# CSV2RDF: GENERATING RDF DATA FROM CSV FILE USING SEMANTIC WEB TECHNOLOGIES

**[1,2]S M HASAN MAHMUD, [3]MD ALTAB HOSSIN, [4,5]HOSNEY JAHAN, [1]SHEAK RASHED HAIDER NOORI, [1]MD. FOKHRAY HOSSAIN**

[1]Daffodil International University, Faculty of Science and Information Technology, Dhaka, Bangladesh

[2]University of Electronic Science and Technology of China, Department of Computer Science & Engineering, Chengdu- 611731, China

[3]University of Electronic Science and Technology of China, Department of Management Science & Engineering, Chengdu- 611731, China

[4]Sichuan University, College of Computer Science, Chengdu- 610065, China

[5]SEEDSlab (Software Intelligence and Data Science Research Group), Dhaka -1207, Bangladesh

E-mail: [1]{hasan.swe, drnoori, international}@daffodilvarsity.edu.bd, [3]altabbd@163.com,
[4]hosney.hstu08@gmail.com

## ABSTRACT

Recently, a large amount of Governments and public administrations data are stored on the Web in various file formats, mostly in the tabular data form such as Comma Separated Values (CSV) or Excel. CSV format is simple and practical, but it is difficult to express the relevant metadata such as data provenance, meaning of data fields, relationships between data fields, and user access approaches/rights, etc. In order to make the CSV data semantically structured, interoperable, accessible and reusable for various Web applications, they need to be extracted from the CSV files and converted into the Resource Description Framework (RDF) format that provides superior data assimilation and query functionality. In this paper, we focus on how the Semantic Web technologies are used to convert CSV data into RDF. Therefore, we present a method and techniques to parse the CSV file; the parsed CSV data are complemented with metadata annotations to generate the annotated tabular data model which is then converted into RDF triples. According to the conceptual correspondences between the CSV data model and RDF data model, we designed a set of algorithms to generate RDF triples from the CSV data. Our developed prototype tool, CSV2RDF, is used for evaluating the performance of the proposed method through real-world CSV datasets. The implementation and experimental outcomes demonstrate that our pro-posed method is feasible to generate RDF data from CSV datasets, with satisfactory performance on any size of data sets.

**Keywords:** *Comma Separated Values (CSV), Resource Description Framework (RDF), Annotated Tables, CSV to RDF Conversion, Metadata, Semantic Web.*

## 1. INTRODUCTION

The Semantic Web [1] is an extension of the current World Wide Web (WWW), which can provide a common mechanism for publishing, sharing and reusing data across application, enterprise, and community boundaries. It is a web of data (a.k.a. Web 3.0), which shows a more structured way of representing data that makes it easy to be processed and analyzed by the machine. It also helps to integrate data from different data sources using the Semantic Web technology (RDF, OWL, SKOS, SPARQL, etc.) and linked data principles. Recently, the World Wide Web consortium (W3C) has combined the e-Government and the Semantic Web technology, and launched Data Activity to create the web of data with different types of data format (e.g. CSV, XML, RDF, JSON, etc.). For this reason, government and private organizations started to publish massive amounts of structured data on the Web, where most of the published data are represented in the form of tabular data such as CSV or XLS [2, 33]. Examples are the data portals of the US government (data.gov) and the UK government (data.gov.uk) or the European Commission (open-data.europa.eu) as well as more than 200 local, regional and national data portal catalogues. In the CSV file, the data are stored as Comma Separated Values in a textual format and each row of the text is a data record, separated by commas [4]. The tabular data has divided into three levels, namely, the core tabular data model, annotated tabular data model, and grouped tabular data model [3]. The tabular data model is annotated with additional metadata and generates the annotated tabular data model. In the annotation process, the annotation contains information about the cells, rows, columns, tables, and groups of tables. In addition, metadata is a description object such as array properties, link proper-ties and natural language properties that describe an element of the model. Our research is inspired by the matter that CSV is an easiest and popular possible analytical format, offer flexible processing, and have two dimensional representations of data. Moreover, the CSV are increasingly prominent and have the potential to automatically convert into different data formats.

Despite recent advances in structured data publishing on the Web [5,38,39], the question arises, how data can be converted to semantically structured and described data, in order to make them easily discoverable and to facilitate data integration. Nowadays, many developers and researchers spend significant amount of time to develop a mapping application to transform published data into RDF for data source and format multiplication. Moreover, RDF [6, 8] is used to represent information and a standard model for data interchange on the web. Further-more, the RDF standard provides a model for representing resources in the form of a graph (directed, labeled graph), where a single resource is uniquely identified by a Uniform Resource Identifier (URI). In RDF statements, information is identified by triples subject-predicate-object. The subject and the object represent the resources, and the predicate indicates the relationship of those resources. Various tools and projects (see, for instance, the LOD2 Project) [7] have been launched aiming at facilitating the lifting of tabular data to semantically RDF data. Vast majority of the tools are for converting RDB to RDF and few tools support CSV to RDF conversion, to the best of our knowledge.

Thus, mapping CSV to RDF is the major creation of the Web of Data. But the W3C Working Group did not recommend any implementation for direct conversion of CSV2RDF. Lately, several existing tools or ongoing projects are implementing direct Mapping for CSV2RDF, including our proposed prototype. However, some tools adopt different methods and mapping techniques, making it difficult to share and reuse the conversion statements across applications or mapping engines. Most of them are not developed by the Semantic Web technologies and W3C recommendation standard. Using Semantic Web technologies (e.g. RDF) and W3C recommendation standard, this research focuses on a prototype system for converting CSV to RDF data. The core of the CSV to RDF conversion includes the CSV data model, annotated tabular data model, and RDF data model, as well as the approach for achieving the conversion based on the semantic correspondence between the three models. Consequently, CSV2RDF's prototype allows for users to input external metadata vocabulary and generate their customized annotated tabular data model by metadata modification feature. Our developed soft-ware system is fully compliant with the W3C Recommendation [8]. To evaluate the performance of our pro-posed approach, we conducted several experiments (e.g. time/space complexity of the algorithms) using the implemented prototype system with real-world CSV datasets. One of the most unique feature is metadata edit (add, edit and delete inputted or generated metadata) in our framework, as metadata edit is needed during RDF triple generation process. This feature also can be applied during converting both CSV and annotated data model process.

This paper is structured as follows: In section 2 we describe methods and tools related to our work. In section 3 we focus on our conversion method, working process and algorithms that covert CSV data into RDF. Section 4 explains the implementation and summarizes the experiment results of the method. Finally, sections 5 concludes the paper and identify future work.

