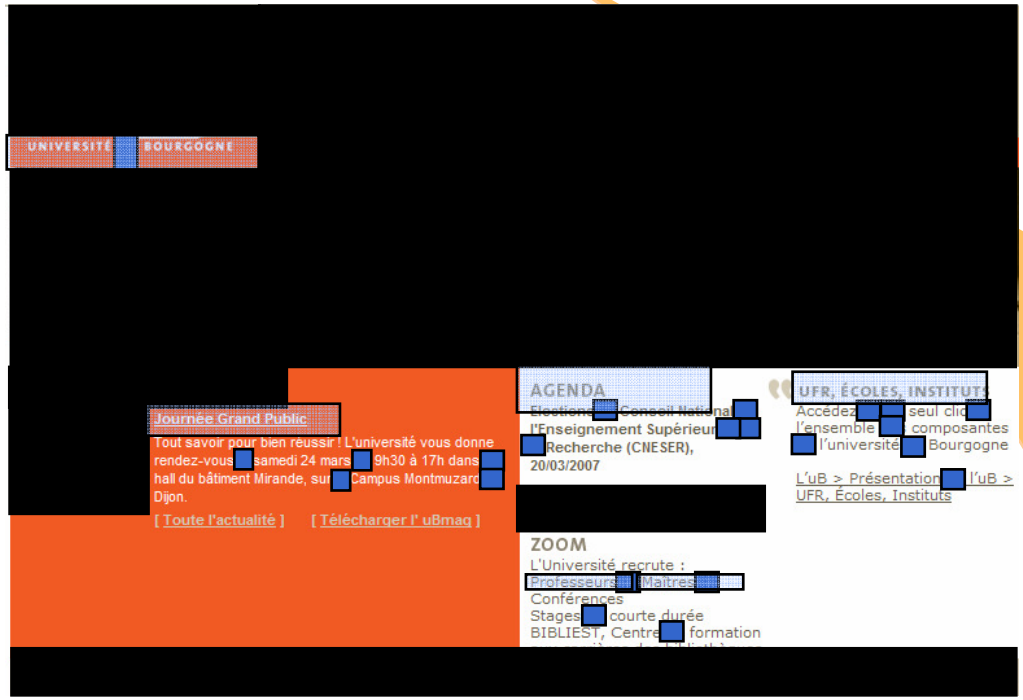
Eliminating stop words

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- *Information retrieval*



Identifying Concepts

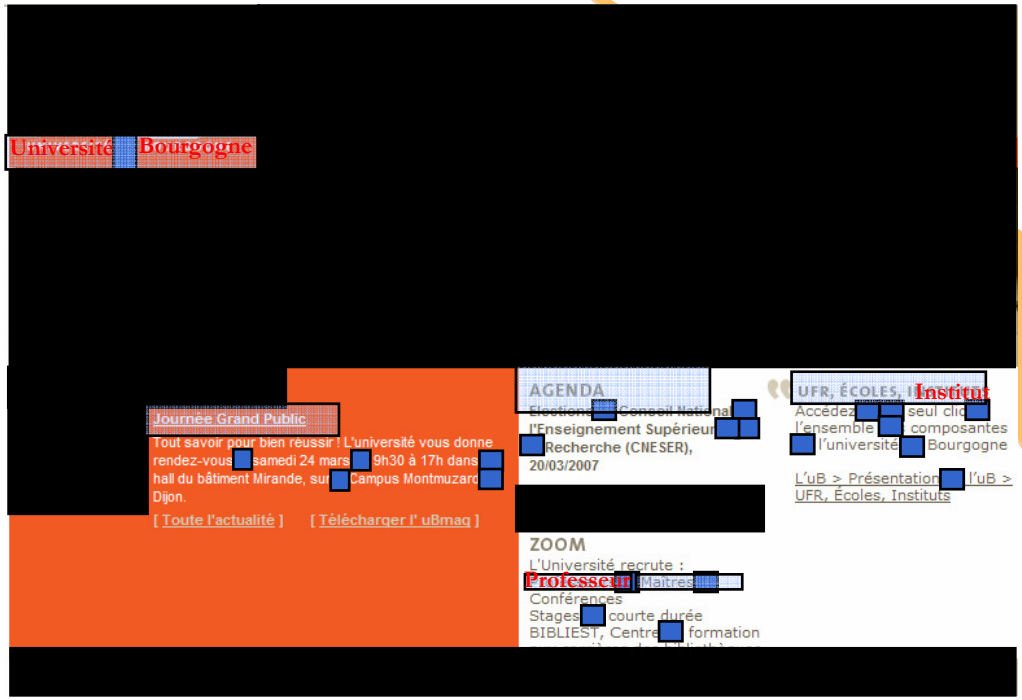
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Research Organization of Information and Systems
国立情報学研究所
National Institute of Informatics

Background

- *Information retrieval*



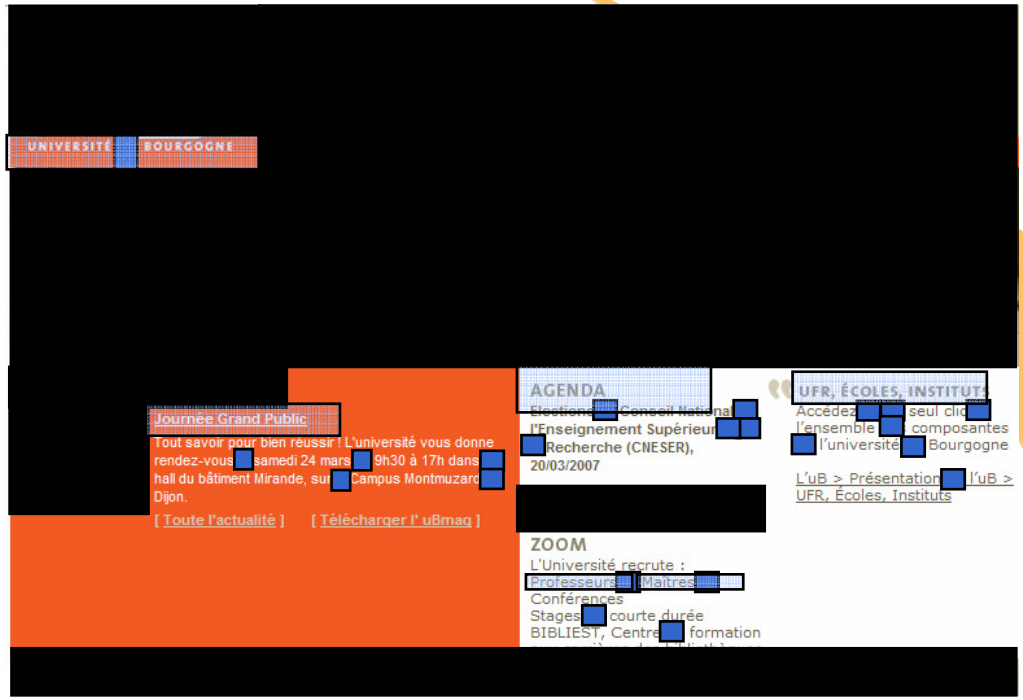
Vector: (Université, Bourgogne, Professeur, Institut)

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- *Information retrieval*



Vector: (Université, Bourgogne, Professeur, Institut)

And each term has its importance according to its weight

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Overview

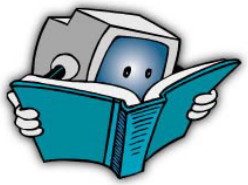
- Introduction
- Background
  - ED-based methods
  - IR-based methods
- Applications
- Our contributions
- Conclusion

Background

ED-based methods for XML similarity

- Most algorithms proposed in the literature make use of the *AI* programming techniques for finding the *Edit Distance* between trees

Finding the cheapest sequence of edit operations that can transform one tree into another



1.[2] Nierman A. and Jagadish H. V., Evaluating structural similarity in XML documents.
2.In *Proceedings of the 5th International Workshop on the Web and Databases,* 2002.

[3] Cobéna G., Abiteboul S. and Marian A., Detecting Changes in XML Documents. In *Proc. of the IEEE Int. Conf. on Data Engineering*, 2002, 41-52.

[4] Chawathe S., Comparing Hierarchical Data in External Memory. In *VLDB ,* 1999, 90-101.

[5] Shasha D. and Zhang K., Approximate Tree Pattern Matching. In *Pattern Matching in Strings, Trees and Arrays*, chapter 14, Oxford University Press, 1995.

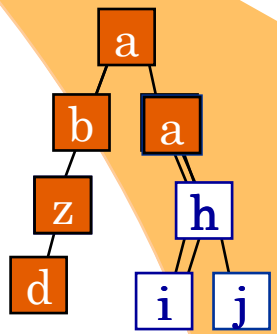
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

ED-based methods for XML similarity

- Algorithms can be distinguished following:
  - The set of edit operations allowed
    - *Insert node*: Insertion of inner/leaf nodes
    - *Delete node*: Deletion of inner/leaf nodes
    - U*pdate node*: Relabelling nodes
    - *Insert tree*
    - *Delete tree*
    - *Move tree*
  - The overall complexity and performance
    - $O(N^2 D^2)$
    - $O(N^2)$
    - $O(N \log(N))$ → Optimality

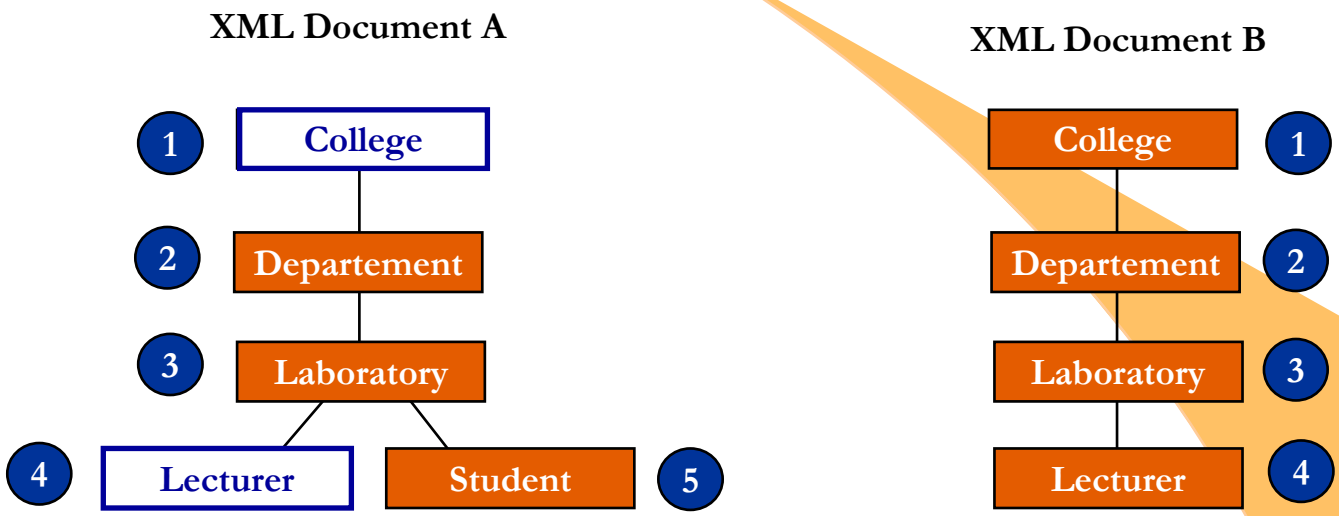


Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- Applying Chawathe's approach [2]



Edit script = **Upd(A[1], B[1]), Upd(A[4], B[4]), Del(A[5])**

Dist(A, B) = 3

Sim = 1 / 1 + Dist

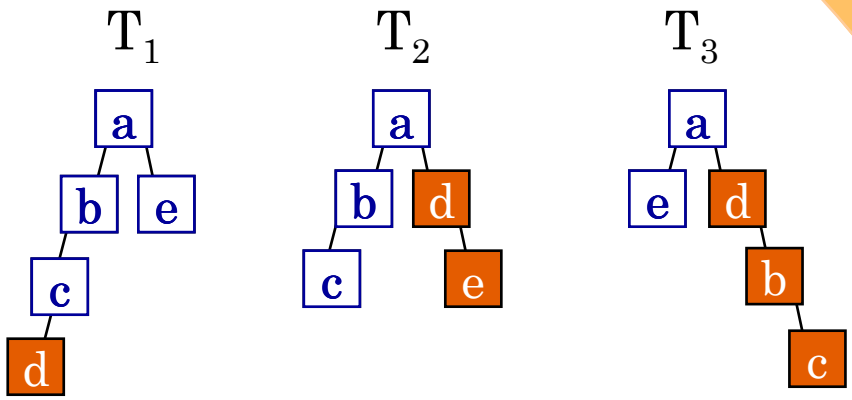
Sim(A, B) = 0.25

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

- Approximations of tree *ED*
  - Tag similarity [6]
  - Path similarity [7]
  - …



Plan :

Introduction
Background
Applications
Contributions
Conclusion

[6] Buttler D., A Short Survey of Document Structure Similarity Algorithms. In *Proceedings of the 5th International Conference on internet Computing*, Las Vegas, USA, 2004.

