Structural Matching and Discovery in Document Databases*

Jason Tsong-Li Wang
Computer and Information Science
New Jersey Institute of Technology
jason@village.njit.edu

Dennis Shasha
Courant Institute
New York University
shasha@cs.nyu.edu

George J. S. Chang
Computer and Information Science
New Jersey Institute of Technology
gchang@homer.njit.edu

Liam Relihan
Piercom Ltd., Inter. Business Center
National Tech. Park, Limerick, Ireland
liam.relihan@ul.ie

Kaizhong Zhang
Computer Science Department
Univ. of Western Ontario, Canada
kzhang@csd.uwo.ca

Girish Patel
Computer and Information Science
New Jersey Institute of Technology
girish@homer.njit.edu

1 Background

Structural matching and discovery in documents such as SGML and HTML is important for data warehousing [6], version management [7, 11], hypertext authoring, digital libraries [4] and Internet databases. As an example, a user of the World Wide Web may be interested in knowing changes in an HTML document [2, 5, 10]. Such changes can be detected by comparing the old and new version of the document (referred to as structural matching of documents). As another example, in hypertext authoring, a user may wish to find the common portions in the history list of a document or in a database of documents (referred to as structural discovery of documents). In SIGMOD 95 demo sessions, we exhibited a software package, called *TreeDiff* [13], for comparing two latex documents and showing their differences. Given two documents, the tool represents the documents as ordered labeled trees and finds an optimal sequence of edit operations to transform one document (tree) to the other. An edit operation could be an insert, delete, or change of a node in the trees. The tool is so named because documents are represented and compared using approximate tree matching techniques [9, 12, 14].

2 System Architecture and Operators

Here we present an extension of *TreeDiff* for querying, comparing and discovering structured documents. Our new system is equipped with a graphical interface and a powerful query language that allows the user to compare documents and to discover the (approximately) common portions of documents. The front end of the system is composed of a query processor and an SGML parser originally developed by James Clark. Given a document type definition (DTD), the SGML parser checks if an input document conforms to the DTD. If the syntax is correct, the parser translates the document to an ordered labeled tree. The query processor checks the syntax of a given query. When the query involves the comparison or structural discovery of documents, the query processor invokes the back end of the system, which performs approximate tree matching and discovery. Figure 1 illustrates the system architecture.

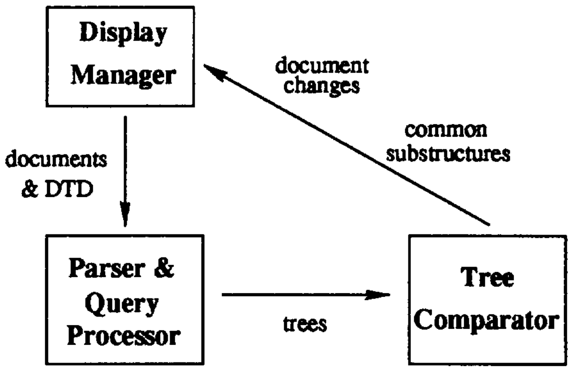


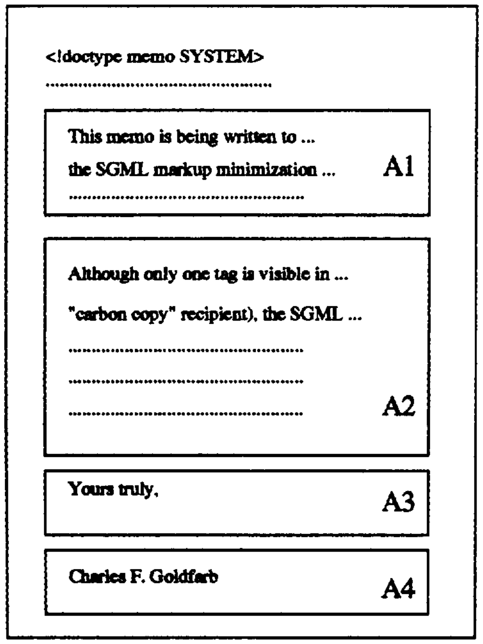
Fig. 1. System architecture.