## 2. RELATED WORKS

Converting tabular data to RDF is introduced by researchers at an increasing rate. Many techniques and tools have been developed to map non-RDF data to RDF. This part briefly describes related work that has support for tabular data. Research efforts in the field of Semantic data has followed several avenues: Early work by Behkamal et al. [9], Sarkar et al. [10] and LUCERO (LUCERO Project) [11] was concerned with publishing University datasets on the Web as linked data, data conversion process done by the D2R tool and RDF datasets are published on the Web. Rowe et al. [12] has done another similar research project Data.dcs which were aimed to generate Linked Data that describes the University of Sheffield's Department of

Computer Engineering. Also, this project proposes an approach to transform such legacy data into a RDF representation which is linked into the Web of Linked Data whereas previous similar research papers described the relational database (RDB). However, the researchers were working with University RDB datasets so those projects can convert only those universities datasets from RDB to RDF. In addition, Use cases and requirements based framework for the map-ping of RDB to RDF was proposed by Hert et al [34], where the authors report the state-of-art methods and applications at the same time.

Some research works are related to HMTL data conversation and generate the machine readable data. Li et al. [13] and Rowe et al. [12] presents an approach to convert the legacy data into Linked Data. The authors were working with legacy data in the HTML format and convert HTML to RDF data using filters pipeline approach. Finally, the links are established between the dataset and the discovered resources over the Web of Data using Linked Data principles [35]. Additional work by Coetzee et al. [14] deals with the tools development of converting legacy HTML datasets into RDF. Previously, HTML to RDF conversion done by the filter pipeline mechanism but here the authors used the SPARQL query language and the HTML Document Object Model (DOM). Another work by Haase et al. [15] presented software based on cloud's application-as-a-Service (PaaS) paradigm. Everyday a large amount of datasets are stored in government data portals and a part of these datasets is a collection of interrelated tabular data. This document specifies the effects of metadata on many researches where the researchers are working with conversation process from legacy government data to RDF. Ermilov et al. [2] developed tools for converting raw government data into high-quality machine readable data. They also developed an application for user-driven transformation and visualization of tabular data. Maali et al. [16] proposed a self-service Linked Government Data (LGD) for publishing, sharing and reusing. Both authors were working with government tabular data.

On the other hand, W3C Recommendation [8] defines the procedures and rules to be applied when converting tabular data into RDF. The tabular data are complemented with metadata annotations that describe its structure, the meaning of its contents and how it may form the resulting RDF. Tabular data are routinely transferred on the Web in a variety of formats, including variants on the CSV. W3C Recommendation document [3, 17] outlines a model for tabular data and the relevant metadata. The recommendation defines a vocabulary for metadata that annotates tabular data, which is used to provide metadata about tabular data at various levels, from individual cells within a CSV table to groups of tables and how they relate to each other. Due to Semantic data conversion importance to the Web of data, more specific state-of-the-art methods with description and supported input-output data formats of each method are presented in Table 1.

Consequently, several projects and tools has been developed for tabular2rdf2lod import and lifting process, including as tabular data to RDF compliant tools such as D2RQ [20, 21], RDF123 [22], Any23 (https://any23.apache.org/), RDF-RDB2RDF (https://metacpan.org/release/RDF-RDB2RDF), Tarql (https://github.com/tarql/tarql), Spread2RDF (https://github.com/marcelotto/spread2rdf), Triplify [23], as well CSV to Linked Data tools such as csv2rdf4lod[19], TabLinker (https://github.com/Data2Semantics/TabLinker). After preceding review, majority systems demand complex configuration and mapping specification, which need to regulate by the end user and often systems provide a less user friendly graphical user interface (GUI). In addition, most of the existing works mainly focused on handling general conversion method and a very few studies follow the latest standards published by the W3C's CSV Working Group. These limitations motivate us to convert the CSV data to RDF with the tabular data model, annotated tabular data model and RDF data model based on the W3C's standards. To overcome the existing issues, our CSV2RDF provides convenient installation principle, automatic mapping specification and friendly GUI to increase the accessibility of these capabilities.

## 3. PROPOSED METHODOLOGY AND IMPLEMENTATION OF CSV2RDF

### 3.1 Overview of the Proposed Methodology

In this part we present the proposed methodology for converting CSV data to RDF. The main ideas of the pro-posed methodology are as follows:

- The CSV syntax and semantics comply with the definitions in the IETF's RFC4180 document [4].

- The CSV data are complemented with metadata annotations that describe its structure, the meaning of its contents and how it may form part of a collection of interrelated CSV data by using the annotated tabular data model specified in the W3C.

*Table 1. Comparison of Related Works*

| Publication/Project | Input Data | | | | Output Data | | | Method | Description |
|---|---|---|---|---|---|---|---|---|---|
| | CSV | RDB | XML | Tabular | RDF | LOD | RDF+ | | |
| Ermilov et al. [27] | ✓ | | | | ✓ | | | Automatic | This research propose a formalization method of semantic mapping and transforming large scale tabular datasets to RDF from open government data catalog such as PublicData.eu. |
| Crotti Junior et al. [26] | ✓ | ✓ | | | ✓ | | | -- | This paper presents FunUL method, which describes uplift mapping and data transformation functions of CSV data into RDF. Also, the authors compared different types of methods and uplifting tools. The proposed functions are reusable. |
| Krataithong et al. [28] | | | | ✓ | ✓ | ✓ | | Automatic | The authors describe a unique approach to provide automatic open linked data service of RDF data which is generated from open government data portal (Data.go.th). Finally, the method was compared with two existing approaches. |
| Stadler et al. [29] | | ✓ | | | ✓ | | | Automatic | The RDB2RDF model and SML (Sparqlification Mapping Language) languages are presented which define the mapping on SQL VIEWS and SPARQL. Moreover, the RDF data are in R2RML form. |
| Lim et al. [36] | | ✓ | | | ✓ | | | Automatic | This is a Hadoop framework based RDB2RDF converter. The system generates RDF and OWL ontology based on RDB schema. Moreover, it generates endemic ontology to implement the direct mapping function using the rich semantic technology. Experiments were conducted on large scale datasets. |
| Umar et al. [37] | | ✓ | | | ✓ | | | Semi-Automatic | It's a semi-automatic R2R methodology; the proposed approach contains set of algorithms, mapping rules and a software architecture (RDB2RDF). This methodology is able to align the input and output data model. |
| Vahdati et al. [32] | ✓ | | ✓ | ✓ | | ✓ | | Automatic<br>Manual | The OpenAIRE is a European Open Access research platform (Datasets and publications). This Linked Data project supports HBase, CSV and XML data formats as input and convert them into RDF. Lastly, the RDF datasets is published on the web as Linked Open Data. |

**CSV (Comma-separated values), RDB (Relational Database), XML (Extensible Markup Language), RDF (Resource Description Framework), LOD (Linked Open Data)

### 3.2 Architecture and Process Workflow

Based on the proposed method, we have designed a prototype architecture, called CSV2RDF, and a converting process of the CSV to RDF data for CSV2RDF. The converting engine takes a CSV file and the metadata vocabulary as input, and produces the converted RDF triples as output. Finally, the resulting RDF triples are exhibited by the client browser or SPARQ query. The software architecture and process workflow of the main components are illustrated in Figure 1.