[7] Rafiei D., Moise D. and Sun D., Finding Syntactic Similarities between XML Documents. In *Proceedings of the 17th International Conference Database and Expert Systems Applications, 2006. DEXA '06*

[8] Flesca S., Manco G., Masciari E., Pontieri L., and Pugliese A., Detecting Structural Similarities Between XML Documents. In *Proceedings of the 5th International Workshop on The Web and Databases (WebDB 2002)*, 2002.

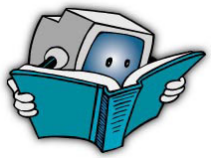
Overview

- Introduction
- Background
  - ED-based methods
  - IR-based methods
- Applications
- Our contributions
- Conclusion

Background

IR-based methods for XML similarity

- Traditional *IR* methods are no longer adequate with semi-structured data such as XML [9]
- With XML, the content is distributed at different levels
  - Therefore, it is to be treated differently w.r.t. flat content so as to improve retrieval precision [10]



Plan :

Introduction
Background
Applications
Contributions
Conclusion

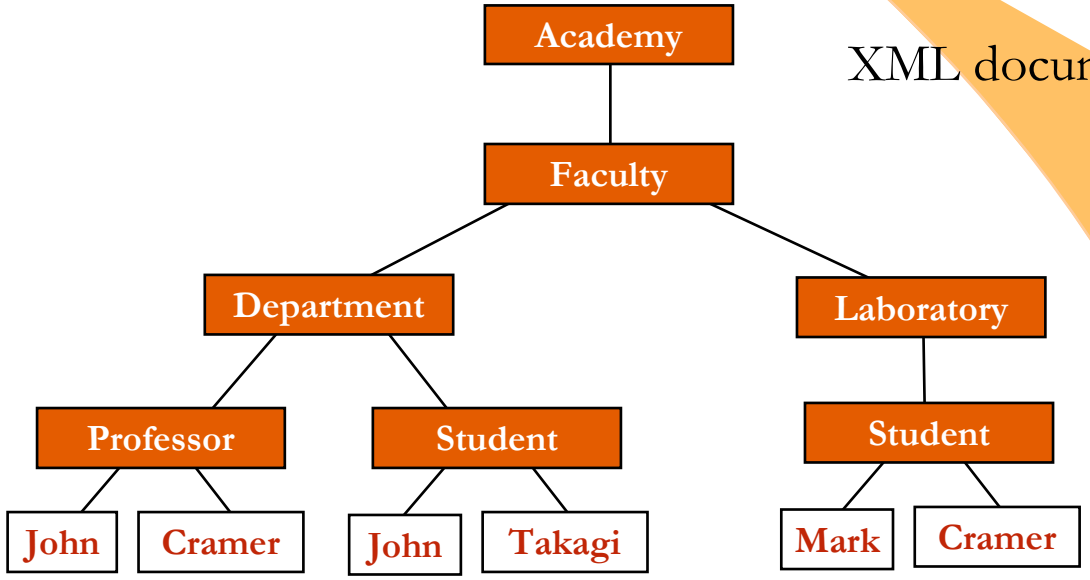
[9] Grabs T., Schek H.-J., Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In *Proceedings of SIGIR'2002 Workshop on XML and information Retrieval*, p.4-13, Finland, 2002.

[10] Schlieder T. and Meuss H., Querying and Ranking XML Documents. *JASIS Spec. Top. XML/IR* 53(6):489-503, 2002.

Background

IR-based methods for XML similarity

- With traditional *IR*



XML document D_1

	John	Cramer	Takagi	Mark
$V_{D1} =$	(2 ,	2 ,	1 ,	1)

→ TF weights

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

IR-based methods for XML similarity

- With traditional *IR*
  - Documents and user queries usually consist of sets of keywords where **Keywords are weighted in order to reflect their relative importance**
    - A *TF-IDF* score is generally utilized

$$Document = ("t_1,w_1", \ldots, "t_i,w_i", \ldots\ldots, "t_n,w_n")$$

$$Query = ("q_1,k_1", \ldots, "q_i,k_i", \ldots\ldots, "q_n,k_n")$$

How to calculate the similarity between them?

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

IR-based methods for XML similarity

- With traditional *IR*
  - Documents and user queries usually consist of sets of keywords where **Keywords are weighted in order to reflect their relative importance**
    - A *TF-IDF* score is generally utilized

$$Document = ("t_1,w_1", \ldots, "t_i,w_i", \ldots\ldots, "t_n,w_n")$$

$$Query = ("t_1,k_1", \ldots, "t_i,k_i", \ldots\ldots, "t_n,k_n")$$

By using the same semantic space

Plan :

Introduction
Background
Applications
Contributions
Conclusion

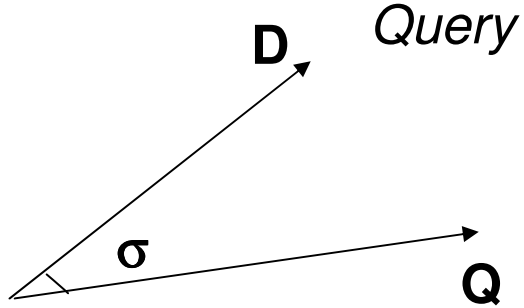
Background

IR-based methods for XML similarity

- With traditional *IR*
  - Documents and user queries usually consist of sets of keywords where **Keywords are weighted in order to reflect their relative importance**
    - A *TF-IDF* score is generally utilized

$$Document = ("w_1", \ldots, "w_i", \ldots\ldots, "w_n")$$

$$Query = ("k_1", \ldots, "k_i", \ldots\ldots, "k_n")$$



$$Similarity(D, Q) = \frac{\sum_{i=1}^{n}(w_i \times k_i)}{\sqrt{\sum_{i=1}^{n} w_i^2 \times \sum_{i=1}^{n} k_i^2}}$$

Following the *vector space model*, documents/queries become weight vectors

Similarity measures between vectors are put to use:

Cosine, Jaccard, Dice, …

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

IR-based methods for XML similarity

- With traditional *IR*
  - Identifying similar documents to a query comes down to:
    - Ranking the documents according to their similarity scores with the query (selection via a similarity threshold)

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

IR-based methods for XML similarity

- Extending traditional *IR* [9, 11]
  - Defining indexing nodes
  - Down-weighting the statistics (*TF-IDF* values) when the terms are propagated upwards in the document tree (*nested retrieval*)
  - Introducing the notion of *category* (*flexible retrieval*)
    - The users can dynamically – at query time – define the scope of their queries

Plan :

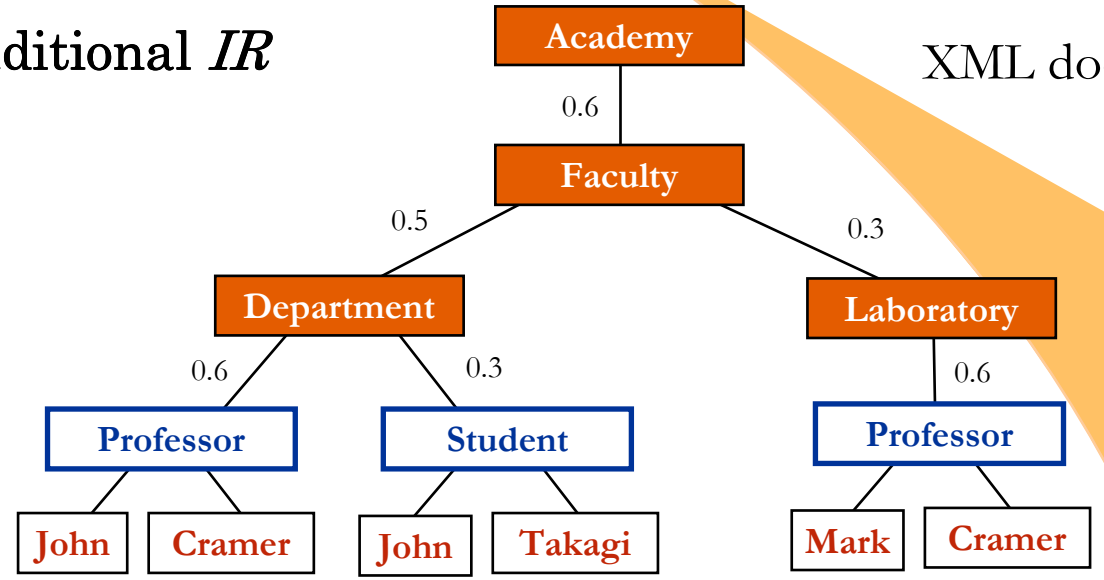
Introduction
Background
Applications
Contributions
Conclusion

1.[9] Grabs T., Schek H.-J., Generating Vector Spaces On-the-fly for Flexible XML Retrieval.
2.In *Proceedings of SIGIR'2002 Workshop on XML and information Retrieval*, p.4-13, Finland, 2002.
[11] Fuhr N. and Großjohann K., XIRQL: A Query Language for Information Retrieval.
In *Proceedings of ACM-SIGIR*, New Orleans, 2001, pp. 172-180.

Background

IR-based methods for XML similarity

- With traditional *IR*



XML document D_1

	John	Cramer	Takagi	Mark
$V_{Professor}$ =	(1 ,	1 ,	0 ,	1)
$V_{Student}$ =	(1 ,	0 ,	1 ,	0)
$V_{Department}$ =	(0.9 ,	0.6 ,	0.3 ,	0)
$V_{Faculty}$ =	(0.45 ,	0.48 ,	0.15 ,	0.18)

→ TF/IDF scores

} Down-weighted TF scores

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Background

IR-based methods for XML similarity

- **Extending traditional *IR*:**
  - Indexing terms w.r.t. their context [12]
    - *(ti, ci)* where each term *ti* is qualified by the context in which it appears.
    - *ci* is the path from the root to *ti*
    - Similarity between contexts (*LCS* algorithm)
  - …
  - Representing XML as matrixes [14]
    - A term is expressed by a vector $w_D(ti)_{1...k}$ reflecting the distribution of the term *ti* in the XML structure of the document collection, w.r.t. to each path *k* occurring in the collection
    - The similarity between a query and a document is evaluated as the correlation between two matrixes

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Overview

- Introduction
- Background
- Applications
- Our contributions
- Conclusion

In which applications, XML Similarity is useful?

Applications

- Four main areas:
  - XML data warehousing
  - XML data integration
  - XML classification and clustering
  - XML querying

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Applications

XML data warehousing

- The need of fine-grained method is mandatory in this field
  - Deltas, i.e. *edit scripts*: means of change detection and management

- Various uses:
  - Version control and change management
  - Querying the past
  - Learning about changes
  - Monitoring changes
  - Archiving
  - Mirroring

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Applications

XML data integration

- Integrating two XML data sources, comes down to performing an *approximate join* between these sources
  - Using a predefined XML similarity measure (most likely a tree edit distance based measure)

- Output of the *approximate join* between two data sources *S1* and *S2*:
  - All pairs of documents (*d1, d2*) of *S1* and *S2* such that $Sim(d1, d2) \geq s$ [15]
  - Subsequently, the identified pairs of documents are integrated to form unified views of the data.

[15] Guha S., Jagadish H.V., Koudas N., Srivastava D. and Yu T., Approximate XML Joins. In *Proceedings of ACM SIGMOD 2002*, pp. 287-298 (2002).

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Applications

XML classification and clustering

- Classification of XML documents gathered from the web against a set of DTDs declared in an XML database [16]
  - It can be undertaking using
    - An XML/DTD comparison method
    - Or an XML/XML comparison method (e.g. tree edit distance methods)
- XML clustering: Grouping similar XML documents together
  - Improves storage indexing [17], thus positively affecting the retrieval process
  - Plays a major role in effective DTD extraction [2]
  - Has a major impact on information extraction [6]

[16] Bertino E., Guerrini G., Mesiti M., A Matching Algorithm for Measuring the Structural Similarity between an XML Documents and a DTD and its Applications, *Elsevier Computer Science, 29*, 2004, 23-46.

