# TREEDIFF: A System for Document Comparison by Structure

Jason Tsong-Li Wang (NJIT)

George J. S. Chang (NJIT)

Girish Patel (NJIT)

Liam Relihan (Piercom Ltd., Ireland)

Dennis Shasha (Courant Institute, NYU)

Kaizhong Zhang (Univ. Western Ontario)

Contact authors:

Wang, jason@village.njit.edu

http://www.cis.njit.edu/~jason

Shasha, shasha@cs.nyu.edu

http://cs.nyu.edu/cs/faculty/shasha/index.html



## Motivation and Operations

We provide a library of functions, listed below, for comparing structured documents (currently in SGML or HTML). They can help co-authors see at a glance what changes have occurred, can help detect approximate plagiarism, and can help unify documents.

In addition, we provide a query language.

- **symdifference**(A, B): to find all the differences between A and B (the structural elements in A not in B, the structural elements in B not in A, and the structural elements changed from A to B).
- **unidifference**(A, B): to find the structural elements in A not in B and the structural elements changed from A to B.
- **union**(A, B): to find the structural elements in either document: (structural elements in common between A and B, the structural elements in A not in B, the structural elements in B not in A, and the structural elements changed from A to B).
- **consensus**(A, B): to find the structural elements in common between A and B.



## Substructure Operations

- **substructure comparison**(A, B): to determine whether or not A approximately occurs as a subtree of B; it returns the closest matching subtree of B and the distance between that subtree and A.
- **superstructure comparison**(A, B): to determine whether or not A contains B as an approximate subtree; it returns the closest matching subtree of A and the distance between that subtree and B.



## Query Languages and a Function Library

- These operations can be called from a query language. We have done a first cut at this. For example, the following query does the join between two documents. It finds the symdifference between document A and document B, where document A is in docset1 and document B is in docset2.

```
SELECT   S
FROM     docset1 A, docset2 B
WHERE    S = SYMDIFFERENCE(A, B)
```

- We can make the tree comparison programs available as a library for research purposes and for collaboration. The programs that we can provide are independent of SGML, so users can use them for other document languages.

## Approach for SGML

- SGML provides sets of rules that define the allowable contents of document elements. These rules are contained in Document Type Definitions (DTDs).
- We represent structured documents (e.g. SGML) by ordered labeled trees based on the underlying DTDs.
- We compare these documents using various tree matching algorithms. The matching algorithms are independent of the source language, so the same could be used for C or SGML.



## A DTD for Article

```
 1.  <!DOCTYPE   article [
 2.  <!ELEMENT   article    - -      (title,author+
                                     ,affil,abstract
                                     ,section+,
                                     acknowl)>
 3.  <!ATTLIST   article             status (final|
                                     draft) draft
                                     #REQUIRED>
 4.  <!ELEMENT   title      - -      (#PCDATA)>
 5.  <!ELEMENT   author     - O      (#PCDATA)>
 6.  <!ELEMENT   abstract   - O      (#PCDATA)>
 7.  <!ELEMENT   section    - O      ((title,body+) |
                                     (title,body*,
                                     subsectn+))>
 8.  <!ELEMENT   subsectn   - O      (title,body+)>
 9.  <!ELEMENT   body       - O      (figure | paragr)>
10.  <!ELEMENT   figure     - O      (picture,
                                     caption?)>
11.  <!ATTLIST   picture             sizex NMTOKEN
                                     “16cm”
                                     sizey NMTOKEN
                                     #IMPLIED
                                     file ENTITY
                                     #IMPLIED>
12.  <!ELEMENT   caption    O O      (#PCDATA)>
13.  <!ENTITY    file       SYSTEM   “/u/chang/TEX
                                     /SGML/fig1.ps”
                                     NDATA>
14.  <!ELEMENT   paragr     - O      (#PCDATA)>
15.  <!ELEMENT   acknowl    - O      (#PCDATA)>
                                     ]>
```



## An SGML article (doc1)

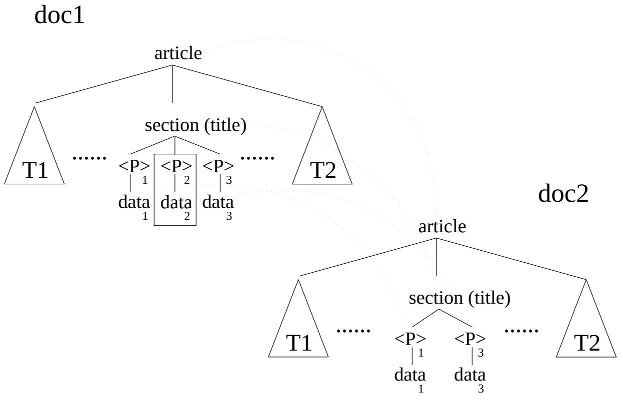
```
<article status=“final”>
<title> Structural Matching and Discovery
in Document Databases
<author> Jason Tsong-Li Wang
<author> Dennis Shasha
<author> George J. S. Chang
...
<abstract> Structural matching and discovery in
documents such as SGML and HTML is important
for data warehousing, version management, ...
<section>
<title> Introduction <\title>
...
<paragr> Advances in hardware and software have
stimulated corresponding advances in document
presentation, in the management and retrieval of
...
<paragr> In particular, the advent of hypertext and
hypermedia has greatly reduced the emphasis on the
...
<paragr> Likewise, the ability to manage ...
...
<\article>
```