#### 3.2.1 CSV Parser

This module implements the CSV Parsing (**Algorithm 1**) algorithm. The CSV Parser performs a method (best practice of CSV data) for expressing tabular data adhering to the annotated tabular data model. The module processes a document that contains a CSV file to create an initial annotated tabular data model and extract embedded metadata to verify the compatibility table and metadata. During this parsing procedure, CSV file format follows the constraints of the IETF's RFC4180 document [4]. Based on the dialect description, the replacement error mode ensures that non-Unicode characters are replaced by U+FFFD (Unicode Character), strings column titles and cells. Sequentially, CSV data are read using the encoding and replacement error mode where the values of the CSV file contain the valid Unicode characters. To provide the row content, quoted values and list of the cell values in the CSV data, the CSV Parser performs several functions such as read row, read quoted values and parse row.

In CSV file process, if the row element begins with a comment prefix with values (not null), a string value will be added to the resulting array. If the string begins with the escape character imitated by the quote character, both the strings will be attached to the row element, quoted value and existent cell value. This module parses the CSV file using the FasterXML/jackson library in order to prepare them for generating annotated tabular data model and extract embedded metadata. Based on the CSV parsing algorithm, the CSV Parser module is de-signed and implemented to read and parse the CSV data model.

#### 3.2.2 Metadata Annotation Creator

The Metadata Annotation Creator implements the Metadata Annotation (**Algorithm 2**) algorithm. Conversion of tabular data to RDF data model, especially for CSV, generally requires supplementary metadata that describes the interpretation process of the data [30]. The module includes the format and structure of metadata documents (metadata vocabularies) and the process of generating annotated tabular data model from CSV file. In order to create annotations, the metadata document contains information about the groups of tables, tables, columns, rows, and cells. A description object (JSON format) describes an element of the annotated tabular data model (group of tables, table or column). In addition, file properties (indicate the table, column, row) are also applied to create an annotation (name, titles, dc: description etc) on the table or column in the data model. For generating the annotated tabular data model, the software processor should be started from a CSV file and the publisher should be linked to the CSV files directly; the processor can also be started from the metadata file. In the CSV data model, there are several methods that can be applied to add metadata such as overriding metadata, link header, default locations, site-wide location configuration and embedded metadata. For the method of locating metadata, the metadata is supplied within a single document to define the CSV data. Our module follows the overriding metadata method to supply the metadata for processing the tabular data and also follows the metadata embedded method.

If there is no metadata in the CSV file, processors must attempt to generate embedded metadata. A number of table data are combined together, and the annotation is added to get the set of table groups of metadata annotation. To verify the compatibility of the CSV file with metadata, there are several techniques that have been proposed such as tabular data object description compatibility, compatibility mode, and column compatibility to complete the compatibility verification process. Annotated tables are the significant form for further processing of the CSV data, such as validating, converting and displaying the tables.

#### 3.2.3 CSV2RDF Converter

The CSV2RDF Converter module performs the procedure of generating RDF triple (subject-predicate-object) from the annotated tabular data model. This RDF is an abstract syntax and the triples are serialized in a concrete RDF syntax (Turtle). The CSV to RDF conversion consists of two modes: Standard mode and Minimal mode. In the standard mode conversion procedure, the information collected from the cells of the CSV file contains detail about the rows, tables, and group of tables. We designed a CSV2RDF Converter module and CSV2RDF Conversion (**Algorithm 3**) algorithm according to the standard mode of conversion based on which our prototype system generates RDF triples. In this module, the RDF triple can be generated based on two conversion methods: CSV to RDF and JSON to RDF. The CSV to RDF conversion process describes the mechanism for transforming the CSV data. The transformed CSV data is then used for creating an annotated tabular data model with the metadata. The JSON to RDF conversion process describes the mechanism for trans-forming the json-ld dialect used for non-core annotation properties and comment properties creating from the metadata into RDF triples. This module calls the Sesame library in the implemented prototype in order to con-vert the RDF data model into a Turtle file.

### 3.3 Algorithm Design and Procedures

#### 3.3.1 CSV Parsing

**Algorithm 1** parses a CSV file, including the tables, columns, rows, cells and comment prefixes in the form of the tabular data model. Firstly, the CSV file is read using the encoding process, then

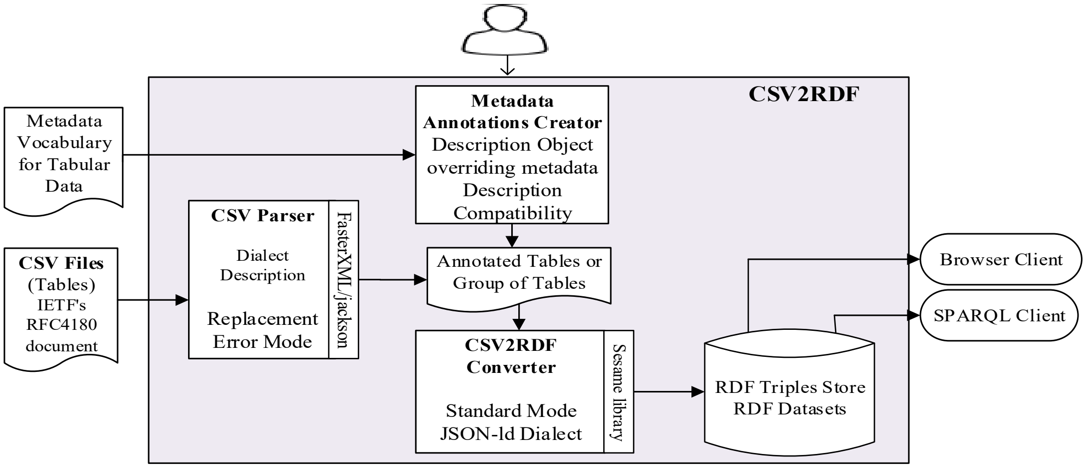


*Figure 1: Software Architecture of the proposed CSV2RDF Converter*

**Algorithm 1** CSV Parsing

```
Input:  CSV files.
Output: Parsed CSV table T.
 1:  initialize a CSVDataModel file;
 2:  get tableAnnotations;
 3:  if CSVDataModel is Unicode then read the file;
 4:  for each skip rows,
     4.1:  if (CommentPrefix ≠ 0) then add result value in CommentArray ;
 5:  for each header rows,
     5.1:  if (CommentPrefix ≠ 0) then add result value in CommentArray ;
 6:  while read the next row,
     6.1:  set SourceColumnNumber 1;
     6.2:  if (CommentPrefix ≠ 0) then  add result value in CommentArray;
           else parse row content to a cell list;
     6.3:  if cell list values is empty and skip blank rows is true then continue
           else get NewRow;
     6.4:  remove the SkipColumnsNumber and add to SourceColumnNumber;
     6.5:  for cell values index i=1….n,
           6.5.1:  if no new column C then add column C to table T;
           6.5.2:  if no new cell D then add cell D to the column C and row R;
     6.6:  add 1 to the SourceRowNumber;
 7:  return table T.
```

### 3.3.2 Metadata Annotation

The basic function of **Algorithm 2** is to add the metadata vocabulary with the parsed CSV data. If the attributes of the metadata are annotated, then those attributes will be used to annotate the table data. After starting the annotation with the CSV data, the metadata is supplied by the locating metadata method and normalizes them into a single descriptor. On the other hand, when the process starts with the metadata file, the normalizing method provides the metadata and verifies the compatibility of the metadata for each CSV data. Finally, Algorithm 2 invokes the foreign key and the primary key annotation for the parsed CSV data.