[17] Schöning H., Tamoni – A DBMS Designed for XML. In *Proceedings of the ICDE Conference*, p. 149-154, 2001.

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Applications

XML query and search systems

- Retrieval of XML documents from an XML data collection
- The database community has proposed several languages for querying XML, such as XML-QL [18], XQL [19], XQuery [20], …
  - These languages are based on exact matching and do not support ranked queries
- Several attempts to extend these languages
  - Most of these are IR-based methods for XML similarity
  - Examples: ELIXIR [21], XIRQL [22], approXQL [23], XQuery/IR [24], …

Hiroshi Ishikawa, Kazumi Kubota, Yasuhiko Kanemasa, Yasuo Noguchi: The Design of a Query Language for XML Data. DEXA Workshop 1999: 919-922

Hiroshi Ishikawa, Shohei Yokoyama, Manabu Ohta, Kaoru Katayama: On Mining XML Structures Based on Statistics. KES (1) 2005: 379-390

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Overview

- Introduction
- Background
- Applications
- Our contributions
- Conclusion

Overview

- Introduction
- Background
- Applications
- Our contributions
  - Integrating semantics in XML similarity
  - Enhancing XML structural similarity
- Conclusion

Contributions

Integrating semantics in XML similarity

- Most algorithms for comparing semi-structured data, e.g. XML documents:
  - Focus on the structure of XML documents
  - Ignore the semantics involved

However, in the field of information retrieval (IR), estimating semantic similarity between documents is of key importance to improving search results [25]

Plan :

Introduction
Background
Applications
Contributions
Conclusion

[25] Maguitman A. G., Menczer F., Roinestad H. and Vespignani A., Algorithmic Detection of Semantic Similarity. In *Proceedings of the 14th International World Wide Web Conference*, 107-116, Chiba, Japan, 2005

Contributions

- Example :

```
<Academy>
  <Department >
    <Laboratory>
      <Professor></Professor>
      <Student></Student>
    </Laboratory>
  </Department>
</Academy>
```

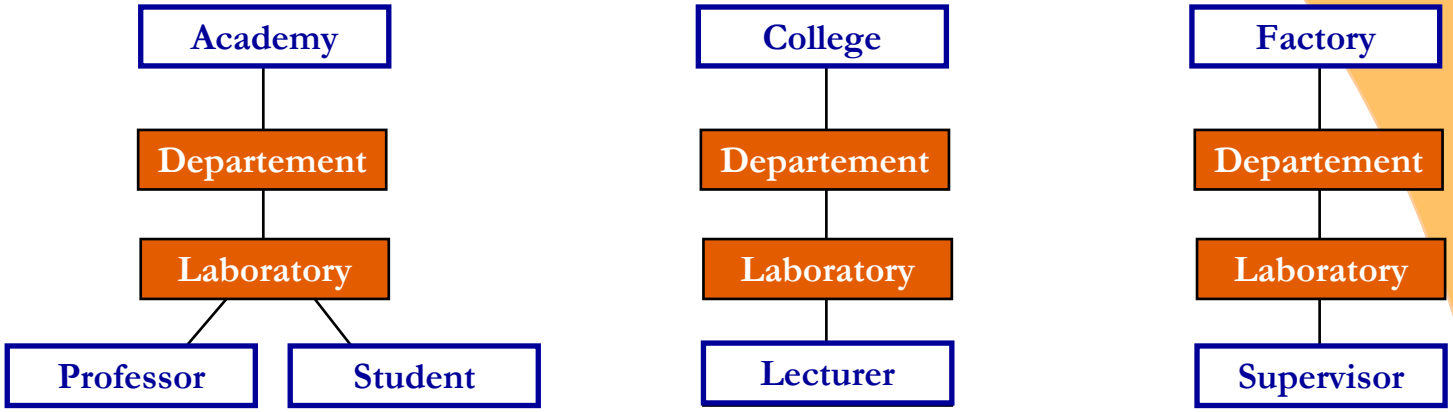
XML Document A

```
<College>
 <Department >
   <Laboratory>
     <Lecturer></Lecturer>
   </Laboratory>
 </Department>
</College>
```

XML Document B

```
<Factory>
 <Department >
   <Laboratory>
     <Supervisor></Supervisor>
   </Laboratory>
 </Department>
</Factory>
```

XML Document C



$$\mathrm{Sim}(A, B) \overset{?}{=} \mathrm{Sim}(A, C)$$

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Integrating semantics in XML similarity

Motivation

How to amend existing XML comparison approaches in order to take into consideration both structural and semantic characteristics of XML documents ?

Plan :

Introduction
Background
Applications
Contributions
Conclusion

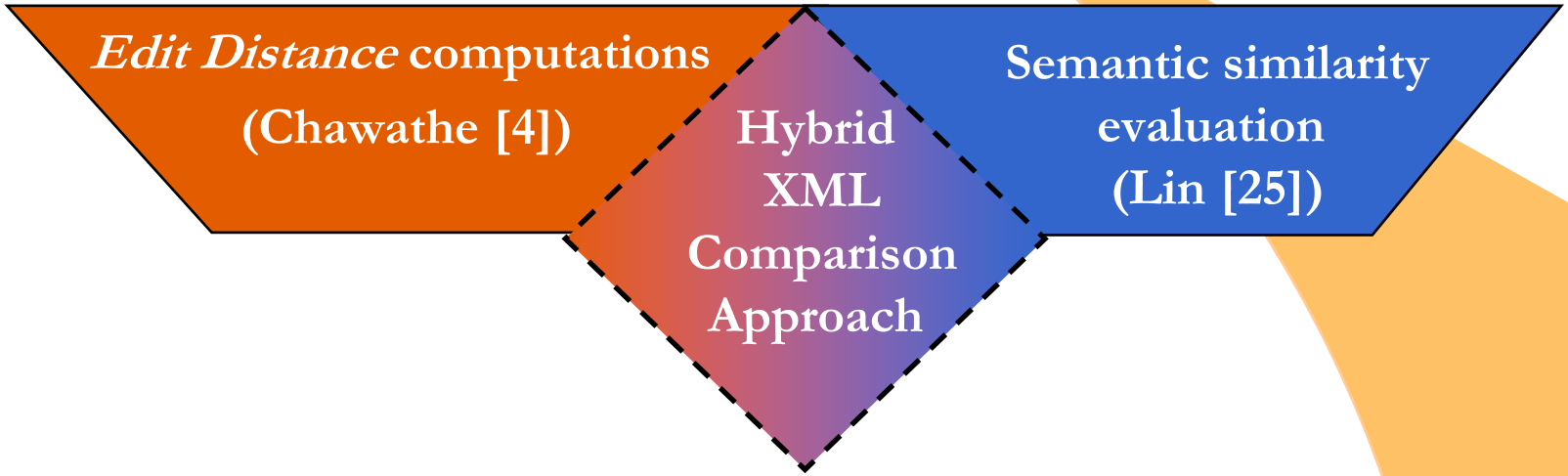
Contributions

Integrating semantics in XML similarity

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Proposal



[4] Chawathe S., Comparing Hierarchical Data in Extended Memory. In *Proceedings of the Twenty-fifth International Conference on Very Large Data Bases. Edinburgh, Scotland, U.K.,* p. 90-101, 1999

[25] Lin D., Am Information-Theoretic Definition of Similarity. *In Proceedings of the 15th International Conference on Machine Learning*, 296-304, Morgan Kaufmann Publishers Inc., 1998

Contributions

A central question in most edit distance approaches:

How to assign edit operations costs?

- Intuitive cost model:
  - $Cost_{Ins} = 1$
  - $Cost_{Del} = 1$
  - $Cost_{Upd} = 1$ when $x.l \neq y.l$ otherwise $Cost_{Upd} = 0$

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Integrating semantics in XML similarity

- Edit operations:
  - $Cost_{Sem_Upd}(x, y) = 1 - Sim_{Sem}(x.l, y.l)$
  - $Cost_{Sem_Ins}(x, i, p, \lambda(x)) = 1 - Sim_{Sem}(\lambda(x), p.l)$
  - $Cost_{Sem_Del}(x, p) = 1 - Sim_{Sem}(x.l, p.l)$

$Cost_{Sem_Op} \nearrow$ when $Sim_{Sem} \searrow$

$Cost_{Sem_Op} \searrow$ when $Sim_{Sem} \nearrow$

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Integrating semantics in XML similarity

- Semantic similarity measure adopted: *Lin* [25]
- $Sim_{Sem}(C_1, C_2) = \dfrac{2 \log P(C)}{\log P(C_1) + \log P(C_2)} \in [0, 1]$

with C is the lowest common ancestor of C1 and C2 (maximizing their pair wise similarity value)

Plan :

Introduction
Background
Applications
Contributions
Conclusion

[25] Lin D., Am Information-Theoretic Definition of Similarity. *In Proceedings of the 15th International Conference on Machine Learning*, 296-304, Morgan Kaufmann Publishers Inc., 1998

Contributions

Integrating semantics in XML similarity

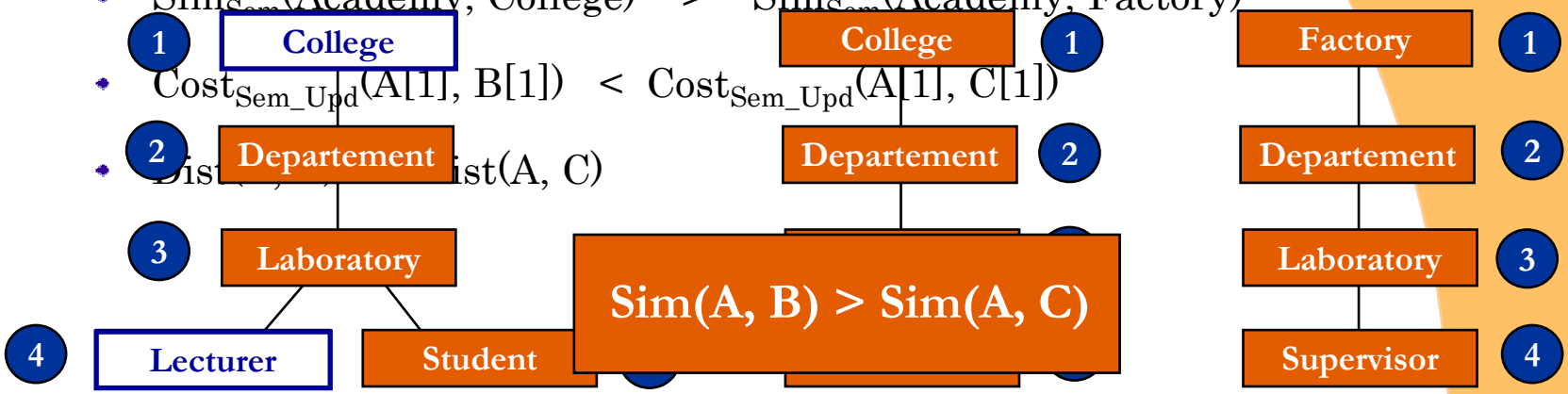
T2

- Example
  - $Cost_{Sem_Upd}(A[1], B[1]) = 1 - Sim_{Sem}(Academy, College)$
  - $Cost_{Sem_Upd}(A[1], C[1]) = 1 - Sim_{Sem}(Academy, Factory)$

Edit script = **Upd(A[1], B[1]), Upd(A[4], B[4]), Del(A[5], A[3])**

XML Document A **XML Document B** **XML Document C**

- $Sim_{Sem}(Academy, College) > Sim_{Sem}(Academy, Factory)$
- $Cost_{Sem_Upd}(A[1], B[1]) < Cost_{Sem_Upd}(A[1], C[1])$
- $Dist(A, B) < Dist(A, C)$



Plan :

