# AUTOMATICALLY CONVERTING TABULAR DATA TO RDF: AN ONTOLOGICAL APPROACH

Kumar Sharma[1], Ujjal Marjit[2]*, and Utpal Biswas[3]
[1] Department of Computer Science and Engineering, University of Kalyani, Kalyani, West Bengal, India
[2]Center for Information Resource Management (CIRM), University of Kalyani, Kalyani, West Bengal, India
[3]Department of Computer Science and Engineering, University of Kalyani, Kalyani, West Bengal, India

## *ABSTRACT*

*Information residing in relational databases and delimited file systems are inadequate for reuse and sharing over the web. These file systems do not adhere to commonly set principles for maintaining data harmony. Due to these reasons, the resources have been suffering from lack of uniformity, heterogeneity as well as redundancy throughout the web. Ontologies have been widely used for solving such type of problems, as they help in extracting knowledge out of any information system. In this article, we focus on extracting concepts and their relations from a set of CSV files. These files are served as individual concepts and grouped into a particular domain, called the domain ontology. Furthermore, this domain ontology is used for capturing CSV data and represented in RDF format retaining links among files or concepts. Datatype and object properties are automatically detected from header fields. This reduces the task of user involvement in generating mapping files. The detail analysis has been performed on Baseball tabular data and the result shows a rich set of semantic information.*

## *KEYWORDS*

*Ontology, Tabular Data, CSV, Semantic Web, RDF, Linked Data.*

## 1. INTRODUCTION

Information residing in relational databases and delimited file systems are inadequate for reuse and sharing over the web. Since, they do not follow commonly set principles for data modeling and representation. Data stored on such formats are easy to read and process by the computer. However, these data still suffer from lack of uniformity, heterogeneity as well as redundancy throughout the web. This is mainly due to the inadequacy in designing the information schema. Sometimes, the system designers do not follow the systematic approach in modeling schema. They use different terms for same meaning, or single term for different meaning. This results in carrying out different semantics for same term and affects the shared understanding. They do not follow the proper class hierarchies. Sometimes the relationship among different classes is ignored. By following such practices, the data is stored as it is, by not considering the cases for future issues. As time goes by, the information residing on such systems starts growing and complexity increases. As a result, it leads in data management and difficulty in understanding the information schema to the future designers. Moreover, the problem in data reusing and sharing increases. The stored data suffers from lack of semantic relationship among different concepts. System designers tend to fix such issues while modifying the existing schema by adding more attributes and concepts or altering them. However, this still increases the elaborateness and liable to produce errors as well as breaks existing features. A suitable approach for solving this problem

is use of Ontology and RDF (Resource Description Framework). Ontology is the basic building block of any information system. It is mainly used for constructing knowledge bases and helps to extract the useful information from unstructured, semi-structured as well as structured data. RDF on the other hand, is the data-modelling framework for the Semantic Web technology. It describes resources that belong to well-defined domain ontology.

Here, ontology based approach has been proposed for extracting domain specific knowledge stored in CSV files. Existing approaches [1, 2] mainly dealt with single file conversion based on mappings. Either user provides the mapping or the program itself automatically creates it. This mapping is allowed to be modified by the user for further refinement. In this article, a slightly different approach has been proposed. Instead of dealing with mapping files, the domain ontology associated with CSV files has been generated. It has been observed that, it is the knowledge base, which matters the process of transformation. Thus, to capture the information stored in legacy data storage, the understanding of the background knowledge is necessary. In existing approaches, the idea is to map a CSV row-column to RDF property using existing ontology classes. User interaction is required for these approaches. Users have to manipulate mapping files manually and provide the appropriate matching between concepts. We argue that, such type of mapping techniques cause redundancy and inconsistency in resulting data. Since, end-users may not have the detail knowledge of the underlying technologies. Therefore, we have proposed an approach where the domain ontology is automatically created from a set of CSV files and attempts to reduce the barrier of providing mapping files. Using domain ontology, the conversion process learns about how or what information is to be mapped to extract meaningful resources. It generates link between files corresponding to their RDF as well as outside resources by adhering to the Linked Data principles [3].