The system provides several useful operators, which can be used alone or included in a query, for comparing two documents A and B:

- 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).
- difference(A, B) – to find the structural elements in A not in B and the structural elements changed from A to B.
- intersection(A, B) – to find the structural elements in common between A and B (cf. Figures 2 – 4).
- union(A, B) – to find the structural elements in common, 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.
- mergable(O, A, B) – to determine whether A and B can be merged where A, B are the documents obtained by modifying document O; this function is true if B is obtained by modifying different portions of O than A.

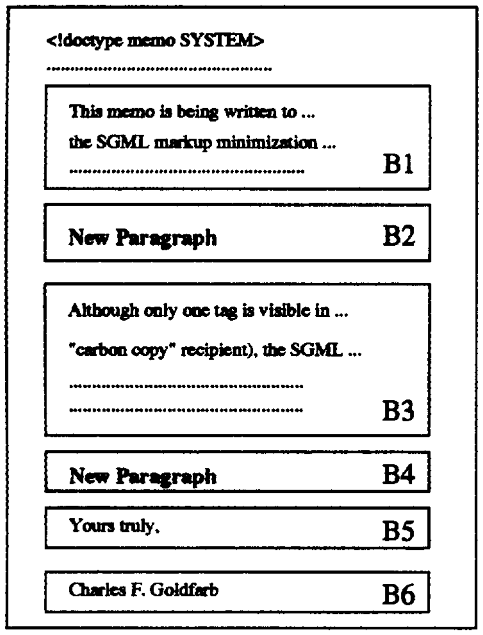
*Work partially supported by NSF grants IRI-9224601, IRI-9224602, IRI-9531548, IRI-9531554, and by the Natural Sciences and Engineering Research Council of Canada under Grant No. OGP0046373.

- merge(O, A, B) – to perform the merge of A and B; whenever some portion has changes from both A and B, then A's changes will occur, but not B's.



<!doctype memo SYSTEM>
This memo is being written to ...
the SGML markup minimization ...
A1
Although only one tag is visible in ...
"carbon copy" recipient), the SGML ...
A2
Yours truly,
A3
Charles F. Goldfarb
A4

Fig. 2. An SGML memo document A.



<!doctype memo SYSTEM>
This memo is being written to ...
the SGML markup minimization ...
B1
New Paragraph
B2
Although only one tag is visible in ...
"carbon copy" recipient), the SGML ...
B3
New Paragraph
B4
Yours truly,
B5
Charles F. Goldfarb
B6

Fig. 3. An SGML memo document B.

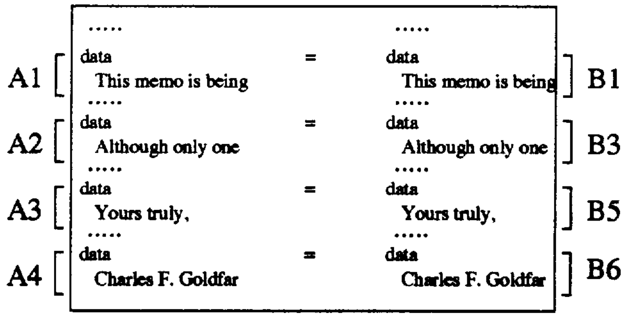


Fig. 4. The result of intersection(A, B). For each document element (e.g. a paragraph), only the first 18 characters are displayed.

3 Related Work

Whereas much database research has been conducted to manage structured documents such as SGML and HTML (e.g., [1, 3, 8]), very few systems have been built to support their comparison or the discovery of patterns in their structures. As far as we know, only the *LaDiff* program developed in [2] supports structural matching. Both our system and *LaDiff* represent hierarchically structured documents as trees and identify changes in them. In contrast to other systems dealing with flat information, the two systems are concerned with not only the "nodes" in the data, but also their relationships. For example, if a node (and its children) is moved from one location to another, both systems represent it as a "move" operation in the data. The two systems differ, however, in some respects. First, they use different notions of editing. We use a generalization of edit distance as used in the Unix utility *diff*. *LaDiff* approximates this edit distance in order to run faster. Second, we include both querying and structural pattern discovery.