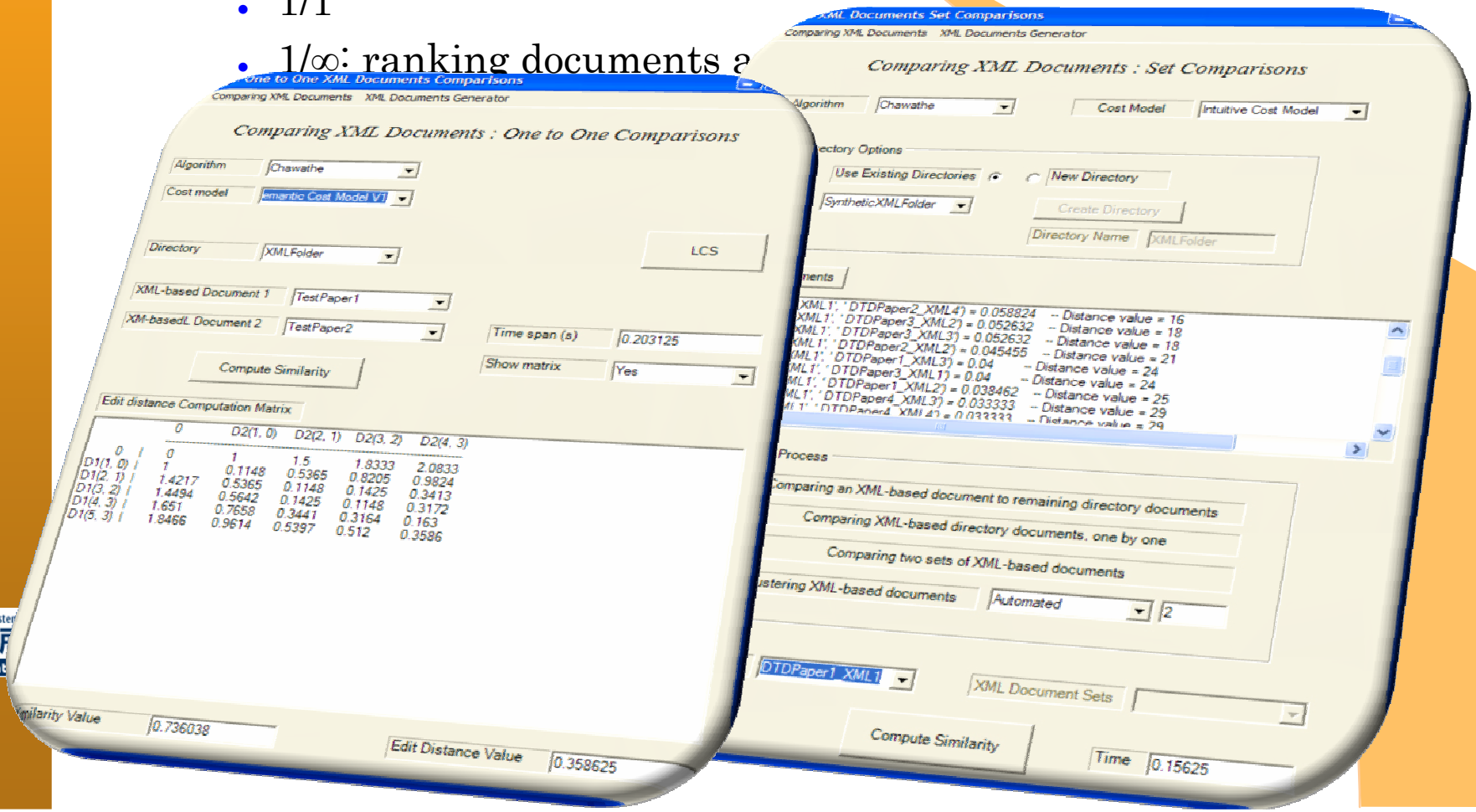
Introduction
Background
Applications
Contributions
Conclusion

T2

Tekli, 2006/06/06

Contributions

- Prototype *XS³* *(XML Structure and Semantic Similarity)*
  - XML documents comparison
    - 1/1
    - 1/∞: ranking documents a

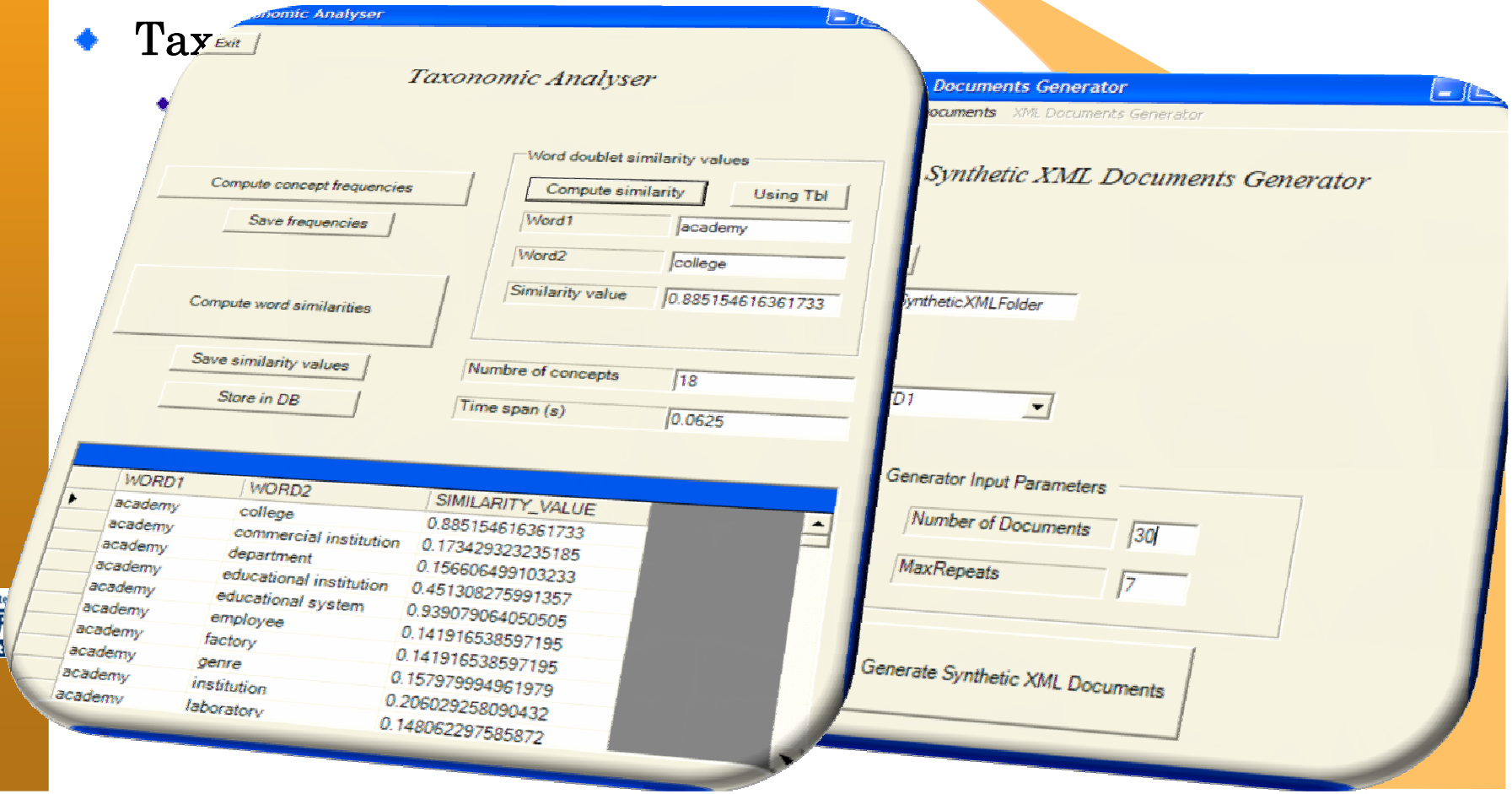


Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

- Synthetic XML documents generator
  - Producing sets of XML documents based on given DTDs
- Tax



Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

- Experimental results
  - Higher average similarity values, underlining similarities (of semantic nature) that were previously undetected
  - Straight distinction between documents corresponding to different DTDs
  - Capturing semantic affinities between document sets

```
<!DOCTYPE DTD1 [
<!ELEMENT Academy (Administrative unit+)>
    <!ELEMENT Administrative unit (Branch?)>
        <!ELEMENT Branch (Educator?, Student+)>
            <!ELEMENT Educator (#PCDATA)>
            <!ELEMENT Student (#PCDATA)>   ]>
```

```
<!DOCTYPE DTD2 [
<!ELEMENT School (Administrative unit+)>
    <!ELEMENT Administrative unit (Section?)>
        <!ELEMENT Section (Educator?, Scholar*)>
            <!ELEMENT Educator (#PCDATA)>
            <!ELEMENT Scholar (#PCDATA)>  ]>
```

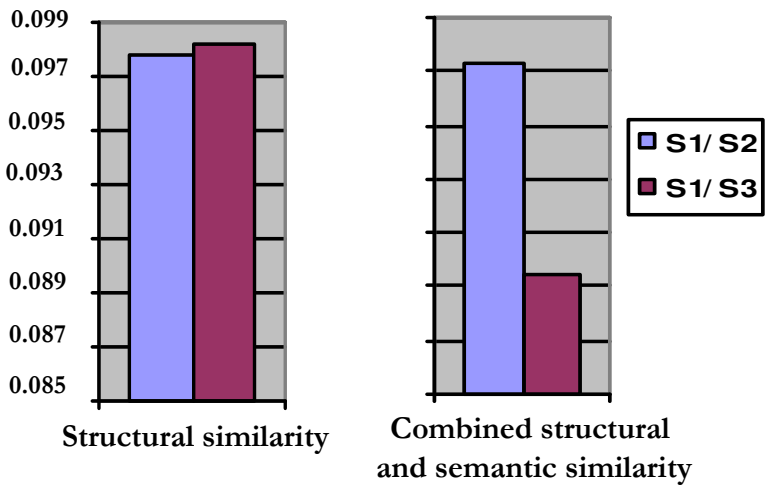
```
<!DOCTYPE DTD3 [
<!ELEMENT Government (Administrative unit+)>
    <!ELEMENT Administrative unit (Section?)>
        <!ELEMENT Section (Professional?, Worker+)>
            <!ELEMENT Professional (#PCDATA)>
            <!ELEMENT Worker (#PCDATA)>   ]>
```

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

- Experimental results
  - Higher average similarity values, underlining similarities (of semantic nature) that were previously undetected
  - Straight distinction between documents corresponding to different DTDs
  - Capturing semantic affinities between document sets



Plan :

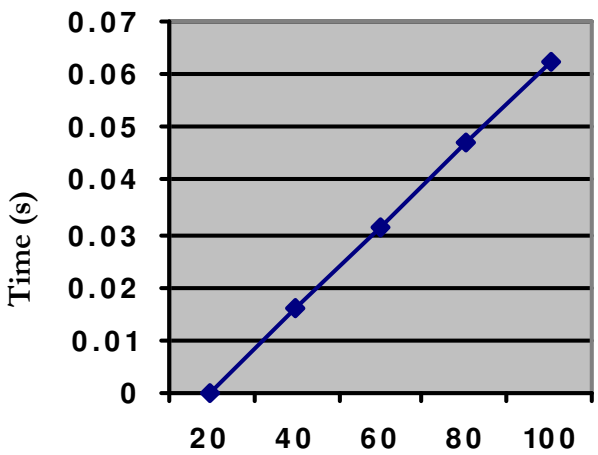
Introduction
Background
Applications
Contributions
Conclusion

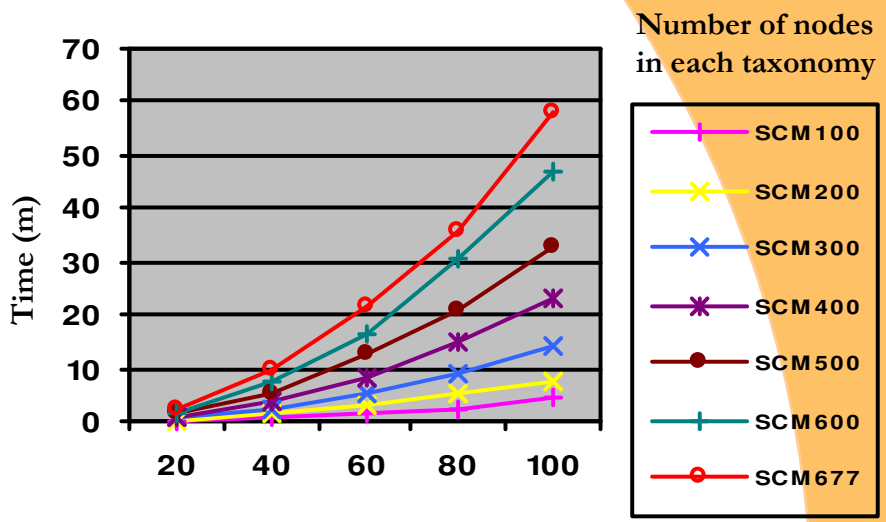
Contributions

- Experimental results

Our approach is of polynomial complexity

- Chawathe's classical *Edit Distance* process [4] being linear in the number of nodes of each tree $O(|A|\ |B|)$





Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Integrating semantics in XML similarity

- This is the first attempt to combine *Edit Distance* structural similarity computations with IR semantic similarity assessment, in an XML context
- Publications:
  - SOFSEM 2007
    - Tekli J., Chbeir R., Yetongnon K., **A Hybrid Approach for XML Similarity**. In *Proc. of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science*, Springer LNCS, Czech Republic, 2007.
  - COMAD 2006
    - Tekli J., Chbeir R., Yetongnon K., **Semantic and Structure based XML Similarity: An Integrated Approach**. In *Proc. of the 13th International Conference on Management of Data (COMAD)*, New Delhi, India, 2006.