The structure of this article is as follows. In section 2, we provide the background information of the related technology and section 3 presents the literature review. The motivation for this work is presented in section 4. Section 5 & 6 deal with the proposed approach. Section 7 shows some experimental results. In section 8, we discuss limitations and improvements of the proposed work and finally section 9 concludes the work.

## 2. BACKGROUND INFORMATION

In the proposed work, our main objective is to convert tabular data into structured data as RDF, from flat representation to semantic representation. In the following subsections we discuss the nature of tabular data and the role of ontology in designing various systems.

### 2.1. Tabular Data

Since the beginning of software applications, CSV file format has become a standard way for exchanging data between applications. The CSV file is supported by most of the business and enterprise applications. In product based applications CSV files are used to store exported data from the relational databases. The exported data, in turn, is provided to some other product based or business applications for importing data into underlying databases. Similarly, with the help of CSV or any delimited text files, various applications communicate with each other by importing and exporting data to or from their underlying databases. In general, a CSV file is consisted of five components such that,

$$F_{csv}=\{h,R,F,d,q\} \quad (1)$$

Where 'h' is a line of header, 'R' is a set of records and 'F' is a set of fields in each record. The header and each record terminate at line break. A record consists of fields separated by delimited character 'd' such as comma ",". Each record must contain same number of fields and each field may contain enclosing character 'q' such as double quote. The header comprises the names corresponding to each field. Hence, the header and each record contain exactly the same number of fields, forming them as matrix, such that,

$$F_{csv[m]} = \begin{matrix} r_1c_1 & r_1c_2 & r_1c_n \\ r_2c_1 & r_2c_2 & r_2c_n \\ r_mc_1 & r_mc_2 & r_mc_n \end{matrix} \quad (2)$$

In the above Equation (2), the components "$r_1c_1$", "$r_2c_2$" are called cells. A cell is an intersection of fields and records, which exactly holds the data value. The horizontal arrangement of such cells is called the row and the vertical arrangement of the cells is called a column. From all these, the header line is separate from the actual data. Header line is consisted of a set of names corresponding to each field. The header name actually resembles the attribute or property of a table. In fact, the whole CSV file resembles a table in a relational database. Each row represents the instance of a table. A column represents a set of values for a particular property of an instance. And the cell is the data value of the property of an instance. Like this the CSV components are mapped to the components of a table in a relational database.

## 2.2. Ontology

Ontology is the main building block of any intelligent system. It describes a domain by providing a set of commonly agreed terms and concepts. Ontology describes semantic of the data and provides a uniform way to enable communication by which different parties can understand each other [4]. Euzenat et al. [5] define Ontology as "a formal explicit specification of a shared conceptualization". Over the last few years, ontologies have been widely used in artificial intelligence, knowledge engineering, natural language processing and the Semantic Web. It groups all the concepts, properties and relations in a hierarchical manner, forming the taxonomy of the conceptual terms. The properties describe various behaviours of a concept, which tells exactly what a resource consists of. In real life, this resource is an instance of some concepts or classes defined in well-defined domain ontology. The behaviour of the resources can be of any thing, which is termed as value. Ontology serves value as either literal or object values. Literal and object values actually differentiate between the data-type and object properties. However, both of them relate things in either ways. The object property is the one, which relates instances of similar or different concepts. The object property builds the link between resources. In fact, the Semantic Web technology is able to create links among resources through object properties, which is popularly known as Linked Data [3]. Linked Data allows linking resources from one network domain to another forming a network of semantic graphs.