4 The Demonstration

In the demonstration of SIGMOD 97, we will show new queries for performing:

- document matching: aligning two documents and detecting changes or differences of the documents;
- document discovery: aligning two documents and identifying the largest (approximately) common portions of the documents;
- multiple documents discovery: finding the (approximately) common portions of a database of documents;
- similarity search: finding the document in a database that is most similar to a given document;
- substructure search: finding the document in a database that contains a given document;
- superstructure search: finding the document in a database that is the same as a substructure of a given document.

The output of a query can be displayed alone or combined with that of other utilities such as *diff*. For example, Figure 5 illustrates document matching, showing the result of symdifference(C, D) of two memos C and D. "=" indicates that the two document elements (e.g. two section titles or two paragraphs) are the same, while "|" indicates that the document elements are changed, ">" indicates that the corresponding document element is inserted, and "<" indicates the document element is deleted. The detailed comparisons of the changed paragraphs using *diff* are displayed at the end.

```
MEMO                              |   MEMO
   <MEMO FINAL SEC="                     <MEMO FINAL SEC="
   ...                                   ...
data                              <
   Here is more infor
   ...                                   ...
                                  >   data
                                         This paragraph has
   ...                                   ...
data                              |   data
   Although only one                     Although only one
   ...                                   ...
data                              |   data
   recipient), the SG                    recipient
   ...                                   ...
   ...                                   ...
   ...                                   ...
   ...                                   ...

 ***** EDITED TEXT *****

 ::::: memo_C.sgm line 13-20         memo_D.sgm: line 14-21 :::::
 3,5c3,5
 < carbon copy recipient), the SGML parser recognizes 15 ......
 < element types. This allows each element to be formatted .....
 < for example, the start and end of a quotation can have ..........
 ---
 > carbon copy recipient), the SGML parser recognizes 14 ......
 > different element types. This allows each element to be .......
 > differently; for example, the start and end of a quotation ......
```

Fig. 5. The result of symdifference(C, D).

Document changes or common portions are highlighted in a colorful fashion through the graphical interface of our system. The system is implemented using C, Perl and Tcl-Tk on SUN SPARC workstations. Figure 6 shows a screen shot of the system. We have prepared three versions: one for SunOS, one for Solaris and one for other operating systems. The software is available for research purposes and can be obtained from the authors (please visit the Web site http://www.cis.njit.edu/~jason/demo.html for details).

References

[1] K. Bohm and K. Aberer. HyperStorM - administering structured documents using object-oriented database technology. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, page 547, Montreal, Quebec, Canada, June 1996.

[2] S. S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change detection in hierarchically structured information. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 493–504, Montreal, Quebec, Canada, June 1996.

[3] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 313–324, Minneapolis, Minnesota, May 1994.

[4] F. Douglis and T. Ball. Tracking and viewing changes on the Web. In *Proceedings of 1996 USENIX Technical Conference*, Jan. 1996.

[5] F. Douglis, T. Ball, Y.-F. Chen, and E. Koutsofios. WebGUIDE: Querying and navigating changes in Web repositories. http://www.ics.forth.gr/~telemed/www5/www181/overview.htm.

[6] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The Stanford Data Warehousing Project. *IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing*, 18(2):41–48, June 1995.

[7] H. C. Howard, A. M. Keller, A. Gupta, K. Krishnamurthy, K. H. Law, P. M. Teicholz, S. Tiwari, and J. Ullman. Versions, configurations, and constraints in CEDB. CIFE Working Paper 31, Center for Integrated Facilities Engineering, Stanford University, April 1994.

[8] T. Nguyen and V. Srinivasan. Accessing relational databases from the World Wide Web. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 529–540, Montreal, Quebec, Canada, June 1996.