### 3.3.3 CSV2RDF Conversion

**Algorithm 3** converts the annotated tables into an RDF data model, where the annotated tables are obtained by the CSV Parsing (Algorithm 1) and Metadata Annotation (Algorithm 2) algorithms. The created RDF resources describes that the annotated tables and the RDF triples are generated from the rows, cells and tables of the an-notated tables. Finally, the RDF data model is established by using the RDF vocabulary defined in the W3C's specification “RDF Schema 1.1” [24] to build relationships between the table groups, table, rows, columns, cells, and metadata attributes.

```
Algorithm 2 Metadata Annotatioan
Input:  Parsed CSV data;
        Metadata vocabulary for tabular data.
Output:  Annotated tables.
 1:  initialize a parsed CSV data;
 2:  for each metadata property do
     2.1:  if property {"@context", "dialect", "@type", "tableSchema"} then continue processing the next attribute;
     2.2:  if the name of property is "@id" then add annotation for table, marked as "id", annotated value for the
                property value;
     2.3:  else add a annotation to the table, annotating the name of the property with the value of  property;
 3:  read the tablemetadata property tableSchema to get the CSV data schema;
 4:  read the tableSchema property "columns" to get the column description array columns;
 5:  read table's columns annotation to get the column array of table data tableColumns;
 6:  read the size of the array tableColumns columnSize;
 7:  for i = 1... columnSize do
     7.1:  for each property schemaColumns[i] do
             add a annotation for the tableColumns[i], the name of the property, the value of the  property value;
 8:  add the foreign key annotation (tableSchema, referenced rows, columnReference, referencedTable) to the  CSV
     file;
9:  add the primary key annotation (tableSchema, KeyColNames, titlesrow) to the table's primary key annotation;
10:  return annotated tables.
```

```
Algorithm 3 CSV2RDF Conversion
Input:  Annotated tabular data.
Output: RDF datasets.
 1:  initialize annotated tabular data from a Annotate tables (single table or group of tables) module;
 2:  Call rows, cells and tables of the annotated tables;
 3:  Call JSON-LD to RDF (json-ld-api);
 4:  for single annotated table,
     4.1:  if single table have cell, row and column then generate RDF triples (subject, predicates, object;
 5:  for group of annotated tables,
     5.1:  if group of tables table have cell, row and column then generate RDF triples (subject, predicates, object;
 6  return d RDF triple in the RDF datasets.
```

### 3.4 Prototype Implementation

The CSV2RDF software system is a Java based desktop application developed using Java (1.7.0 J2SDK), Apache Jena and third party kits (JavaFx library, FasterXML/jacson, google/gson, Sesame) on the IntelliJ IDEA (15.0.2). Our proposed algorithms generate RDF file when the CSV data are used as an input in the tool and the resulting RDF can be displayed in both RDF/XML and Turtle serialization. The java functions which are related to CSV2RDF implementation are listed in Table 2. The main working step is as follows:

(1) The CSV Parser uses a FasterXML/Jackson library, especially, the Jackson package, to read and parse the elements of a CSV file, then stores the parsed CSV data in a Java data structure (Java class) which is listed in bellow. The CSV Parsing algorithm is responsible to parse the CSV data in order to prepare them for annotation. (2) The Metadata Annotations Creator uses Java class methods corresponding to Metadata Annotation algorithm in order to generate the annotated tabular data model (single table or group of table) from the parsed CSV data. The Metadata is added to the CSV data which describes how the data should be explained. (3) The CSV2RDF Converter uses Java class methods compatible with a CSV2RDF Conversion algorithm to execute the mapping from CSV to RDF and JSON to RDF in the triple syntax. The JSON to RDF conversion applies Deserialized JSON-LD to RDF (json-ld-api) algorithm, which also outputs in equivalent to RDF triples. Finally, the algorithm calls a Sesame Rio class model through an output stream to display the resulting RDF data. More specifically, Figure 2 represents the Java functions with description related to the prototype implementation.

## 4. EXPERIMENTS AND DISCUSSION

### 4.1 Experimental Design and Set-up

The implemented prototype is installed and run on our Lab's Web server. We carried out CSV data conversion experiments with our CSV2RDF tool on a Lab PC with configurations as Intel(R) Xeon(R)

CPU E5640/ 2.66 GHz, 12M Cache; Memory 16 GB (8×2 GB). In order to verify the effectiveness of the proposed approach, this research uses a number of CSV files, metadata files, and RDF triples. For this experiment, the transformation includes: CSV to Annotated Tabular Data Model and Annotated Tabular Data Model to RDF.

*Table 2: List of the implemented Java functions.*

| Module | Functions | Description |
|---|---|---|
| CSV Parser | `FileInputStream` | It represents the input of the CSV file. |
| | `setDefaultDialect` | This function indicates a dialect description in the metadata file which is associated with the CSV data. |
| | `BufferReader` | Buffer Reader reads the CSV data model using the encoding process. |
| | `ObjectMapper` | A parser class of the CSV document which is provided by the FasterXML/Jackson module. |
| | `embeddedMetaData` | It creates a metadata structure from the header row of the CSV file. |
| | `sourceRowNumber`<br>`headerRowCoun` | Both functions count the source row number of the CSV file. |
| | `createObjectNode` | This method is called to create new rows, columns and cells. |
| Metadata Annotation | `metaData` | It describes the metadata object of annotated tabular data. |
| | `createAnnotatedTables` | The normalizing method to design the metadata normalization algorithm that achieves the createAnnotatedTables from the CSV metadata. |
| | `convertedMetaData` | It converts the CSV data to annotated tabular data with metadata. |
| | `tabularGroupModel` | This function indicates the annotation of a group of tables. |
| CSV2RDF Converter | `RepositoryConnection` | It used for creating links with the repository. |
| | `coennetion.Add` | It is a link that emits the RDF triples as the subject, predicate and objects. |
| | | This function creates the RDF resources from Annotated tabular data model. |
| | `tableUrlString` | It specifies the source CSV data file URL for the current table based on the url annotation. |
| | `outputStream` | It calls the Sesame Rio class to pass the RDF output model. |

```
public class Main { // CSV class main class
      public static final String DISLAY_DELIMETER = "\t"; // Class id
      public static void main(String[] args) {
        String dialctPath = "D:\\hasan\\dialect.json";//Json File data path
        String tabularFilePath = "D:\\hasan\\tabular_data_file.txt";//input
        CSVParser csvParser = new CSVParser();
        ResultTable resultTable = null;//Resultant table for parsed CSV
        try {
         resultTable = csvParser.parse(dialctPath , tabularFilePath);
          } catch (IOException e) {
              e.printStackTrace();
          }
          System.out.println(); // parsed tabular data display
          System.out.println("the tabular data is as follows:");
          for( Row row : resultTable.getT().getRows() ){
              for( Cell cell : row.getCells()){
               System.out.print(cell.getStringValue());//call to get string
               System.out.print(DISLAY_DELIMETER); //display the parsed CSV
              }
              System.out.println();
```