The most important benefit of using ontology is that, it provides reusable and shareable knowledge to the domain users. Domain users can reuse an existing knowledge base or domain ontology to describe similar resources. Ontology is the basis for solving various types of problems such as poor communication, defining specification or requirements of a system and reusing resources. These problems arise due to lack of shared understanding [6]. Ontology has also been used for requirement management by capturing the engineering design knowledge [7]. It has been used in product life cycle management for sharing product knowledge [8], collaborative design [9] and software engineering [10]. In General, Ontology is composed of five

components [4] - Concept (C), Relations (R), Properties (P), Axioms (A) and Instances (I). Such that,

$$Ont=\{C,R,P,A,I\} \quad (3)$$

Where, C is a set of concepts and R is set of relationships among C. P is the set of properties belonging to concepts, which are shared by concepts and a set of instances (I). By this, we understand that, in a domain there exist certain concepts, which are related to each other using relationship properties. Concept represents a thing in a domain and it is a collection of objects sharing common properties. An Instance is a member of a concept. Relations bring relationship between two individuals or concepts using a property, exposing their domain and range. Axioms (A) are used for specifying constraints on the data values.

## 3. LITERATURE REVIEW

An ontology based information extraction process has been presented in [11]. Here, ontology is used to guide the information extraction process from the unstructured text. The extracted information is then stored into RDF triple form. In the field of delimited text files, the mapping-based approaches have been applied so far. RDF 123 [1] is based on map graphs. Map graph maps a particular spreadsheet row to a row-graph. It produces row-graphs by taking column values at $i^{th}$ row and the row number "i" as input. For each row, the row-graph is created and merging of all row-graphs produces the final RDF graph. Ermilov et al. [2] proposed an approach based on mapping from tabular data to RDF. The mapping files are automatically generated. To prevent errors the mapping files are allowed to be edited by the users. Eventually, the mapping files are used to map corresponding tabular data to RDF. An incremental enhancement based approach has been proposed by Lebo and Williams [12]. In this approach, first of all, CSV to simple RDF data is generated, called the raw conversion. Iterative enhancements are then applied to the simple RDF data to improve the quality of the dataset. Casting values to datatypes, restructuring relationship and linking to the external data sources improve the quality of the dataset. Spreadsheet-to-RDF wrapper has been implemented by Langegger and Wöß [13], for transforming spreadsheet data to arbitrary RDF graphs through mapping specification and template graphs. It supports mapping of a portion of RDF triples to particular cells of a spreadsheet file.

An automatic approach has been proposed for converting tabular data to direct Linked Data [14]. It is based on mapping column headers to the target ontology in the knowledge base such as DBpedia, Freebase, WordNet and Yago. The tabular data involved are mainly open government data stored in native file formats, such as excel sheet. It also discovers implicit relations between tables and identifies the semantic classes associated with the columns. Heuristic Baseline System [15] has been used for predicting appropriate ontology to map every column header to a class. Using predicted class, the table cell values are linked to entities from the Linked Data sources. Mulwad et al. [16], using existing Knowledge-Base (KB) discuss entity extraction from tabular data to RDF. This approach assigns the table headers to existing class labels using appropriate ontology. It also links cell values to entities and generates links among table columns.

Apart from this, mapping based techniques have been applied in other format as well for converting legacy data into RDF. Spanos et al. [17] have stated that ontologies are the main vehicle of knowledge in the field of Semantic Web technology and presented a survey on various approaches for transitioning relational databases into the Semantic Web. They have listed different approaches based on new ontology, existing ontology, database reverse engineering, and domain schema ontology. Likewise, Lin et al. [18] have used database reverse engineering