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

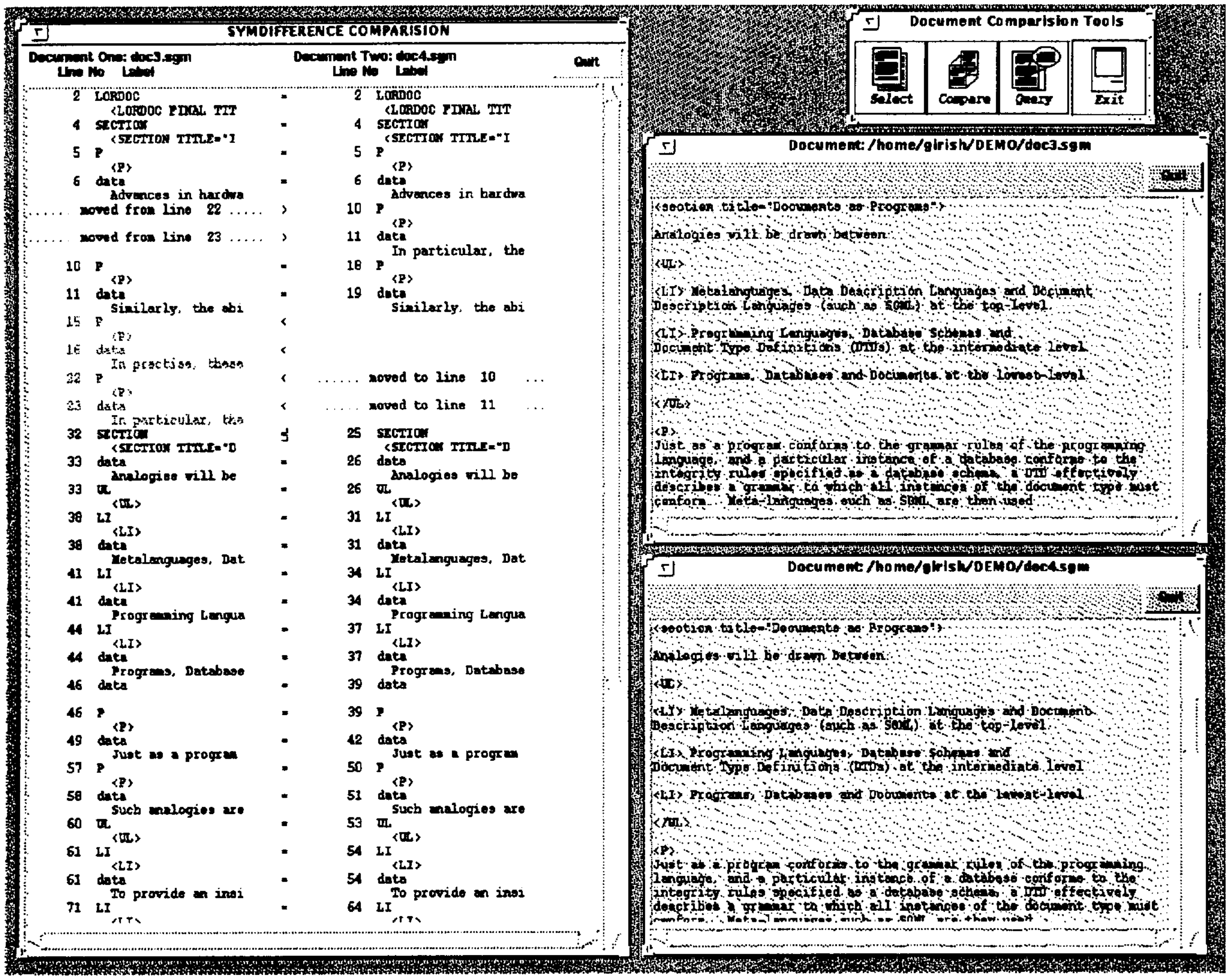
[10] The C3 Project at Stanford. Changes, consistency, and configurations in heterogeneous distributed information systems. http://www-db.stanford.edu/c3/c3.html#999.

[11] W. Tichy. RCS: A system for version control. *Software - Practice and Experience*, 15(7):637–654, July 1985.

[12] 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, August 1994.

[13] 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, May 1995.

[14] 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, Dec. 1989.