*Figure 2: Parsed CSV data in a Java data structure (Java class)*

### 4.1.1 Experimental Dataset

For the experimental purpose, we have collected some experimental data sets from the CSVW Implementation Report [25] of the W3C Recommendation and US Government data portal (data.gov). In the experimental stage, the CSVW Implementation Report data sets are used for testing the transformation process and the US Government (data.gov) Datasets are used for measuring the algorithmic efficiency. Our downloaded CSV and JSON test data files are saved in the experimental PC's disk, as shown in Table 3. Also, The US Government test datasets are listed in Table 4. Table 3 lists the properties of the five test datasets (CSV and JSON) which are tested (CSV2JOSN and JSON2RDF) using our tool and obtained PASS for all the datasets in the testing phase of the case study (Subsection 4.2).:

Thus, we have selected the following five datasets from the W3C's document of CSVW Implementation Re-port [25].

- Test Data 01 (Simple table): The simplest tabular data without metadata information.
- Test Data 012 (tree-ops example with directory metadata): The processor applies overriding metadata for processing the CSV file; otherwise, the processor should collect metadata from the link header or site-wide configuration.
- Test Data 023 (dialect:header=false): For true, set header row count to 1, otherwise set false. The URL and dialect description object of this CSV file is recorded.
- Test Data 032 (events-listing.csv example): To show the list of the CSV data record, the "events-listing" is from metadata. Here, metadata file of the CSV file is in JSON format. The CSV file URL; dialect description objects and tabular data model are recorded in "csv-metadata.json".
- Test Data 039 (valid inherited properties): This data have various combinations of valid inherited proper-ties. The metadata file of the CSV file is in JSON format. The "test039-metadata.json" contains tabular data language, text direction, the default value of the cells, cell type, and tabular data model.

*Table 3: List of the Experimental Datasets for Metadata Annotation and RDF Conversion Test.*

| Test Datasets | Test Type | Properties of Test Datasets | Test Status |
|---|---|---|---|
| **Test 001**<br>Simple table | Annotated<br>test/JSON test | Name: test001.csv<br>(9 rows x 2columns) | PASS |
| **Test 012**<br>tree-ops example with directory metadata | Annotated<br>test/JSON test | Name: tree-ops.csv<br>(3 rows x 5 columns) | PASS |
| **Test 023**<br>dialect:header=false | Annotated<br>test/JSON test | Name: tree-ops.csv<br>(3 rows x 5 columns) | PASS |
| **Test 032**<br>events-listing.csv example | RDF test | Name: csv-metadata.json<br>(3 rows x 5 columns) | PASS |
| **Test 039**<br>valid inherited properties | RDF test | Name: test039-metadata.json<br>(2 rows x 10 columns) | PASS |

*Table 4: List of the Experimental Datasets for Conversion Time Performance test.*

| Test Datasets | Properties | Domains |
|---|---|---|
| Mile Markers | 843 kb | Transportation |
| Youth Tobacco Survey (YTS) Data | 2749 kb | Health |
| Allegheny County Sheriff Sales | 2269 kb | Local Government |
| New Business List - June | 1825 kb | Education |

### 4.1.2 Conversion Time Performance Tests

In order to verify the time performance of the CSV2RDF data converter, four data sets are tested in the experiments. The experimental CSV files consist of data with different numbers of rows, columns and sizes. We con-ducted the experiment by means of the number of CSV rows, the phased running time of CSV parsing, annotated table generation, and RDF triple conversion as well as the total running time. In this experiment, to minimize data dependence of our approach, ten stage (based on row numbers) experiment was performed on four datasets. Therefore, every data set (Mile Markers, Youth Tobacco Survey (YTS) Data, Allegheny County Sheriff Sales and New Business List - June) was sequentially split into ten parts based on the rows of the CSV file. The method is applied four times to obtain the running time. Note that, the parameters of the algorithm were same for four datasets. The ten stages experimental results of the proposed approach on the four datasets are listed in Tables 5–8. After employing the proposed algorithms (CSV Parser-Algorithm 1, Metadata Annotation- algorithm 2 and CSV2RDF Conversion- Algorithm 3) to the test data Mile Markers, we achieve the less conversion time of CSV parsing, annotated table generation, RDF triples conversion and total running time of 100.4, 178, 198.4 and 465.8 milliseconds respectively. On the Test data Youth Tobacco Survey (YTS), our algorithms return the average CSV parsing, annotated table generation, RDF triples conversion and total running time of 218.4, 318, 347.8 and 884.2 milliseconds, respectively. Similarly, we can see from Table 7 that the CSV parsing, annotated table generation, RDF triples conversion and total running time on the test data Allegheny County Sheriff Sales reach 258.4, 378, 397.8 and 1034.2 milliseconds respectively. In Table 8, the averages of CSV parsing, annotated table generation, RDF triples conversion and total running time on the test data New Business List - June are 289.4, 403, 427.8 and 1116.2 millisecond respectively. Therefore, from Tables 5–8, we can note that the effective automatic approach linked with the algorithms is effective for con-verting CSV datasets.

*Table 5: Running Time of Data Conversion by algorithms on Mile Markers Dataset*

| CSV Rows | CSV Parsing | Annotated table | RDF Triples | Total Time |
|---|---|---|---|---|
| 2500 | 56 | 100 | 112 | 253 |
| 10000 | 78 | 127 | 155 | 350 |
| 20000 | 100 | 176 | 198 | 464 |
| 30000 | 123 | 217 | 242 | 572 |
| 50000 | 145 | 270 | 285 | 690 |
| Average | 100.4 | 178 | 198.4 | 465.8 |

*Table 6: Running Time of Data Conversion by algorithms on Youth Tobacco Survey (YTS) Dataset*

| CSV Rows | CSV Parsing | Annotated table | RDF Triples | Total Time |
|---|---|---|---|---|
| 2500 | 176 | 240 | 262 | 678 |
| 10000 | 188 | 267 | 305 | 760 |
| 20000 | 220 | 316 | 345 | 881 |
| 30000 | 243 | 357 | 392 | 992 |
| 50000 | 265 | 410 | 435 | 1110 |
| Average | 218.4 | 318 | 347.8 | 884.2 |