Plan :

Introduction
Background
Applications
Contributions
Conclusion

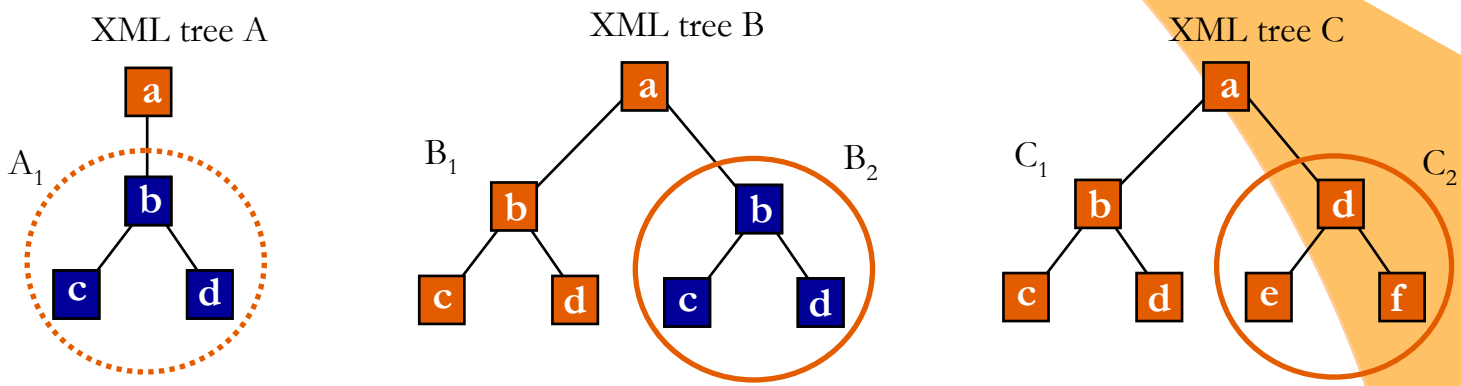
Overview

- Introduction
- Background
- Applications
- Our contributions
  - Integrating semantics in XML similarity
  - Enhancing XML structural similarity
- Conclusion

Contributions

Enhancing XML structural similarity

- We identified certain cases where existing edit distance approaches yield inaccurate similarity results:
  - Similarity between sub-trees A_1 and B_1



Following Chawathe [4], Dist(A, B) = Dist(A, C) = 3

An extension of Chawathe [4], provided by Nierman and Jagadish [2], is able to detect such sub-tree similarities, i.e. only when the *containment* relation is fulfilled

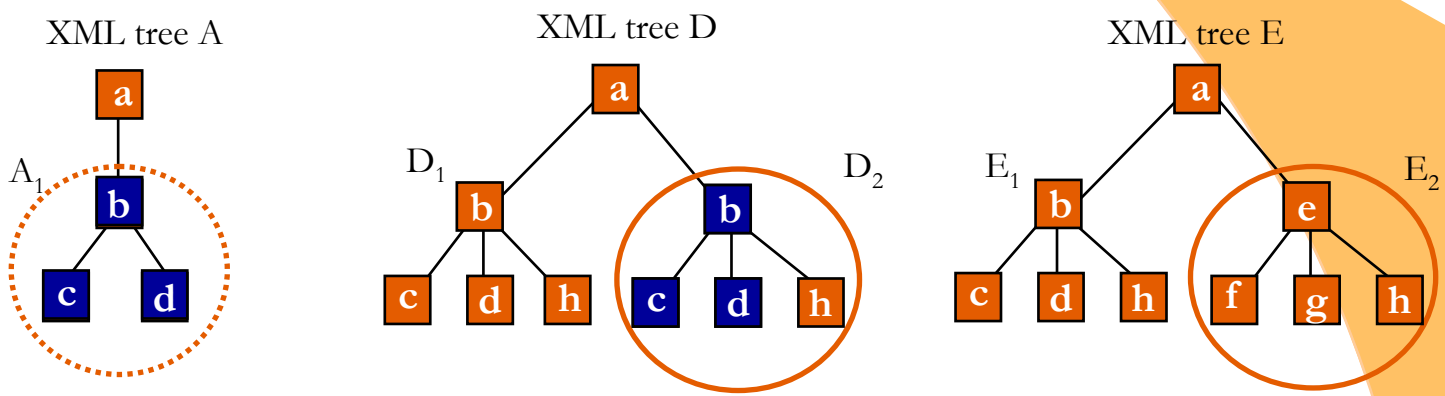
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Enhancing XML structural similarity

- We identified certain cases where the existing edit distance approaches yield inaccurate similarity results:
  - Similarity between XML trees A/D (sub trees A_1 and D_2) w.r.t. A/E



Following Chawathe [4], Dist(A, D) = Dist(A, E) = 5

The containment relation is not fulfilled

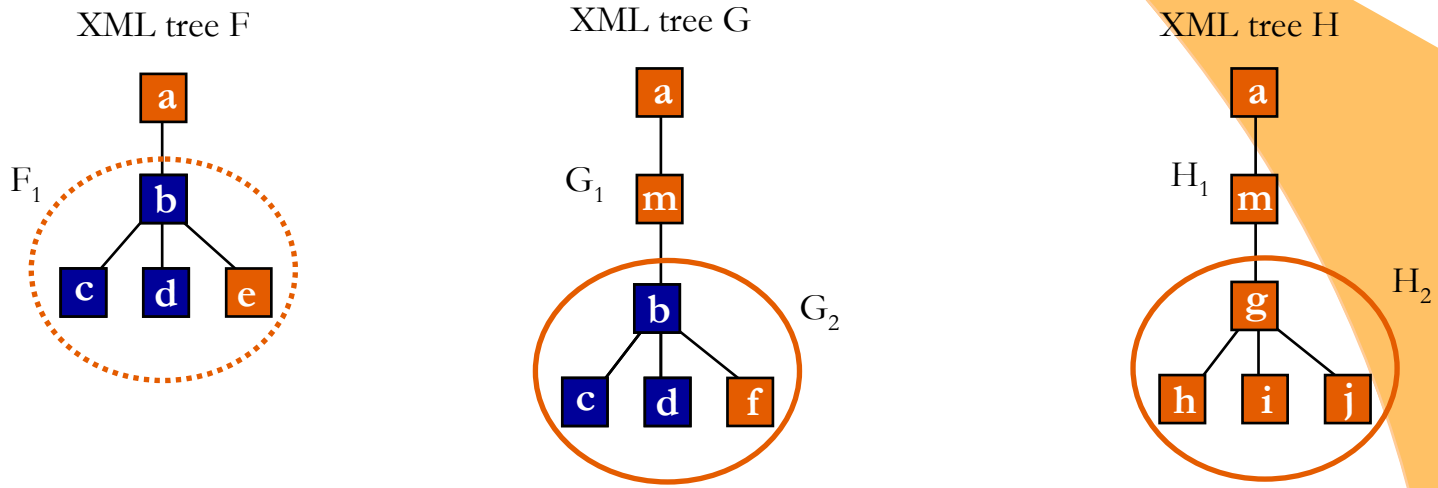
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Enhancing XML structural similarity

- We identified certain cases where the existing edit distance approaches yield inaccurate similarity results:
  - Similarity between XML trees F and G (sub-trees F_1 and G_2) w.r.t. F and H



Following Chawathe [4], Dist(F, G) = Dist(F, H) = 7

The containment relation is not fulfilled and the sub-trees sharing structural similarities occur and different depths

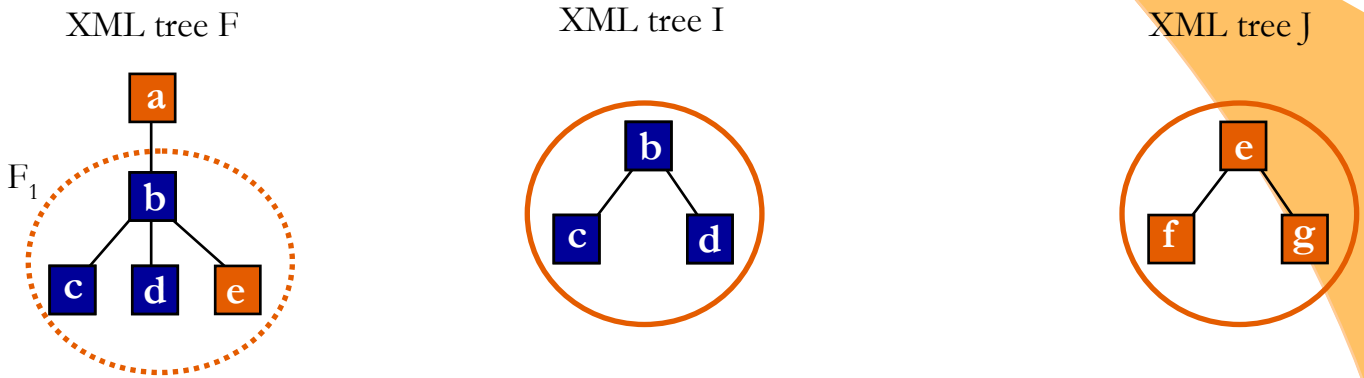
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Enhancing XML structural similarity

- We identified certain cases where the existing edit distance approaches yield inaccurate similarity results:
  - Similarity between XML trees F and I (sub-tree F_1 and tree I) w.r.t. F and J



Following Chawathe [4], Dist(F, I) = Dist(F, J) = 6

The containment relation is not fulfilled and structural similarities occur, not only among sub-trees, but also at the sub-tree/tree level

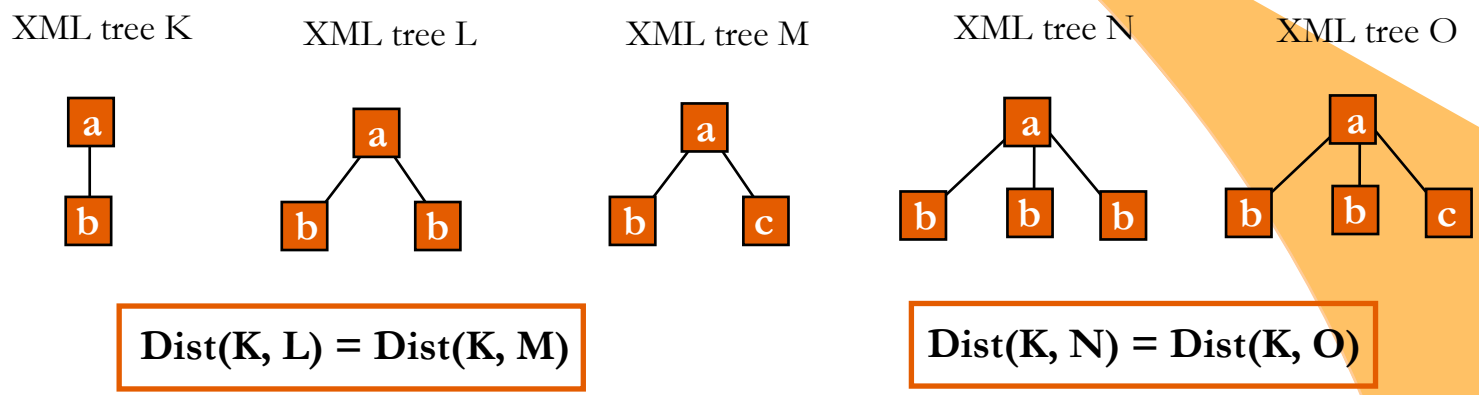
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Enhancing XML structural similarity

- In addition, current XML structural similarity approaches overlook the special case of leaf node repetitions:



- We explicitly mention the case of leaf node since:
  - Leaf nodes are a special kind of sub trees: *single node sub-trees*
  - Leaf node repetitions are as frequent as substructure repetitions in XML documents
  - Detecting leaf node repetitions is spontaneous in the XML context, and would help increase the discriminative power of XML comparison methods

