BIDEL: An XML-based System for Effective Fast Change Detection of Genomic and Proteomic Data

Song Yang[2] Sourav S. Bhowmick[1,2]

Singapore-MIT Alliance, Nanyang Technological University, Singapore[1]
School of Computer Engineering, Nanyang Technological University, Singapore[2]
assourav@ntu.edu.sg

Abstract. A key issue to address in biological data integration is how to detect changes to the underlying biological data sources. In this demonstration, we present a novel system called BIDEL for detecting changes to genomic and proteomic data (sequences and annotations). We transform heterogeneous biological data to XML format (if necessary) and then detect changes between two versions of unordered XML representation of biological data. This demonstration will showcase the functionality of our system and the effectiveness of change detection in life sciences environment.

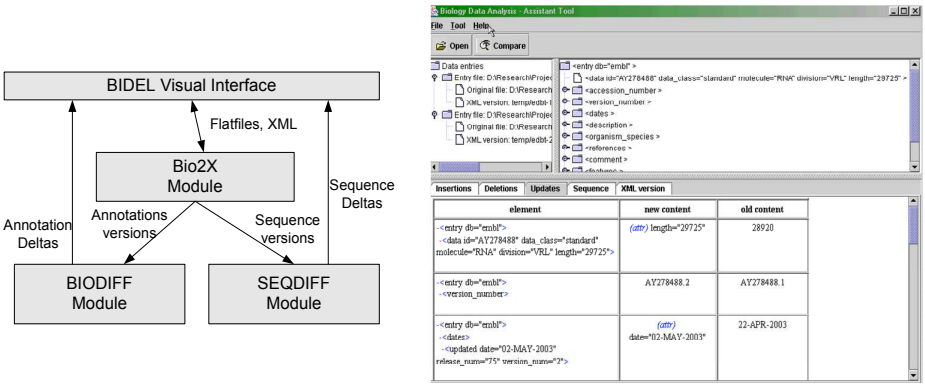
1 Introduction

Detecting changes to the underlying biological data sources is a key challenge in biological data integration. In this demonstration, we present a system called BIDEL[1] (***Bi****ological* ***Del****ta Detector*) for detecting changes to old and new versions of genomic and proteomic data. In our system, we first transform heterogeneous genomic and proteomic data to XML format [4] (if necessary) and then detect changes between two versions of unordered XML representation of biological data [2, 3]. Specifically, the BIODIFF [2] component of BIDEL detects *exact* changes to the *annotation* (non-sequence) data associated with gene or protein sequences. It *extends* X-Diff [6], a published unordered XML change detection algorithm, by addressing its limitations to exploit structural characteristics of underlying data (discussed in Section 3 and [2]). On the other hand, the SEQDIFF module detects changes to *sequence* data. Note that existing XML change detection techniques [6] are not designed to compute changes to sequences. To the best of our knowledge, *this is the first system to detect changes to both annotation and sequence data associated with biological entities.*

2 System Overview

Figure 1(a) shows the architecture of BIDEL and consists of the following modules.
The Visual Interface Module: Figure 1(b) depicts the screen dump of the visual interface of BIDEL. It consists of three panels. The top-left panel displays a list of versions of biological data (genomic and proteomic data in XML as well as flat file format) that we wish to compare in BIDEL. A user can view the details of a document in the top right panel by clicking on the correspond item in the list. Note that the transformation of a flat file to XML format is achieved by clicking on the `Open` icon in the menu. It

[1] In Maltese, bidel means "to change".



(a) BIDEL Architecture **(b) GUI of BIDEL**

Fig. 1. Architecture and visual interface of BIDEL.

invokes the Bio2X [4] algorithm for the transformation (discussed below). Given an old and new versions of XML representation of biological data, the changes to the data are computed by clicking on the `Compare` icon in the menu. It invokes the BIODIFF [2] and SEQDIFF [3] algorithms to detect the changes to annotation (non-sequence) and sequence data, respectively. The bottom panel displays various types of changes that are computed by these two algorithms. A user can click on one of the four tabs (`Insertions`, `Deletions`, `Updates`, and `Sequence`) to view a specific type of changes. For instance, in Figure 1(b) clicking on the `Updates` tab results in the display of a list of updates detected by BIODIFF. Similarly, Figure 2 depicts the changes to sequence data when the `Sequence` tab is clicked. A user can also view the complete set of changes in XML format by clicking on the `XML version` tab.