*Table 7: Running Time of Data Conversion by algorithms on Allegheny County Sheriff Sales dataset*

| CSV Rows | CSV Parsing | Annotated table | RDF Triples | Total Time |
|---|---|---|---|---|
| 2500 | 216 | 300 | 312 | 828 |
| 10000 | 228 | 327 | 355 | 910 |
| 20000 | 260 | 376 | 395 | 1031 |
| 30000 | 283 | 417 | 442 | 1142 |
| 50000 | 305 | 470 | 485 | 1260 |
| Average | 258.4 | 378 | 397.8 | 1034.2 |

*Table 8: Running Time of Data Conversion by algorithms on New Business List – June Dataset*

| CSV Rows | CSV Parsing | Annotated table | RDF Triples | Total Time |
|---|---|---|---|---|
| 2500 | 236 | 325 | 342 | 883 |
| 10000 | 265 | 352 | 385 | 1002 |
| 20000 | 284 | 401 | 425 | 1110 |
| 30000 | 314 | 442 | 472 | 1228 |
| 50000 | 348 | 495 | 515 | 1358 |
| Average | 289.4 | 403 | 427.8 | 1116.2 |

Figure 3(a), 3(b), 3(c) and 3(d) illustrated the line graph of the algorithmic running time which is performed by CSV2RDF tool on the four data sets. Analyzing the results, we have the following findings:

a) The comparison of the running time of the algorithm modules indicates that, the running time is mainly spent on the RDF triple conversion (including metadata information) whereas the CSV parsing and annotated table generation spent a short time, which implies the efficiency of our CSV2RDF conversion approach.

b) The conversion running time of RDF triples generation (Test Data: 04-case New Business List – June, 1825 kb) is the most, while another RDF triples generation (Test Data: 01- Mile Markers, 843 kb) has the least run-time. The run-time usually depends on the properties (Kilobytes) of the data sets.

c) The running time of each module as well as the total running time exhibits approximately a linear growth rate as the size of the CSV file (i.e., the number of CSV rows) increases. As shown, when the row amount of the CSV file is low, then the running time is less; it gradually increases based on the row amounts.

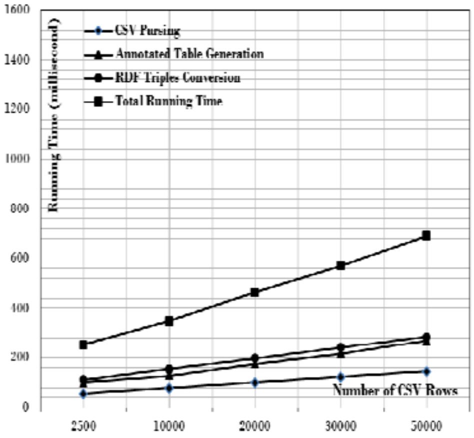


*Figure 3: (a) Conversion running of algorithmic on the Mile Markers dataset;*

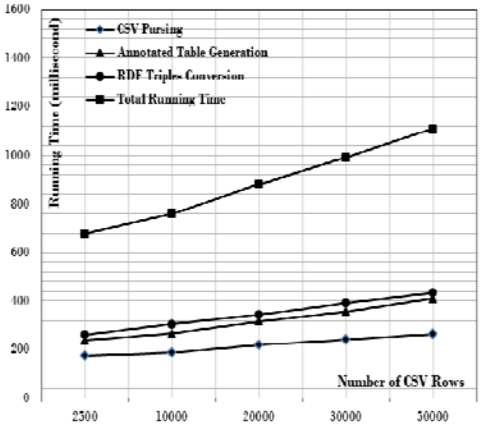


*Figure 3: (b) Conversion running of algorithmic on the Youth Tobacco Survey (YTS) Dataset;*

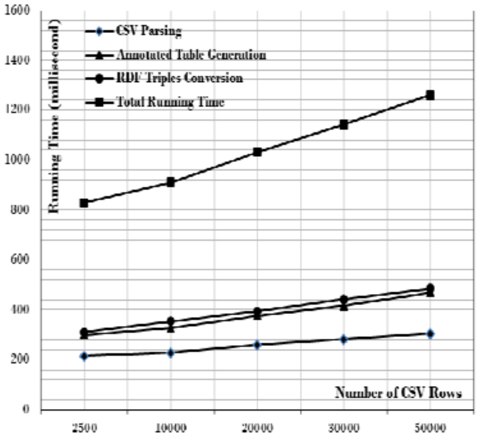


*Figure 3: (c) Conversion running of algorithmic on the Allegheny County Sheriff Sales;*

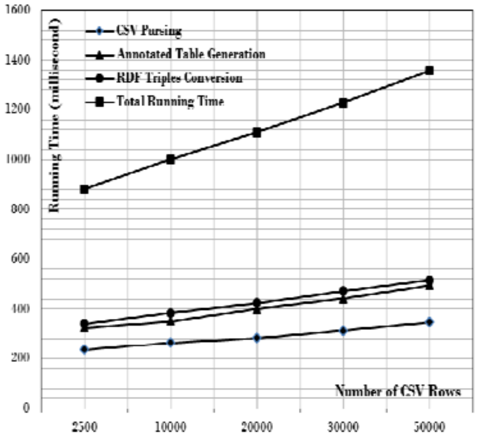
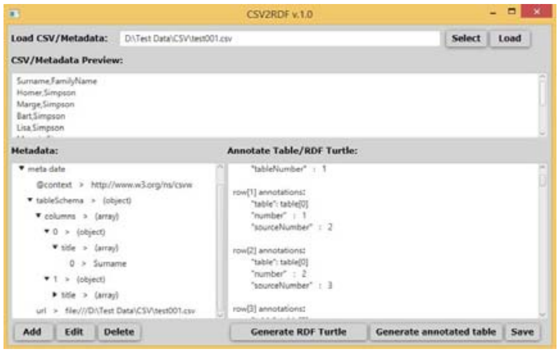


*Figure 3: (d) Conversion running of algorithmic on New Business List – June dataset;*

### 4.2 Case Study

Using the CSV2RDF conversion engine, several case studies have been conducted to demonstrate the performance of our proposed system. To reduce space, few case studies are given in Figure 4. Figure 4 (a) represents the tree structure of the file “test001.csv” in the CSV to annotated tabular data generation stage. The CSV file parses the file to generate an embedded metadata which is displayed as a tree structure on the left side of the prototype text area. In the metadata tree structure, the object properties and array properties are identified by the “objects” and “arrays”. The root node “Metadata” represents top object of the metadata (Object Top-Level). From the prototype GUI, metadata can be modified (add, edit and delete operation) using the metadata editor panel to add a sub-attribute "primaryKey" and attribute value "Surname" with the "tableSchema" attribute.