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

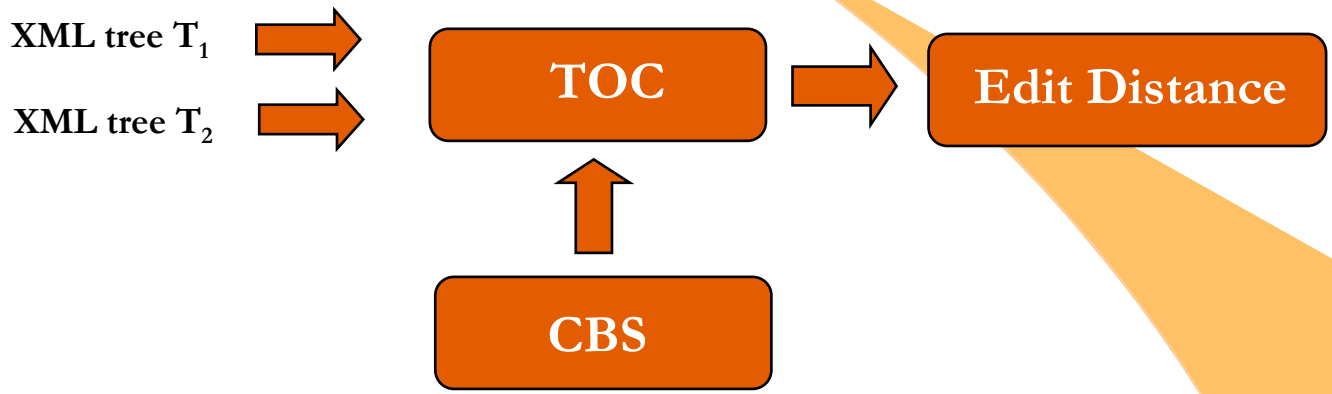
- Our approach
  - is able to provide an improved method for comparing heterogeneous XML documents
  - takes into account sub-tree structural commonalities while comparing XML trees

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

- Overview of our structural similarity approach:



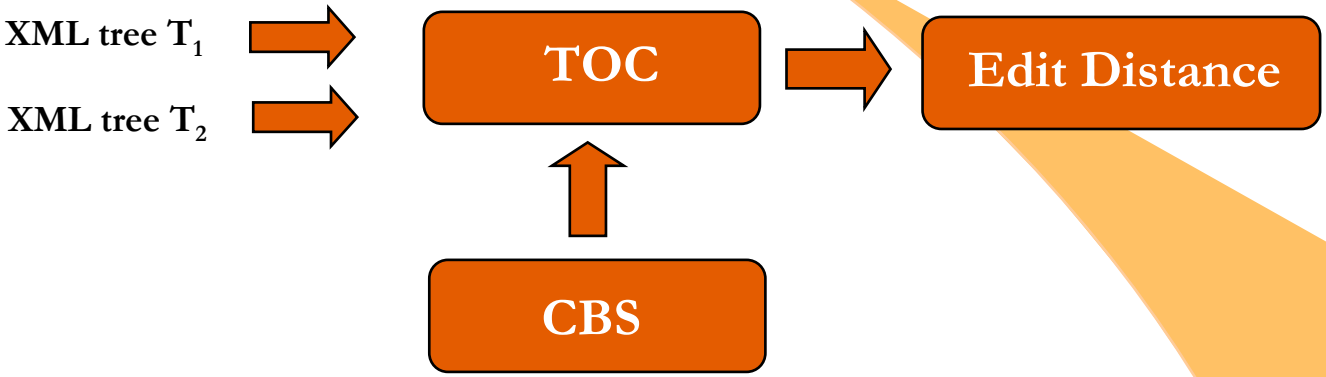
- *CBS*
  - an algorithm for identifying the *Commonality Between Sub-trees*

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

- Overview of our structural similarity approach:



- *TOC*
  - an algorithm for computing the *Tree edit distance Operations Costs*
  - makes use of *CBS*, its results being exploited via [2]'s main edit distance algorithm (Nierman and Jagadish), so as to identify the structural similarity between two XML documents

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Commonality Between Sub-trees (*CBS*)

- Given two sub-trees $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$, the structural commonality between A and B, designated by $ComSubTree(A, B)$, is a set of nodes $N = \{n_1, \ldots, n_p\}$ such that $ni \in N$, ni occurs in A and B with the same label, depth and relative node order (in preorder traversal ranking) as in A and B.
- **Formal definition** : For $1 \leq i \leq p$; $1 \leq r \leq m$; $1 \leq u \leq n$:
  - $n_i.l = a_r.l = b_u.l$
  - $n_i.d = a_r.d = b_u.d$
  - For any $n_j \in N / i \leq j$, $as \in A$ and $bv \in B$ such as:
    - $n_j.l = a_s.l = b_v.l$
    - $n_j.d = a_s.d = b_v.d$
    - $r \leq s$, $u \leq v$
  - There is no set of nodes N' that satisfies conditions *1*, *2* and *3* and is of larger cardinality than N.

Plan :

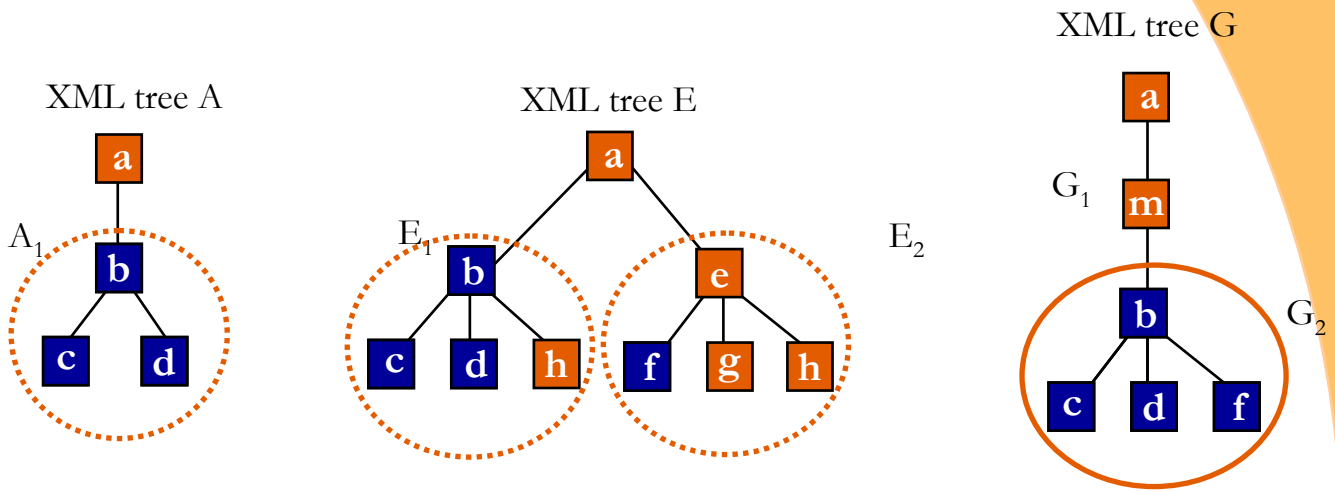
Introduction
Background
Applications
Contributions
Conclusion

Contributions

Commonality Between Sub-trees (*CBS*)

- In other words, the problem of finding the structural commonality between two sub-trees T_i and T_j is equivalent to finding the maximum number of matching nodes in T_i and T_j

- For example:
  - *ComSubTree*(A_1, E_1) = 3 (3 structurally matching nodes), *ComSubTree*(A_1, E_2) = 0
  - *ComSubTree*(E_1, G_2) = 3 , *ComSubTree*(E_2, G_2) = 1
  - …



Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Commonality Between Sub-trees (*CBS*)

- In other words, the problem of finding the structural commonality between two sub-trees T_i and T_j is equivalent to finding the maximum number of matching nodes in T_i and T_j
  - On the other hand, the problem of finding the shortest edit distance between T_i and T_j comes down to identifying the minimal number of edit operations that can transform T_i to T_j.
  - Those are dual problems since identifying the shortest edit distance between two sub-trees (trees) underscores, in a roundabout way, their maximum number of matching nodes.

- Therefore, our algorithm (*CBS*), for identifying the structural commonality between sub-trees, is based on the edit distance concept

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Commonality Between Sub-trees (*CBS*)

```
Algorithm CBS()
Input: Sub-trees SbT_i and SbT_j (in Id-pair representations)
Output: |ComSubTree(SbT_i, SbT_j)|

Begin
  Dist [][] = new [0...|SbT_i|][0...|SbT_j|]
  Dist[0][0] = 0

  For (n = 1 ; n ≤ |SbT_i| ; n++)
      { Dist[n][0] = Dist[n-1][0] + Cost_Del(SbT_i[n]) }
  For (m = 1 ; m ≤ |SbT_j| ; m++)
      { Dist[0][m] = Dist[0][m-1] + Cost_Ins(SbT_j[m]) }

  For (n = 1 ; n ≤ |SbT_i| ; n++)
  {
     For (m = 1 ; m ≤ |SbT_j| ; m++)
     {
        Dist[n][m] = min{
              If (SbT_i[n].d = SbT_j[m].d & SbT_i[n].l = SbT_j[m].l)
                  { Dist[n-1][m-1] },
              Dist[n-1][m] + Cost_Del(SbT_i[n]),        // simplified node
              Dist[n][m-1] + Cost_Ins(SbT_j[m])         // operations syntaxes.
                        }
     }
  }

  Return (|SbT_i|+|SbT_j| - Dist[|SbT_i|][|SbT_j|]) / 2        // |ComSubTree(SbT_i, SbT_j)|
End
```

CBS can be equally applied on whole trees. Nonetheless, in our approach, its use is couples with sub-trees

Returns the number of structurally matching nodes between two sub-trees

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Commonality Between Sub-trees (*CBS*)

- $|ComSubTree(SbT_i, SbT_j)|$, is identified w.r.t. the minimum edit distance:
  - Total number of deletions - we delete all nodes of SbT_i except those having matching nodes in SbT_j:
    - $|SbT_i| - |ComSubTree(SbT_i, SbT_j)|$
  - Total number of insertions - we insert into SbT_i all nodes of SbT_j except those having matching nodes in $SbTi$:
    - $|SbT_j| - |ComSubTree(SbT_i, SbT_j)|$

- Following *CBS*, the edit distance between sub-trees SbT_i and SbT_j becomes as follows:
  - $Dist[|SbT_i|][|SbT_j|] = \Sigma_{Deletions} \times 1 + \Sigma_{Insertions} \times 1$
    $= |SbT_i| + |SbT_j| - 2\ |ComSubTree(SbT_i, SbT_j)|$

$$|ComSubTree(SbT_i, SbT_j)| = \frac{|SbT_i| + |SbT_j| - Dist[|SbT_i|][|SbT_j|]}{2}$$

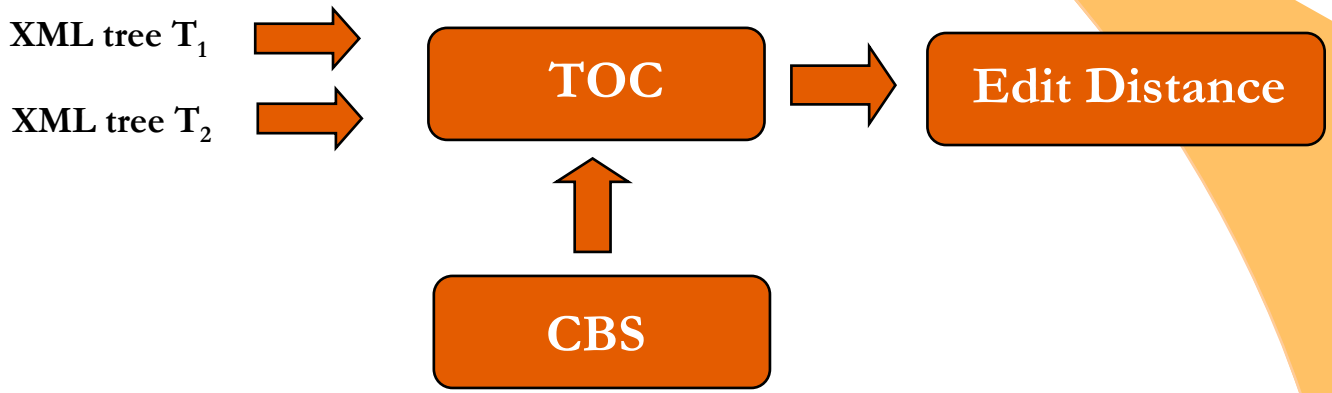
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