The Bio2X Module: This module currently converts flat file data from GenBank, EMBL, Swiss-Prot, and PDB into XML format. The rule bases are designed in a consistent manner so that a single transformer is sufficient to parse any data file from any database. The transformer chooses a suitable rule base for parsing the input flat file based on its origin database and generates the XML data file. The rule base exploits the *hierarchical* structure of the source to constrain the data extraction problem. It allows for extraction of target patterns based on surrounding landmarks, *line types* and other lexical patterns in the flat files. It also allows for more advanced features such as disjunctive pattern definitions. Finally, it involves machine-learning techniques to refine the rules in order to improve the accuracy of the transformation. The reader may refer to [4] for details related to the transformer.

The BIODIFF Module: This module implements the algorithm BIODIFF [2] that identifies *exact* changes to the *annotations* associated with primary biological objects (gene and protein sequences). It takes as input two versions of unordered XML representation of annotations of a gene or a protein (the sequence data is excluded) from the *Bio2X* module, denoted by D_1 and D_2, and detect changes between them. The algorithm extends X-Diff [6] by addressing some of its limitation and consists of four phases, namely the *identifier checking* phase, the *parsing and hashing* phase, the *matching* phase, and the *edit script generation* phase. The *identifier checking phase* determines whether the two versions are identical by comparing the *version identifiers* of the biological data

records. If the two entries are not identical, then BIODIFF parses D_1 and D_2 into DOM trees $tree1$ and $tree2$ in the *parsing and hashing phase*. This step is similar to the one in X-Diff [6]. The goal of the *matching phase* is to compute the minimum cost matching between $tree1$ and $tree2$. Each XML tree is divided into a set of smaller subtrees rooted at distinct first-level nodes. Note that each first-level element nodes resulted from *Bio2X* has a unique name and hierarchy. Each smaller tree is compared with another smaller tree from the second XML tree having the node with same name. This step makes it possible to use *different* methods of matching for subtrees having *different* characteristics (e.g, elements containing distinct or identical subelements). Note that these characteristics of the subtrees are extracted by the *Bio2X* module during XML transformation. BIODIFF employs four types of matching techniques for different subtree characteristics, namely *one-to-one* comparison, *identical subelement* comparison, *extended signature* comparison, and *bipartite matching*. Lastly, similar to X-Diff, the *edit script generation phase* generates a minimum-cost edit script for changes to annotation data based on the minimum cost matching found in the matching phase.

The SEQDIFF Module: This module implements a heuristic non-optimal algorithm called SEQDIFF [3] to detect changes between two versions of biological sequences of the same biological entity extracted by the *Bio2X* module. Specifically, it detects insert, delete, and update of a nucleotide or protein sequence at a specific position. The algorithm consists of two phases, namely the *sequence comparison* phase and the *edit script generation* phase. The first phase is based on the local alignment concept used in BLAST. That is, genes adjacent in one sequence should also remain near to each other in the new sequence. Based on this heuristic, the sequence is divided into segments (that act like sliding windows) and an optimal alignment is performed within each segment without any consideration for other choices from the other segment. Note that the algorithm computes the alignment twice by first matching the first sequence onto second one; and then matching the second one onto the first one. This is because it is not known *apriori* in which direction of match generates the higher score. In the second phase, the edit script is generated based on the alignment with a higher score. The alignment result is traversed and the aligned segments with same values are *matched*. The aligned segments with different values are *updated*. The unaligned segments in the new version are *inserted*, while the ones in the old version are *deleted*. Figure 2 shows a screenshot of the edit script generated by the SEQDIFF module.