Load CSV/Metadata: D:\Test Data\CSV\test001.csv

CSV/Metadata Preview:

```
Surname,FamilyName
Homer,Simpson
Marge,Simpson
Bart,Simpson
Lisa,Simpson
```

Metadata:

```
▼ meta date
    @context > http://www.w3.org/ns/csvw
  ▼ tableSchema > (object)
    ▼ columns > (array)
      ▼ 0 > (object)
        ▼ title > (array)
            0 > Surname
      ▼ 1 > (object)
        ▶ title > (array)
    url > file:///D:\Test Data\CSV\test001.csv
```

Annotate Table/RDF Turtle:

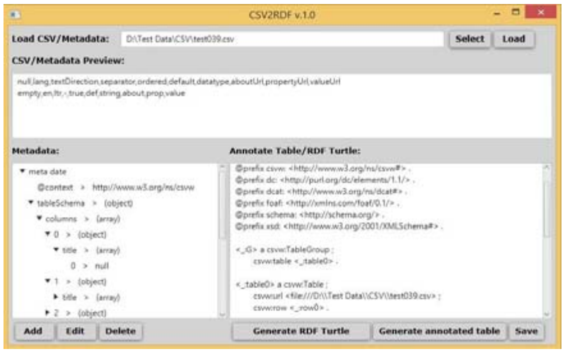
```
    "tableNumber" : 1

row[1] annotations:
    "table": table[0]
    "number" : 1
    "sourceNumber" : 2

row[2] annotations:
    "table": table[0]
    "number" : 2
    "sourceNumber" : 3

row[3] annotations:
```

*Figure 4: (a) Screenshot of CSV2RDF- the resulting annotated tabular data;*



D:\Test Data\CSV\test039.csv

```
null,lang,textDirection,separator,ordered,default,datatype,aboutUrl,propertyUrl,valueUrl
empty,en,ltr,-,true,def,string,about,prop,value
```

```
▼ meta date
    @context > http://www.w3.org/ns/csvw
  ▼ tableSchema > (object)
    ▼ columns > (array)
      ▼ 0 > (object)
        ▼ title > (array)
            0 > null
      ▼ 1 > (object)
        ▶ title > (array)
      ▶ 2 > (object)
```

```
@prefix csvw: <http://www.w3.org/ns/csvw#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<_:G> a csvw:TableGroup ;
    csvw:table <_:table0> .

<_:table0> a csvw:Table ;
    csvw:url <file:///D:\\Test Data\\CSV\\test039.csv> ;
    csvw:row <_:row0> .
```

*Figure 4: (b) Screenshot of CSV2RDF- the resulting RDF data*

The annotated tables are shown in the right side of the prototype. The resulting RDF triple generated from the annotated table is displayed in Figure 4 (b).

## 5. CONCLUSION

In this research, we presented a formal approach for automatic conversion of the CSV datasets to RDF. Our complete approach includes: (1) the design of an algorithm to read and parse a CSV file; (2) the design of an algorithm to attach metadata annotations to CSV data using the annotation technology; (3) the design of an algorithm to generate RDF from the annotated tabular data model; (4) the development of a tool implementing the above three algorithms, guided by the W3C recommendations, and providing an automatic process of annotating and converting. The use of W3C recommendations proved to be a simple, flexible and expedite way of defining the conversion process. These theoretical and technical descriptions were incorporated in the CSV2RDF converter, and the design and implementation of the CSV2RDF converter are also a relevant contribution of this work. To validate the proposed approach, the CSV2RDF tool was used for converting several preexisting CSV files which were collected from Open Government data portal (data.gov) and CSVW Implementation Report. At present, our developed tool support W3C recommended CSV files. In future, we plan to develop a tool which will support all types of CSV files. The experimental results show that the total running time of CSV2RDF conversion exhibits approximately a linear growth rate as the size of the CSV file increases. In addition, our implementation and case studies demonstrate that the proposed approach is effective and feasible. Future work includes more data sets and experiments to compare the expressiveness and performance of our CSV2RDF converter with the existing tools such as RDF123, csv2rdf4lod and Tarql. Moreover, we plan to improve our approach to provide RDF publishing service as Linked data on the Web using Linked Data principles.

## REFRENCES:

[1] J. Domingue, D. Fensel, J. A, Hendler, Introduction to the Semantic Web Technologies, In: Domingue John, Fensel Dieter & James A Hendler (eds.), Handbook of Semantic Web Technologies, pp. 3-41, Springer-Verlag Berlin Heidel-berg, 2011.

[2] I. Ermilov, S. Auer, C. Stadler, CSV2RDF: User-Driven CSV to RDF Mass Conversion Framework, in: Proc. 2013 ISEM, Graz, Austria, 04-06 September 2013.

[3] J. Tennison, G. Kellogg (Editors), Model for Tabular Data and Metadata on the Web, W3C Recommendation, 17 Decem-ber 2015. http://www.w3.org/TR/tabular-data-model/. [Accessed 16 December 2017]

[4] Y. Shafranovich, Common Format and MIME Type for Comma-Separated Values (CSV) Files, RFC 4180, October 2005. http://tools.ietf.org/html/rfc4180. [Accessed 13 November 2017]

[5] L. Sikos, Mastering Structured Data on the Semantic Web: From HTML5 Microdata to Linked Open Data, Apress: New York, USA, 2015. ISBN 978-1-4842-1049-9.

[6] R. Cyganiak, D. Wood, Lanthaler, M (Editors), RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, 25 February 2014. http://www.w3.org/TR/rdf11-concepts/. [Accessed10 December 2017].

[7] S. Auer, V. Bryl, S. Tramp (Editors), Linked Open Data -- Creating Knowledge Out of Interlinked Data: Results of the LOD2 Project, Lecture Notes in Computer Science, Volume 8661, Springer International Publishing, 2014..

[8] J. Tandy, I. Herman, G. Kellogg (Editors), Generating RDF from Tabular Data on the Web, W3C Recommendation, 17 December 2015. https://www.w3.org/TR/csv2rdf/. [Accessed 5 November 2017].

[9] C. Bizer, R. Cyganiak, T. Heath, How to Publish Linked Data on the Web, 2007. http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/. [Accessed 20 December 2017].

[10] A. Sarkar, U. Marjit, U. Biswas, Linked data generation for the university data from legacy database, International Journal of Web &

Semantic Technology (IJWesT), Volume 2, No. 3, pp. 21-31, 2011. DOI.10.5121/ijwest.2011.2302.

[11] LUCERO Project (2011), available at: http://lucero-project.info/lb/about/ (Accessed 7 December 2017).

[12] M. Rowe, F. Ciravegna, Data.dcs: converting legacy data into linked data, in: proc. 2010 Linked Data on the Web Work-shop, World Wide Web Conference, Raleigh, North Carolina, USA, 27 April 2010..

[13] J. Li, Y. Zhao, A case study on linked data generation and consumption, in: proc. 2008 Linked Data on the Web (LDOW), Beijing, China, 22 April 2008..