- The *CBS* algorithm, for the identification of the *commonality* between sub-trees, is to be utilized in *TOC*: an algorithm dedicated to computing the tree edit distance operations costs



- Consequently, those costs will be exploited via [2]'s main edit distance approach (Nierman and Jagadish) providing an improved and more accurate XML structural similarity measure

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

- Using *CBS*, *TOC* identifies the structural *commonality* between each and every pair of sub-trees (SbT_i, SbT_j) in the two trees *A* and *B* being compared, as well as their commonalities with the whole trees A and B.
- Consequently, those values are normalized via corresponding tree/sub-tree cardinalities $Max(|SbT_i|, |SbT_j|)$, to be comprised between 0 and 1 :
  - $$\frac{CBS(SBT_i, SBT_j)}{Max(|SBT_i|, |SBT_j|)} = 0$$ when there's no commonality between SBT_i and SBT_j : $CBS(SBT_i, SBT_j) = 0$
  - $$\frac{CBS(SBT_i, SBT_j)}{Max(|SBT_i|, |SBT_j|)} = 1$$ when SBT_i and SBT_j are identical $CBS(SBT_i, SBT_j) = |SBT_i| = |SBT_j|$

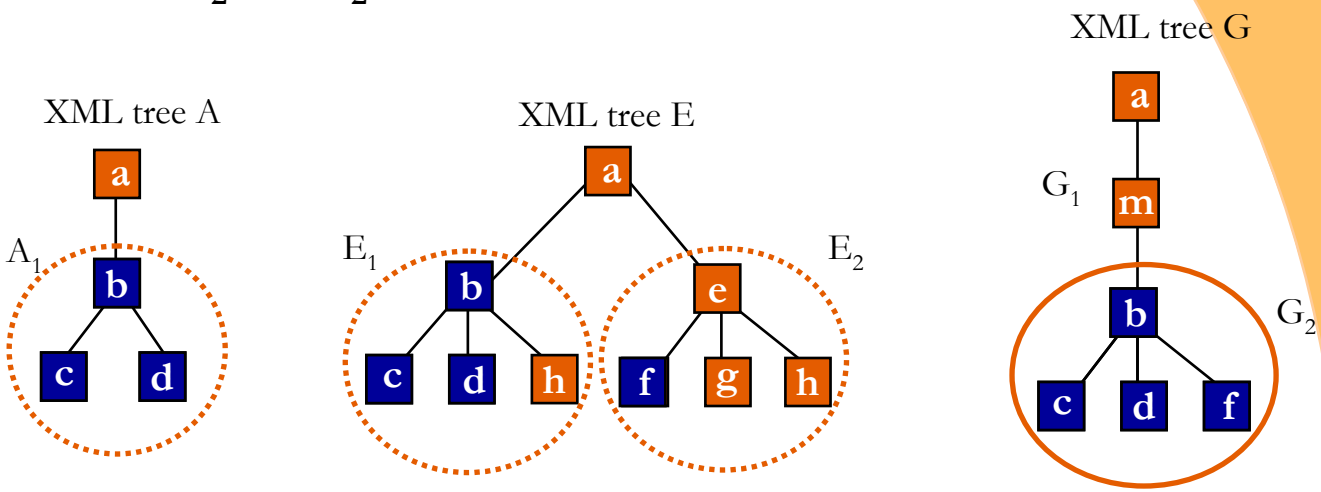
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

- For example:
  - $$\frac{CBS(A_1, E_1)}{Max(|A_1|, |E_1|)} = \frac{3}{4} = 0.75$$
  - $$\frac{CBS(E_2, G_2)}{Max(|E_2|, |G_2|)} = \frac{1}{4} = 0.25$$



Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

- Consequently, tree insert/delete operations cos
normalized commonality between the concerne

Algorithm TOC()

Input: Trees T_1 and T_2

Output: insert tree and delete tree operations costs

Begin

For each sub-tree SbT_i in T_1 // Going through // all sub-trees in T_1.

{

$$Cost_{DelTree}(SbT_i) = \sum_{\text{All nodes } x \text{ of } SbT_i} Cost_{Del}(x)$$

For each sub-tree SbT_j in T_2 // Going through // all sub-trees in T_2.

{

$$Cost_{InsTree}(SbT_j) = \sum_{\text{All nodes } x \text{ of } SbT_j} Cost_{Ins}(x)$$

$$Cost_{DelTree}(SbT_i) = \text{Min}\{ Cost_{DelTree}(SbT_i), \sum_{\text{All nodes } x \text{ of } SbT_i} Cost_{Del}(x) \times \frac{1}{1 + \frac{CBS(SbT_i, SbT_j)}{\text{Max}(|SbT_i|, |SbT_j|)}} \}$$

$$Cost_{InsTree}(SbT_j) = \text{Min}\{ Cost_{InsTree}(SbT_j), \sum_{\text{All nodes } x \text{ of } SbT_j} Cost_{Ins}(x) \times \frac{1}{1 + \frac{CBS(SbT_i, SbT_j)}{\text{Max}(|SbT_i|, |SbT_j|)}} \}$$

}

}

Usually, the cost of deleting a tree is equal to the sum of the costs of deleting its nodes

Tree operations costs vary w.r.t. the normalized commonality between the sub-trees in the XML trees being compared

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Research Organization of Information and Systems
国立情報学研究所
National Institute of Informatics

Contributions

Tree edit distance Operations Costs (*TOC*)

- Tree operations costs vary as follows:
  - **Max Cost:** $Cost_{InsTree/DelTree}(SbT_i) = \sum Cost_{Ins/Del}(x) \times \frac{1}{1+0}$
    - Sum of the costs of inserting/deleting the sub-tree nodes

Minimum normalized commonality

  - **Min Cost:** $Cost_{InsTree/DelTree}(SbT_i) = \sum Cost_{Ins/Del}(x) \times \frac{1}{1+1}$
    - Half its maximum cost

Maximum normalized commonality

We assign tree operation costs in such a way so as to guaranty that the cost of inserting/deleting a non-leaf node sub-tree will never be less than the cost of inserting/deleting a single node.

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

- In fact, *TOC* is based on the intuition that tree operations are more costly than node operations.

- *Proof:*
  - *The smallest non-leaf node sub-tree that can be treated via a tree operation is a sub-tree consisting of two nodes.*
  - *For such a tree, the maximum tree operation cost is equal to 2*
  - *The minimum tree operation cost is equal to 1:*
    - *Equivalent to the cost of inserting/deleting a single node,*
    - ***That is the lowest tree operation cost attainable, for a non-leaf node sub-tree following TOC.***

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

The special case of leaf node sub-trees:

- The maximum cost for inserting/deleting a single node sub-tree is 1
  - The cost of inserting/deleting the single node at hand
    - $Cost_{InsTree/DelTree}(SbT_i) = Cost_{Ins/Del}(x) \times 1 = 1$

- The minimum cost for deleting a single node sub-tree is equal to $\frac{1}{2}$
  - Half its maximum insert/delete cost
    - $Cost_{InsTree/DelTree}(SbT_i) = Cost_{Ins/Del}(x) \times \frac{1}{2} = \frac{1}{2}$

Following the intuition that tree operations are more costly than node operations.

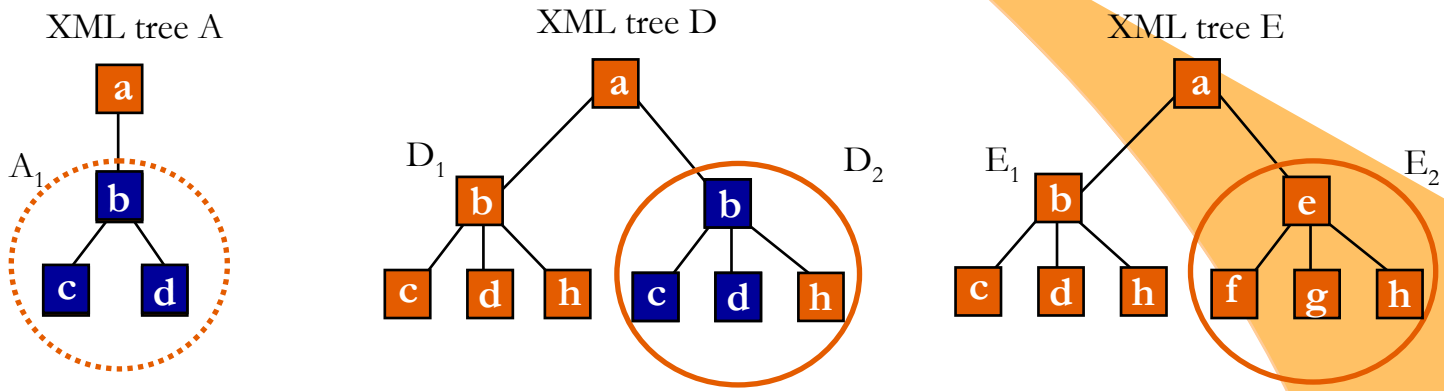
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

- Computation examples: None leaf-node sub-trees



Following Chawathe [4], Dist(A, D) = Dist(A, E) = 5

- Cost $_{InsTree}(D_2) = 4 \times \dfrac{1}{1 + \dfrac{3}{4}} = 2.2856$

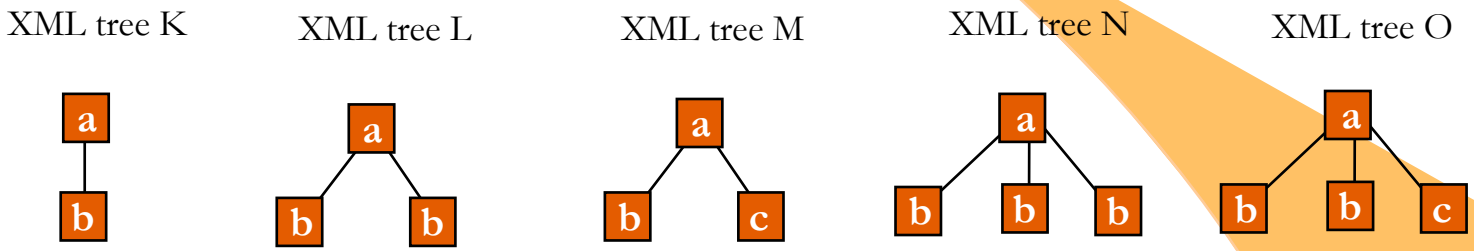
Insertion of node *h*

- Following our approach, **Dist(A, D) = 1 + 2.2856 = 3.2856** $\neq$ Dist(A, E)=5

Contributions

Tree edit distance Operations Costs (*TOC*)

- Computation examples: Leaf node sub-trees



Dist(K, L) = Dist(K, M) = 1

Dist(K, N) = Dist(K, O) = 2

- $\text{Cost}_{\text{InsTree}}(b) = 1 \times \dfrac{1}{1 + \dfrac{1}{1}} = 0.5$
- Following our approach, **Dist(K, L) = 0.5** $\neq$ Dist(K, M) =1
- Following our approach, **Dist(K, N) = 0.5 + 0.5 = 1** $\neq$ Dist(K, O) =2

Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

Tree edit distance Operations Costs (*TOC*)

- Computation examples: Comparing results with current approaches

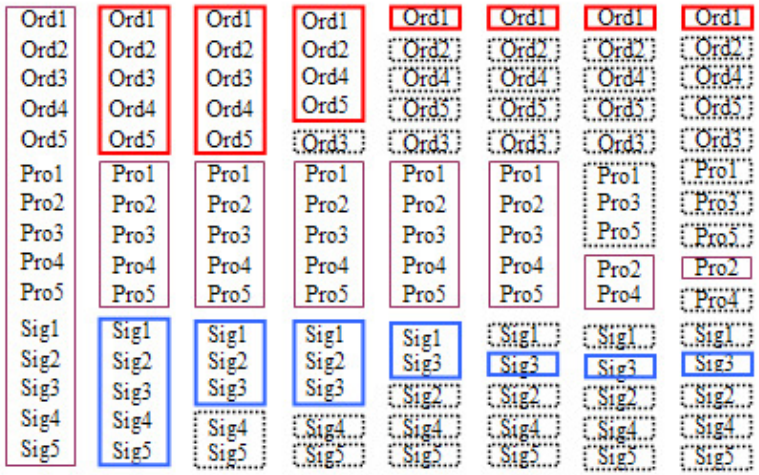
	Our Approach		N. & J. [13]	Dalamagas *et al.* [6]	Chawathe [4]
	Distance	*Similarity*			
A/B	1.5	0.4	*Detected*	*Not detected*	*Not detected*
A/C	3	0.25			
A/D	3.2856	0.2333	*Not detected*	*Not detected*	*Not detected*
A/E	5	0.1667			
F/G	5.4106	0.1560	*Not detected*	*Not detected*	*Not detected*
F/H	7	0.125			
F/I	4.2857	0.1892	*Not detected*	*Not detected*	*Not detected*
F/J	6	0.1429			
K/L	0.5	0.6667	*Not detected*	*Not detected*	*Not detected*
K/M	1	0.5			
K/N	1	0.5	*Not detected*	*Not detected*	*Not detected*
K/O	1.5	0.4			