3 Related Systems and Novelty

A number of techniques for detecting changes to ordered and unordered XML data has been proposed (e.g., [6]). BIDEL differs from these approaches in the following ways. Firstly, since the min-cost max-flow algorithm for computing the bipartite mapping between two XML trees is the most time consuming part, it is desirable to reduce the size of data set during mapping. Existing XML change detection techniques fail to do so for biological data as it ignores the structural semantics of the underlying data. In contrast, BIDEL reduces the data size for bipartite mapping by exploiting the structural characteristics of XML representation of biological data. Consequently, BIODIFF shows better performance than X-Diff (up to 6 times faster) [2]. Secondly, none of these techniques are designed to detect changes to sequence data. The SEQDIFF component of BIDEL

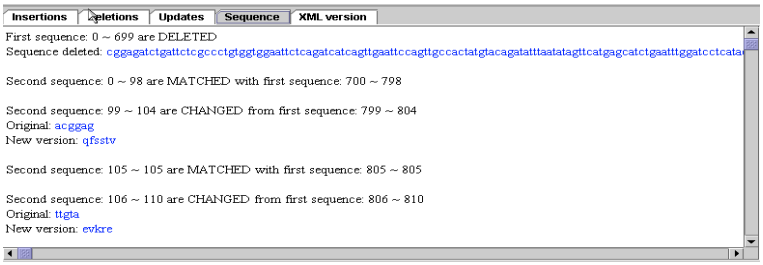


Fig. 2. Output of SEQDIFF module.

implements a heuristic strategy to detect different types of changes to old and new versions of sequence data. Lastly, to the best of our knowledge, none of the existing XML change detection system has been demonstrated in a major database conference.

Pairwise sequence alignment techniques [1, 5] can be considered as the closest to the SEQDIFF module. Our change detection tool differs from these techniques in the following ways. First, sequence alignment techniques focus on finding *similarities* between the sequences whereas our technique focus on finding *differences* between a pair of sequences. Second, these approaches are designed to compare sequences among *different* biological entities. However, in change detection problem we are interested in detecting changes to two versions of a sequence of the *same* entry in terms of insertion, deletion, and update. Third, is the issue of performance. SEQDIFF trades off optimality for better performance. Specifically, SEQDIFF is significantly faster than DCLBDA [1], an optimal sequence alignment algorithm (highest observed factor being 350 times [3]).

4 Demonstration

Our demonstration aims to showcase the functionality and effectiveness of the BIDEL system in detecting changes to genomic and proteomic data. We will showcase the followings. (a) Demonstrate detection of different types of changes to annotation and sequence data using real-world datasets (EMBL, PDB, and Genebank). We will show how this process is simplified by the BIDEL visual interface. (b) Demonstrate the cases when the result quality of BIDEL is comparable to X-Diff as far as detection of changes to annotation data is concerned. (c) Demonstrate better efficiency and scalability of BIODIFF module compared to general unordered XML change detection algorithms (say X-Diff). We will also show cases where X-Diff fails to detect changes to annotation data due to lack of memory but BIODIFF is able to detect these changes.

References

1. Davidson A., A Fast Pruning Algorithm for Optimal Sequence Alignment. *In IEEE BIBE*, 2001.
2. Song,Y., Bhowmick,S.,S., BioDiff: An Effective Fast Change Detection Algorithm for Genomic and Proteomic Data. *In DASFAA* , 2007.
3. Song,Y., Bhowmick,S.,S., SeqDiff: An Effective Fast Change Detection Algorithm for Biological Sequences. *Technical Report* , Available at `www.cais.ntu.edu.sg/~assourav/TechReports/SeqDiff-TR.pdf`, 2008.
4. Song,Y., Bhowmick,S.,S., Bio2X: A Rule-based Approach for Semi-automatic Transformation of Semistructured Biological Data to XML, *Data and Knowledge Engineering Journal*, **52(2)**, 249-271, 2003.
5. Tatusova,T.A., Madden, T.L., BLAST 2 Sequence, a new tool for comparing protein and nucleotide sequences. *FEMS Microbiology Letters* , 174, pp. 247–250, 1999.
6. Wang,Y., DeWitt,D., Cai,J-Y, X-Diff: A Fast Change Detection Algorithm for XML Documents, *In ICDE* , 519-530, 2003.