SYMDIFFERENCE COMPARISION
Quit
Document Comparision Tools
Select
Compare
Query
Exit
Document: /home/girish/DEMO/doc3.sgm
Quit
Document: /home/girish/DEMO/doc4.sgm
Quit

```
Document One: doc3.sgm                 Document Two: doc4.sgm
     Line No   Label                        Line No   Label
     2  LORDOC                      =       2  LORDOC
          <LORDOC FINAL TIT                   <LORDOC FINAL TIT
     4  SECTION                     =       4  SECTION
          <SECTION TITLE="I                   <SECTION TITLE="I
     5  P                           =       5  P
          <P>                                 <P>
     6  data                        =       6  data
          Advances in hardwa                  Advances in hardwa
...... moved from line  22 .....    >      10  P
                                              <P>
...... moved from line  23 .....    >      11  data
                                              In particular, the
    10  P                           =      18  P
          <P>                                 <P>
    11  data                        =      19  data
          Similarly, the abi                  Similarly, the abi
    15  P                           <
          <P>
    16  data                        <
          In practise, these
    22  P                           <   ...... moved to line  10   ...
          <P>
    23  data                        <   ...... moved to line  11   ...
          In particular, the
    32  SECTION                     ≤      25  SECTION
          <SECTION TITLE="D                   <SECTION TITLE="D
    33  data                        =      26  data
          Analogise will be                   Analogies will be
    33  UL                          =      26  UL
          <UL>                                <UL>
    38  LI                          =      31  LI
          <LI>                                <LI>
    38  data                        =      31  data
          Metalanguages, Dat                  Metalanguages, Dat
    41  LI                          =      34  LI
          <LI>                                <LI>
    41  data                        =      34  data
          Programming Langua                  Programming Langua
    44  LI                          =      37  LI
          <LI>                                <LI>
    44  data                        =      37  data
          Programs, Database                  Programs, Database
    46  data                        =      39  data

    46  P                           =      39  P
          <P>                                 <P>
    49  data                        =      42  data
          Just as a program                   Just as a program
    57  P                           =      50  P
          <P>                                 <P>
    58  data                        =      51  data
          Such analogies are                  Such analogies are
    60  UL                          =      53  UL
          <UL>                                <UL>
    61  LI                          =      54  LI
          <LI>                                <LI>
    61  data                        =      54  data
          To provide an insi                  To provide an insi
    71  LI                          =      64  LI
```

```
<section title="Documents as Programs">

Analogies will be drawn between

<UL>

<LI> Metalanguages, Data Description Languages and Document
Description Languages (such as SGML) at the top-level.

<LI> Programming Languages, Database Schemas and
Document Type Definitions (DTDs) at the intermediate level.

<LI> Programs, Databases and Documents at the lowest-level

</UL>

<P>
Just as a program conforms to the grammar rules of the programming
language, and a particular instance of a database conforms to the
integrity rules specified as a database schema, a DTD effectively
describes a grammar to which all instances of the document type must
conform.  Meta-languages such as SGML are then used
```

```
<section title="Documents as Programs">

Analogies will be drawn between

<UL>

<LI> Metalanguages, Data Description Languages and Document
Description Languages (such as SGML) at the top-level

<LI> Programming Languages, Database Schemas and
Document Type Definitions (DTDs) at the intermediate level

<LI> Programs, Databases and Documents at the lowest-level

</UL>

<P>
Just as a program conforms to the grammar rules of the programming
language, and a particular instance of a database conforms to the
integrity rules specified as a database schema, a DTD effectively
describes a grammar to which all instances of the document type must
```

Fig. 6. The result of comparing two SGML articles. The two windows on the right show the articles. The left window displays the output of the comparison. In addition to showing the inserts, deletes and changes of paragraphs, the system can also show their movement (lines 10 and 22). When the user clicks on the particular document element of interest, that portion of the document is scrolled to the top in the right windows. If the user is interested in section changes only, rather than paragraph changes, the system displays a size parameter for each section, where "size = n" means that the total number of inserts, deletes and changes of paragraphs in that section is n. ,Moreover, the section with a larger change size is displayed in a deeper color.