Plan :

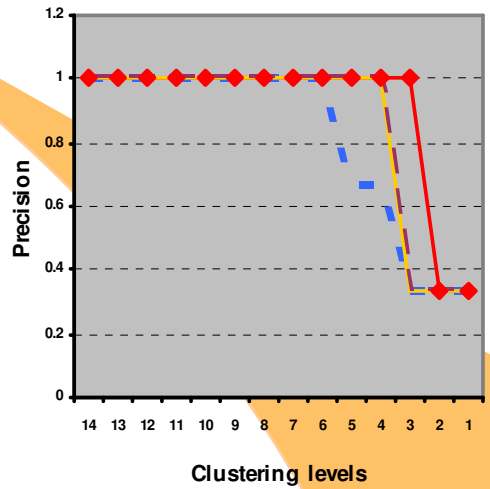
Introduction
Background
Applications
Contributions
Conclusion

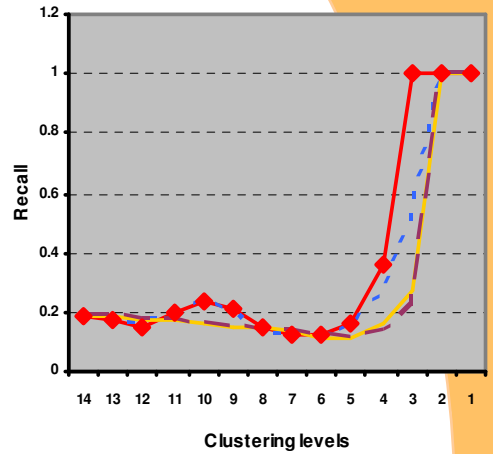
Contributions

- Experiments are based on structural clustering



Level:	1 & 2	3 & 4	5	6 & 7	8	9	10	…, 14
∑a =	5	15	13	12	8	7	4	3
∑b =	10	0	0	0	0	0	0	0
∑c =	0	0	2	3	7	8	11	12
PR =	0.3333	1	1	1	1	1	1	1
R =	1	1	0.8667	0.8	0.5333	0.4667	0.2667	0.2





Plan :

Introduction
Background
Applications
Contributions
Conclusion

Research Organization of Information and Systems
国立情報学研究所
National Institute of Informatics

Contributions

An improved XML structural similarity approach

- Experiments are based on structural clustering
  - Evaluation on a real data set:
    - Sigmod Record
      - OrdinaryIssuePage.dtd
      - ProceedingsPage.dtd
      - SigmodRecord.dtd
  - Evaluation on a synthetic data set:
    - Two sets of 600 documents each based on real word and synthetic DTDs
      - From http://www.xmlfiles.com and http://www.w3schools.com
    - The first set was created with *MaxRepeats = 5*
    - The second with *MaxRepeats = 10*, the latter set underlining XML documents with greater size and variability w.r.t. the former

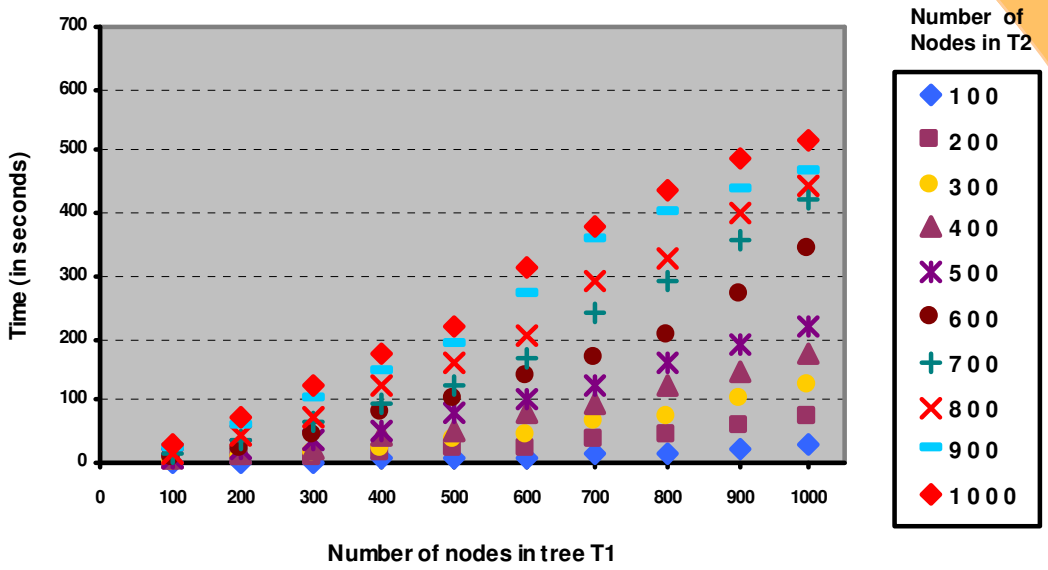
Plan :

Introduction
Background
Applications
Contributions
Conclusion

Contributions

An improved XML structural similarity approach

- Timing analysis
  - Our approach is linear in the number of nodes of each tree $\mathcal{O}(|A|\ |B|)$



Plan :

Introduction
Background
Applications
Contributions
Conclusion

Research Organization of Information and Systems
国立情報学研究所
National Institute of Informatics

Overview

- Introduction
- Background
- Applications
- Our contributions
- Conclusion

Conclusion

- In the past few years, XML has been established as the *de facto* standard format for web publishing [26]
  - Attracting growing attention in information retrieval, database as well as multimedia related research

- As a result, XML similarity becomes a central issue, especially in:
  - Data warehousing
  - Data integration
  - Classification and clustering
  - Information search and retrieval

Plan :

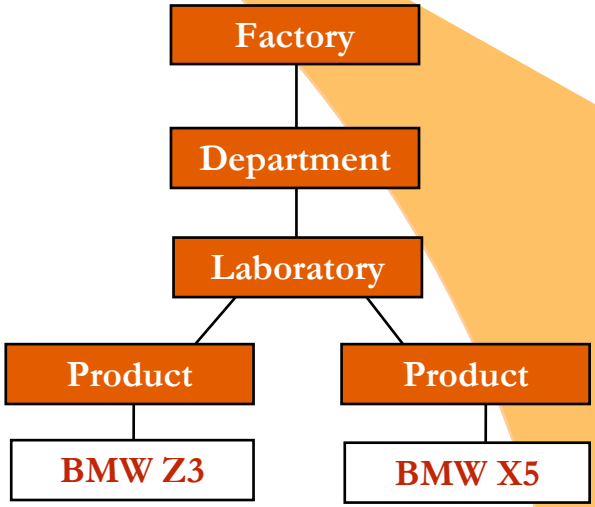
Introduction
Background
Applications
Contributions
Conclusion

[26] Wang Y., DeWitt D.J. and Cai J.Y., X-Diff: An Effective Change Detection Algorithm for XML Documents. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, p. 519-530, 2003.

Conclusion

- Future work
  - Exploiting semantic similarity to compare, not only the structure of XML documents, but also their information content (values)

```
<Factory>
        <Department>
            <Laboratory>
                <Product> BMW Z3 </Product>
                <Product> BMW X5 </Product>
            </Laboratory>
        </Department>
</Factory>
```



Plan :

Introduction
Background
Applications
Contributions
Conclusion

ありがとう

質問 …

Richard.chbeir@u-bourgogne.fr

Some references

[1] Tversky A., Features of Similarity. *Psychological Review*, 84(4):327-352, 1977.

1.[2] Nierman A. and Jagadish H. V., Evaluating structural similarity in XML documents.
2.In *Proceedings of the 5th International Workshop on the Web and Databases,* 2002.

[3] Cobéna G., Abiteboul S. and Marian A., Detecting Changes in XML Documents.
In *Proc. of the IEEE Int. Conf. on Data Engineering*, 2002, 41-52.

[4] Chawathe S., Comparing Hierarchical Data in External Memory. In *VLDB* , 1999, 90-101.

[5] Shasha D. and Zhang K., Approximate Tree Pattern Matching. In *Pattern Matching in Strings, Trees and Arrays*, chapter 14, Oxford University Press, 1995.

[6] Buttler D., A Short Survey of Document Structure Similarity Algorithms.
In *Proceedings of the 5th International Conference on internet Computing*, Las Vegas, USA, 2004.

[7] Rafiei D., Moise D. and Sun D., Finding Syntactic Similarities between XML Documents.
In *Proceedings of the 17th International Conference Database and Expert Systems Applications, 2006. DEXA '06*

[8] Flesca S., Manco G., Masciari E., Pontieri L., and Pugliese A., Detecting Structural Similarities Between XML Documents. In *Proceedings of the 5th International Workshop on The Web and Databases (WebDB 2002)*, 2002.

[9] Grabs T., Schek H.-J., Generating Vector Spaces On-the-fly for Flexible XML Retrieval.
In *Proceedings of SIGIR'2002 Workshop on XML and information Retrieval*, p.4-13, Finland, 2002.

[10] Schlieder T. and Meuss H., Querying and Ranking XML Documents. *JASIS Spec. Top. XML/IR* 53(6):489-503, 2002.

[11] Fuhr N. and Großjohann K., XIRQL: A Query Language for Information Retrieval.
In *Proceedings of ACM-SIGIR*, New Orleans, 2001, pp. 172-180.

[12] Carmel D., Efraty N., Landau G.M., Maarek Y.S. and Mass Y., An Extension of the Vector Space Mpdel for Querying XML Documents via XML Fragments. In *Proceedings of SIGIR' 2002 Workshop on XML and Information Retrieval*, p. 14-25, Finland, 2002.

Some references

[13] Schlieder T. and Meuss H., Querying and Ranking XML Documents. *JASIS Spec. Top. XML/IR* 53(6):489-503, 2002

[14] Pokorny, J., Rejlek, V.: A Matrix model for XML Data. Chap. in: Databases and Information Systems, *6th International Baltic Conference DB&IS´2004, V. 118 Frontiers in Artificial Intelligence and Applications*, IOS Press, 2005, pp. 53-64.

[15] Guha S., Jagadish H.V., Koudas N., Srivastava D. and Yu T., Approximate XML Joins. I n *Proceedings of ACM SIGMOD 2002*, pp. 287-298 (2002).

[16] Bertino E., Guerrini G., Mesiti M., A Matching Algorithm for Measuring the Structural Similarity between an XML Documents and a DTD and its Applications, *Elsevier Computer Science, 29*, 2004, 23-46.

[17] Schöning H., Tamoni – A DBMS Designed for XML. In *Proceedings of the ICDE Conference*, p. 149-154, 2001.

[18] Deutsch A., Fernandez M., Florescu D, Levy A. and Suciu D., XML-QL: A Query Language for XML. In *Proceedings of the 8th International World Wide Web Conference*, 1999.

[19] Robie J., XQL (XML Query Language), August 1999. http://metalab.unc.edu/xql/xql-proposal.xml.

[20] Chamberlin D., Florescu D., Robie J., Simeon J. and Stefanescu M., XQuery : A Query Language for XML, 2001. http://www.w3.org/TR/2001/WD-xquery-20010215.

[21] Chinenyanga T.T. and Kushmerick N., An Expressive and Efficient Language for XML Information Retrieval. In *Proceedings of the 2001 SIGIR Conference*, 2001.

[22] Fuhr N. and Großjohann K., XIRQL: A Query Language for Information Retrieval. In: *Proceedings of ACM-SIGIR*, New Orleans, 2001, pp. 172-180.

[23] Schlieder T., Similarity Search in XML Data Using Cost-based Query Transformations. In *Proceedings of SIGMOD WebDB Workshop*, 2001.

[24] Bremer J.-M. and Gertz M., XQuery/IR: Integrating XML Document and Data Retrieval. In *Proceedings of the 5th International Workshop on the Web and Databases (WebDB)*, June 2002.

Research Organization of Information and Systems
国立情報学研究所
National Institute of Informatics