## Another SGML article (doc2)

```
<article status=“final”>
<title> Structural Matching and Discovery
in Document Databases
<author> Jason Tsong-Li Wang
<author> Dennis Shasha
<author> George J. S. Chang
...
<abstract> Structural matching and discovery in
documents such as SGML and HTML is important
for data warehousing, version management, ...
<section>
<title> Introduction <\title>
...
<paragr> Advances in hardware and software have
stimulated corresponding advances in document
presentation, in the management and retrieval of
...
<paragr> Likewise, the ability to manage ...
...
<\article>
```

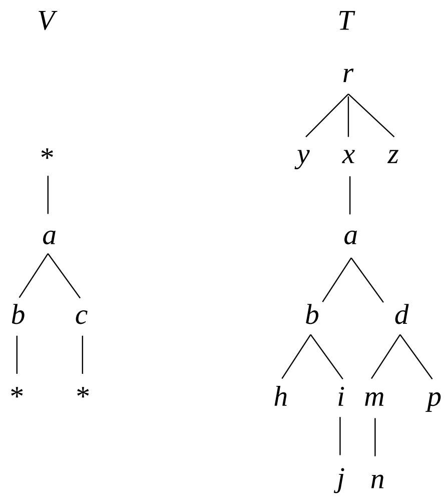
## Comparing Two Documents using Tree Matching




Comparing doc1 and doc2 shown in the previous pages. The figure shows a mapping from doc1 to doc2. The nodes in the rectangular box in doc1 are to be deleted. The distance between doc1 and doc2 would be 2 (representing the cost of deleting the two nodes in the rectangular box). The underlying algorithm is used to support the **symdifference** and **unidifference** operations in TREEDIFF.



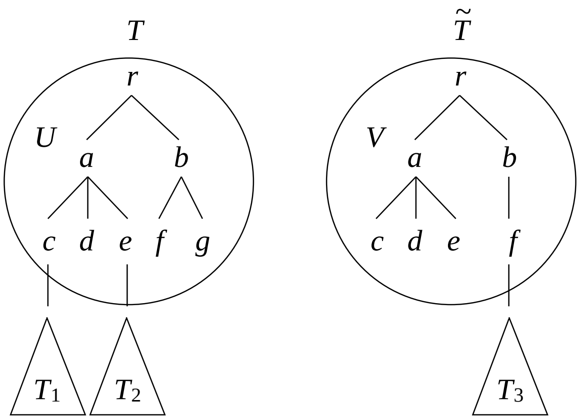
## VLDC Tree Matching




Matching a VLDC pattern $V$ and a tree $T$. The root $*$ in $V$ would be matched with nodes $r, x$ in $T$, and the two leaves $*$ in $V$ would be matched with nodes $i, j$ and $m, n$ in $T$, respectively. Nodes $y, z, h, p$ in $T$ would be cut. The distance of $V$ and $T$ would be 1 (representing the cost of changing $c$ in $V$ to $d$ in $T$). The underlying algorithm is used to support the **substructure comparison** and **superstructure comparison** operations in TREEDIFF.



## Finding Common Portion in Two Trees




To find the common portion of two trees $T$ and $\tilde{T}$, we want to maximize $|U| + |V|$ subject to the constraint $dist(U, V) \leq d$ where $d$ is a user-provided distance value. $T_1$, $T_2$ and $T_3$ are removed without incurring any cost. The underlying algorithm is used to support the **union** and **consensus** operations in TREEDIFF.



## References


- D. Shasha and K. Zhang. Fast algorithms for the unit cost editing distance between trees. *Journal of Algorithms*, 11(4):581–621, 1990.

- J. T. L. Wang, K. Zhang, K. Jeong, and D. Shasha. A system for approximate tree matching. *IEEE Transactions on Knowledge and Data Engineering*, 6(4):559–571, 1994.

- J. T. L. Wang, K. Zhang, and D. Shasha. Pattern matching and pattern discovery in scientific, program, and document databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, page 487, San Jose, California, 1995.

- K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

- K. Zhang, D. Shasha, and J. T. L. Wang. Approximate tree matching in the presence of variable length don't cares. *Journal of Algorithms*, 16(1):33–66, 1994.