[14] P. Coetzee, T. Heath, E. Motta, SparqPlug: Generating linked data from legacy HTML, SPARQL and the DOM, in: proc. 1st Workshop on Linked Data on the Web (LDOW2008), Beijing, China, 22 April 2008.

[15] P. Haase, M. Schmidt, A. Schwarte, The information workbench as a self-service platform for linked data applications, in: proc. Second International Workshop on Consuming Linked Data (COLD'11). Bonn, Germany, Volume 782, pp. 119-124, 2011.

[16] F. Maali, R. Cyganiak, V. Peristeras, A publishing pipeline for Linked Government Data, in: proc. 9th Extended Semantic Web Conference (ESWC2012), Heraklion, Greece, May 27-31 2012, pp. 778-792..

[17] J. Tennison, G. Kellogg (Editors), Metadata Vocabulary for Tabular Data, W3C Recommendation, 17 December 2015. https://www.w3.org/TR/tabular-metadata/. [Accessed 12 October 2017].

[18] M. Fiorelli, T. Lorenzetti, M. T. Pazienza, A. Stellato, A. Turbati, Sheet2RDF: A Flexible and Dynamic Spreadsheet Import & Lifting Framework for RDF, in: Proc. 2015 International Conference on Industrial, Engineering & Other Appli-cations of Applied Intelligent Systems (IEA/AIE 2015), Seoul, Korea, 10-12 Jun, 2015. pp. 131-140.

[19] T. Lebo, G.T. Williams, Converting governmental datasets into linked data, in: proc. 6th International Conference on Semantic Systems (I-SEMANTICS '10), Graz, Austria, 13 September, 2010, pp. 381–383, 2010.

[20] C. Bizer, R. Cyganiak, D2RQ — lessons learned. Position paper for the W3C Workshop on RDF Access to Relational Databases, 08 September 2007. https://www.w3.org/2007/03/RdfRDB/papers/d2rq-positionpaper/. [Accessed 16 October 2017].

[21] V. Eisenberg, Y. Kanza, D2RQ/update: updating relational data via virtual RDF, in: proc. 21st World Wide Web Confer-ence Lyon, France, 16 – 20 April, 2012, pp. 497-498..

[22] L. Han, T.W. Finin, C.S. Parr, J. Sachs, A. Joshi, RDF123: from spreadsheets to RDF, in: proc. 2008 International Se-mantic Web Conference (ISWC), Karlsruhe, Germany, October 26-30, 2008, pp. 451–466..

[23] S. Auer, S. Dietzold, J. Lehmann, H. Hellmann, D. Aumueller, Triplify: light-weight linked data publication from rela-tional databases, in: proc. 18th International Conference on World Wide Web, Madrid, Spain, 20-24 April 2009, pp. 621-630.

[24] D. Brickley, R.V. Guha (Editors), RDF Schema 1.1, W3C Recommendation 25 February 2014. https://www.w3.org/TR/rdf-schema/. [Accessed 10 December 2017]

[25] G. Kellogg, CSVW Implementation Report, W3C Document 28 October 2015. http://w3c.github.io/csvw/publishing-snapshots/PR-earl/earl.html. [Accessed 11 December 2017]

[26] A.C. Junior, C. Debruyne, R. Brennan, D. O'Sullivan, An evaluation of uplift mapping languages, International Journal of Web Information Systems, volume 12, No 4, 2017, pp. 405-424. DOI: 10.1108/IJWIS-04-2017-0036. .

[27] I. Ermilov, S. Auer, C. Stadler, User-driven Semantic Mapping of Tabular Data, in: proc. 9th International Conference on Semantic Systems (I-SEMANTICS '13), Graz, Austria, 4-6 September, 2013. pp. 105-112..

[28] P. Krataithong, M. Buranarach, N. Hongwarittorrn, T. Supnithi, A Framework for Linking RDF Datasets for Thailand Open Government Data Based on Semantic Type Detection, in: proc. 18th International Conference on Asia-Pacific Digi-tal Libraries (ICADL 2016), Tsukuba, Ibaraki, Japan, 5-9 December 2016..

[29] C. Stadler, J. Unbehauen, P. Westphal, M. Sherif, J. Lehmann, Simplified RDB2RDF Mapping, In Workshop on Linked Data on the Web. (2015).

[30] S. M. H. Mahmud, M. A. Hossin, H. Jahan, S. R. H. Noori and T. Bhuiyan, CSV-ANNOTATE: Generate annotated tables from CSV file, 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD),

Chengdu, 2018, pp. 71-75. doi: 10.1109/ICAIBD.2018.8396169.

[31] A. Dimou, D. Kontokostas, M. Freudenberg, R. Verborgh, J. Lehmann, E. Mannens, S. Hellmann, R. Walle, Assessing and Refining Mappings to RDF to Improve Dataset Quality, in: proc. 14th International Semantic Web Conference (ISWC), Bethlehem, PA, 10-15 November 2015, pp. 133 – 149.

[32] S. Vahdati, F. Karim, J. Huang, C. Lange, Mapping Large Scale Research Metadata to Linked Data: A Performance Comparison of HBase, CSV and XML, in: proc. 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, pp. 133-149.

[33] M. Skjæveland, E. Lian, I. Horrocks, Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data, in: proc. 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, pp. 162–177.

[34] M. Hert, G. Reif, H. Gall, A comparison of RDB-to-RDF mapping languages, in: proc. 7th International Conference on Semantic Systems (I-Semantics), Graz, Austria,7- 9 September 2011. pp. 25-32.

[35] C. Bizer, T. Heath, T. Berners-Lee, Linked Data—the story so far, International Journal on Semantic Web and Information Systems, volume. 5, no. 3, July-September 2009, pp. 1-22, doi:10.4018/jswis.2009081901.

[36] K-B, Lim, H-G. Jun, H-J. Kim, Semantics preserving MapReduce process for RDB to RDF transformation, Int. J. Metadata, Semantics and Ontologies, Volume. 10, No. 4, pp. 229–239.

[37] S. Umar, P. Gayathri, C. Anil, N. Priya, R. Srikanth, Design of Rivalize and Software Development to Convert RDB to RDF, In: Satapathy, S; Bhateja, V; Das, S. (eds), Smart Computing and Informatics. Smart Innovation, Systems and Technologies, volume 77, pp. 255-263.

[38] S. M. H. Mahmud, M. A. Kabir, O. A. M. Salem and K. N. G. Fernand, The comparative analysis of online shopping information platform's security based on customer satisfaction, 2016 5th International Conference on Computer Science and Network Technology (ICCSNT), Changchun, 2016, pp. 157-161. doi: 10.1109/ICCSNT.2016.8070139

[39] R. Hossain, S. M. H. Mahmud, M. A. Hossin, S. R. H. Noori and H. Jahan, PRMT: Predicting Risk Factor of Obesity among Middle-Aged People Using Data Mining Techniques, Procedia Computer Science Volume 132, pp. 1068-1076. https://doi.org/10.1016/j.procs.2018.05.022