

Complementary Approaches to Representing Differences Between Structured Documents

David T. Barnard and George M. Logan

Computing and Information Science, Queen's University, Kingston, Ontario K7L 3N6 Canada and

English Language and Literature, Queen's University, Kingston, Ontario K7L 3N6 Canada

KEYWORDS: structured documents, SGML

AFFILIATION: Queen's University, Canada

E-MAIL: David.Barnard@queensu.ca
logang@qucdn.bitnet

FAX NUMBER: 1-613-545-6513

PHONE NUMBER: 1-613-384-1537

Structured documents

Documents can be represented as structures with a hierarchical arrangement of text and non-text nodes, where nodes are labelled by category names such as "paragraph" and "section". Representing documents this way is a natural consequence of using the Standard Generalized Markup Language (SGML) to encode the content and form of documents [10, 11, 7]. SGML is widely used. HTML, the encoding used for World Wide Web documents, is an application of SGML [6]; although HTML is used to build hypertext networks of documents rather than hierarchies, each document is itself a hierarchy with explicitly coded links to build the network. The Text Encoding Initiative uses SGML to encode complex texts [21, 4, 2]. Even documents that are not simple hierarchies can be represented using SGML [3].

Formally, documents represented in SGML or HTML are trees with labelled nodes where the left to right ordering of the offspring of a node is significant. Any piece of text in the document can be treated as a single labelled node, all leaf nodes are text nodes (or empty) and any node with children is a structural or non-text node.

Families of structured documents:

Complementary views

There are many circumstances in which one structured document is a variant of another. A situation in which a scholar deals with variants of a document can come about in two complementary ways. In the first case, two or more existing documents are known to be closely related. For example, there may be several manuscript or printed versions of an existing text; the details of the relationships

among the versions may be known or may have to be inferred, but the variations among them are small compared to the amount of text in common among them. Or a text may exist in more than one language with the original and the translations being structurally identical. In this view of variant texts, then, the documents are given, and the relationships among them are derived.

In the second case, a new document is derived from an existing document by a known sequence of transformations. If, for example, a text is being created in a machine-readable form there may be several versions produced for different audiences (e.g., a text with minimal apparatus for students and a more extensive apparatus for researchers), or there may be interest in maintaining several versions of a document that is being produced in a cooperative manner by several authors [9]. In this setting, the changes made to the document can be stored to produce a record of its evolution, and to enable regeneration of any of its (intermediate) forms.

In this second view of variant documents, where the relationships (editing changes) are explicitly given and the documents are derived, it is natural to use hypertext structures [17, 23].

Applications

Software to handle variant texts in each of these two complementary views can be useful to researchers.

Editing or Co-authoring: When an author presents a modified version of a document to an editor or co-author, the two versions could be compared to isolate only the changed components. A sophisticated display mechanism could highlight the differences. This would be a powerful editing tool in either of the settings discussed above. If the documents are given, the differences to be displayed must first be computed. If a sequence of transformations is given, in general some sequence of changes must be applied to determine their net effect.

Storing: Documents are often published in several versions. A "document control system" might be modelled on a source code control system to provide help in managing versions. One of the characteristics of such a system should be that it minimizes storage requirements by retaining only the computed differences between versions of a document without having to store more than one complete document. This is a natural application of the second view of document families.

Querying: If documents are stored in an archive using a structured representation, a query against the archive could also have a tree structure. The document (or document fragments) that satisfy the query would be those that most closely match the query tree, given an appropriate metric for dis-

ce. This is a natural application of the first view of document families.

Computational requirements

We have studied the problem of managing document families using the first approach, where documents are given and relationships among them are to be derived (or computed) [5].

The general approach to “edit distance” problems (for strings or trees) in the literature has been to define a sequence of primitive operations that can be applied to one object to produce another, and to define the distance between two objects as a function computed on a sequence of such operations [19, 20, 22, 24, 25, 26]. In simple cases it can be sufficient to determine the length of the sequence. More realistically and more generally, each operation is assigned a “cost” that represents the difficulty of making the indicated change to the object. The cost could be thought of as the perceived unlikelihood of the change having arisen at random in whatever process produced the changed object. The problem is attacked by deriving an algorithm to search for a sequence of operations with minimal cost. (In effect, a linear programming algorithm is used to search for the desired sequence.)

For string-to-string editing, the operations most frequently considered are: insert a character, delete a character, and change one character into another. With trees, other operations are useful, including:

- insertTree: Add an entire subtree (possibly one leaf node) to the target tree.
- deleteTree: Remove an entire subtree (possibly one leaf node) from the original tree.
- insert: Add an internal node to the target tree. delete: Remove an internal node from the original tree.
- change: Relabel a node in the original tree with the label of a node in the target tree.
- swap: Swap subtrees rooted at adjacent siblings.
- swap with editing: Swap subtrees rooted at adjacent siblings and edit the subtrees in the original tree to the corresponding subtrees in the target tree.

Determining the set of primitive operations that can represent a change between versions is an important aspect of this approach. In addition to the kinds of changes just enumerated, machine-readable documents can have global changes applied (for example, change all occurrences of “co-

lor” to “color”) in addition to those already described. Further work is required to incorporate such operations if the set of transformations is intended accurately to represent what might actually occur as documents evolve, rather than simply to be a mathematical model of distance.

