A Matching Algorithm for Electronic Data Interchange

Rami Rifaieh\textsuperscript{1}, Uddam Chukmol\textsuperscript{2}, and Nabila Benharkat\textsuperscript{3}

\textsuperscript{1} San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Dr. La Jolla, CA 92093-0505, USA
rrifaieh@sdsc.edu

\textsuperscript{2} Computer Science Department, Cambodia Technological Institute, P.O. Box 86, Bld. Pochentong, Phnom Penh, Cambodia
uddam.chukmol@itc.edu.kh

\textsuperscript{3} LIRIS – National Institute of Applied Science of Lyon, 7 Avenue J.Capelle, 69621 Villeurbanne, France
nabila.benharkat@insa-lyon.fr

Abstract. One of the problems in the actual electronic commerce is laid on the data heterogeneity (i.e. format and vocabulary). This representation incompatibility, particularly in the EDI (Electronic Data Interchange), is managed manually with help from a human expert consulting the usage guideline of each message to translate. This manual work is tedious, error-prone and expensive. The goal of this work is to partially automate the semantic correspondence discovery between the EDI messages of various standards by using XML Schema as the pivot format. This semi-automatic schema matching algorithm take two schemata of EDI messages as the input, compute the basic similarity between each pair of elements by comparing their textual description and data type. Then, it computes the structural similarity value basing on the structural neighbors of each element (ancestor, sibling, immediate children and leaf elements) with an aggregation function. The basic similarity and structural similarity values are used in the pair wise element similarity computing which is the final similarity value between two elements. The paper shows as well some implementation issues and a scenario of test for EX-SMAL with messages coming from EDIFACT and SWIFT standards.

1 Introduction

Electronic Data Interchange (EDI) is characterized by the possibility of sending/treating messages between information systems without any human intervention. The emergence of EDI enables companies to communicate easily (e.g. Send Orders, Funds Transfer, Acknowledgement of receipt, etc.). With growing business, many companies have to treat different type of messages and standards. Therefore, a large number of translations are needed in order to enable the communication between an enterprise and its suppliers and clients [21].

Although the use of XML has simplified the task of data exchange, the problem of data heterogeneity remains largely unresolved [17]. For the same kind of data, independent developers and systems often use XML syntaxes (i.e. messages) that have very little in common. For example, a Payment Order schema can generate an XML document where the date of the payment order looks like:
Whereas the encoding chosen by a partner defines the date of a payment order with:

```
<POrder>
  <Header>
    <POIssue Date = "30-07-04" />
  </Header>
</POrder>
```

These two documents contain the same data (date of payment order) but with two incompatible representation. Thus, establishing translation between different sources is a hard task without the presence of an expert which can identify the similarity between different elements of equivalent representations. In order to simplify this manual tedious and error-prone work, we suggest automating the similarity findings. We explore in this paper the development of an EDI/XML semi-automatic Schema Matching Algorithm. The algorithm uses XML Schema, as the pivot, to represent the schemas of EDI messages.

The paper is organized as follows. In the Section 2, we survey the literature for related works. We bring out the difficulties in existing approaches to suit EDI Schema matching. We present in Section 3 our similarity algorithm. We examine the used module and argue their usefulness. In section 4, we show some practical issues concerning implementation, scenario of test, and results. We wrap up our paper with future works and conclusion showing the assets, opportunities, and limits of our algorithm.

## 2 Related Works

In order to apply a translation between different representations, we should use two distinguished processes: matching and mapping. Firstly, matching process helps to identify the correspondent elements. Afterward a mapping process is required to express how the matched elements can be mapped. The input of the first is the schema or the branching diagram of the messages. The output is a list of matched elements. The input of the mapping process is the list of matched elements and the output is the individual mapping expression of target elements. We are only interested in this paper by the first process (i.e. similarity matching). Other works concerning mapping expression can be found in [22] and [18].

There is much literature on matching algorithms using learning module or thesaurus. These algorithms differentiate between being content based analysis (instance matching) XMapper [13], Automatch [2], and LSD [7], representation based analysis (schema matching) [12], [24], and [15], and usage based analysis (ontology matching) H-Match [5], Glue [8], and [3]. However, the algorithms with learning capabilities have a handicap concerning the needed training. Using thesaurus for improving the matching process is very interesting where schema’s elements have linguistically representative names. In all these approaches, we are only interested by representation based analysis since EDI branching diagrams, i.e. usage guide, are very likely to schemas.