Algorithms for dealing with tree-to-tree correction can be extended to structured documents. If the documents are encoded in SGML, the tag name and the attributes are taken together as the label on the node.

In the complementary view, the computational requirements appear more simple and straightforward: the changes encoded in the editing transactions must be applied to the original document to form the derived document. Other more complex computations are also useful, though, such as finding a minimal representation of a sequence of changes. This requires the definition of a calculus of basic editing operations, and an algorithm to implement that calculus. For example, if as a document is being edited a word is inserted into a text and later deleted, the minimal representation of the changes should not include either the insertion or the deletion.

Conclusion

A powerful electronic document processing system for research purposes should provide tools to maintain document versions given an explicit set of relationships among them, but should also include an algorithm for deriving a set of relationships from explicitly presented versions. For such an algorithm to be useful, it must incorporate a model of a rich set of document-editing primitives.

References

- [1] Lloyd Allison and Trevor I. Dix; A Bit-String Longest-Common-Subsequence Algorithm; *Information Processing Letters* 23 (1986) 305–310.
- [2] David T. Barnard, Lou Burnard, Steven J. DeRose, David G. Durand and C.M. Sperberg-McQueen; Lessons for the World Wide Web from the Text Encoding Initiative; *Proceedings 4th International Conference on the World Wide Web*, Boston (1995).
- [3] David T. Barnard, Lou Burnard, Jean-Pierre Gaspart, Lynne Price and C.M. Sperberg-McQueen; Hierarchical Encoding of Text: Technical Problems and SGML Solutions; *Computers and the Humanities* 29,3 (1995) 211–231.
- [4] David T. Barnard, Lou Burnard and C.M. Sperberg-McQueen; Lessons Learned From Using SGML in the Text Encoding Initiative; *Computer Standards and Interfaces* (accepted 1995).
- [5] David T. Barnard, Nicholas Duncan and Gwen

- Clarke; Tree-to-tree Correction for Document Trees; Technical Report 95-372, Department of Computing and Information Science, Queen's University, Kingston, 1995.
- [6] T. Berners-Lee and D. Connolly; Hypertext Markup Language - 2.0, `aft-ietf-html-spec-06.txt`; HTML Working Group, Boston (1995).
- [7] Robin C. Cover; SGML Web Page; <http://www.sil.org/sgml/sgml.html> (1994).
- [8] Karel Culik II and Derick Wood; A Note on Some Tree Similarity Measures; *Information Processing Letters* 15,1 (1982) 39-42.
- [9] David G. Durand; Palimpsest: A Data Model for Version Control; Proceedings of the CSCW 94 Workshop on Collaborative Hypermedia Systems, Chapel Hill, North Carolina (1994).
- [10] International Organization for Standardization; Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML); ISO 8879 (1986).
- [11] Charles Goldfarb; *The SGML Handbook*; Oxford University Press (1990).
- [12] R. Lowrance and R.A. Wagner; The Extended String-to-String Correction Problem; *Journal of the ACM* 22,2 (1975) 177-183.
- [13] Jean Pallo; Enumerating, Ranking and Unranking Binary Trees; *The Computer Journal* 29,2 (1986) 171-175.
- [14] Jean Pallo; On the Rotational Distance in the Lattice of Binary Trees; *Information Processing Letters* 25 (1987) 369-373.
- [15] D.F. Robinson; Comparison of Labelled Trees with Valency Three; *Journal of Combinatorial Theory* 11 (1971) 105-119.
- [16] D.F. Robinson and L.R. Foulds; Comparison of Phylogenetic Trees; *Mathematical Biosciences* 53 (1981) 131-147.
- [17] Andrew Ross; A Data Model for Versions; M.Sc. Thesis, Department of Computing and Information Science, Queen's University (1995).
- [18] David Sankoff and Joseph B. Kruskal; *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*; Addison-Wesley (1983).
- [19] Stanley M. Selkow; The Tree-to-Tree Editing Problem; *Information Processing Letters* 6,6 (1977) 184-186.
- [20] Dennis Shasha and Kaizhong Zhang; Fast Parallel Algorithms for the Unit Cost Editing Distance Between Trees; Technical Report, Courant Institute of Mathematical Sciences, NYU (1989) Draft Version.
- [21] C.M. Sperberg-McQueen and Lou Burnard, eds.; *Guidelines For Electronic Text Encoding and Interchange (TEI P3)*; ACH-ACL-ALLC Text Encoding Initiative (1994).
- [22] Kuo-Chung Tai; The Tree-to-Tree Correction Problem; *Journal of the ACM* 26,3 (1979) 422-423.
- [23] Fabio Vitali and David G. Durand; Using Versioning to support collaboration on the WWW; Proceedings 4th International Conference on the World Wide Web, Boston (1995).
- [24] R.A. Wagner; On the Complexity of the Extended String-to-String Correction Problem; Proceedings of the Seventh Annual ACM Symposium on Theory of Computing (1975) 218-223.
- [25] R.A. Wagner and M.J. Fischer; The String-to-String Correction Problem; *Journal of the ACM* 21,1 (1974) 168-173.
- [26] Kaizhong Zhang and Dennis Shasha; Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems; *SIAM Journal of Computing* 18,6 (1989) 1245-1262.