method for ontology extraction. Telnarova [19] has expressed that the relational database is the source of ontology creation and presented an approach for creating ontology using relational databases to ontology mapping. Mona et al. [20] have proposed an approach for constructing ontologies from relational databases. In this approach, the ontological concepts have been generated by extracting knowledge from the relational schema and database components such as constraints and triggers. In [21], RDF Mapping Language (RML) is applied for mapping heterogeneous data to RDF. RML is a language, which provides generation of customized mapping from heterogeneous data sources to RDF data models. Linked Data knowledge-base (such as DBpedia) is used to extract encyclopaedia knowledge contained in Wikipedia [22]. Existing vocabularies (statistical vocabularies) have been used to publish statistical data coming from tabular data sources as Linked Open Data [23]. A huge amount of data from Library domain has also been translated into Semantic Web domain [24]. Other works include extraction of XML data into RDF, which are discussed in [25, 26, 27]. Krextor [25] stands for KWARC RDF extractor based on EXtensible Stylesheet Language (XSL). Transitioning MARC 21 data into RDF has been presented by Sharma et al. [28]. It is based on the Linked Data principles and the conversion process deals with direct mapping from MARC 21 fields to corresponding RDF properties.

## 4. MOTIVATION

In real world, tabular data is exposed over the web for sharing and interlinking with other resources. Subsequently, these data are stored in legacy data format. They are not easy to import into the web for the purpose of reasoning & analysing over data. For reasoning & analysing, machine should be able to understand semantics of the data. Let us consider an example of retail environment, which motivates using such linking process. In retail environment, there exist several entities such as Product, Product-Group, Supplier, Staff, Customer and many more. We consider only three entities for quick understanding – Product, Product-Group and Supplier as shown in the Table 1.

Table 1. Retail Domain

(Product Group)

| Product Group Code | Name | Description |
|---|---|---|
| GR01 | Accessories | Retail |
| GR02 | Catering | Hospitality |
| GR03 | Gift | Retail |

(Supplier)

| Supplier ID | Name | Address |
|---|---|---|
| SB100 | ABC Product | Kolkata |
| SB101 | Premium Suppliers | New Delhi |

(Product)

| Product Code | Product Group Code | Name | Description | Supplier ID |
|---|---|---|---|---|
| P100 | GR01 | Gift Paper | Gift Paper | SB100 |
| P101 | GR02 | Samosa | Snacks | SB101 |
| P103 | GR03 | Watch | Wrist Watch | SB100 |

Product entity is related to product-group and supplier using (Many-to-One) relationship, i.e., a product-group or supplier can have many products. In addition to this, product is related to many other entities, which include sales, transaction logs, inventory and customer. Table 1 shows a description regarding said three entities. It reveals that the product entity needs to be linked to product-group and supplier using ‘Product Group Code’ and ‘Supplier ID’ respectively. In retail environment, everyday new entities are manipulated using various kinds of data items. These include order, order items, notes, product modifiers and order instructions. Traditional softwares

are used to generate data for these entities and they are stored in comma-separated files. These softwares produce huge amount of data and stored in spreadsheet files. Though, logically they are linked, but physically they are separated from each other. It is difficult to analyse these data manually. When need arises to comply new technologies, these files need to be processed using newly designed software. So, legacy data can be moved to a newer version of the web. Moreover, the format of data storage needs to be changed. To bring them into other formats (e.g. XML or RDBMS), user performs some sort of conversion process. Though, we can convert data from XML or RDBMS and expose them in the web. However, we need direct approach for converting CSV to RDF without any barrier. This reduces extra job to be performed in between the CSV data and RDF data. As well as, it reduces time and efforts while writing two different types of conversion process.

## 5. THE PROPOSED DESIGN ARCHITECTURE

The design architecture is shown in the Figure 1. It starts with a set of CSV files and the domain name. The domain name is required for naming the domain ontology. The successive concepts corresponding to each CSV file are created using this domain ontology. The domain ontology identifies and gives taxonomy of the classes and their essential properties. Hence, in the initial stage, the domain ontology is created by understanding the nature of domain from the user. CSV files may contain unsupported characters in place of data values, such as the presence of delimited character. Sometimes the empty header as well as redundant header names may also present in the file. Such anomalies, if present, are removed.

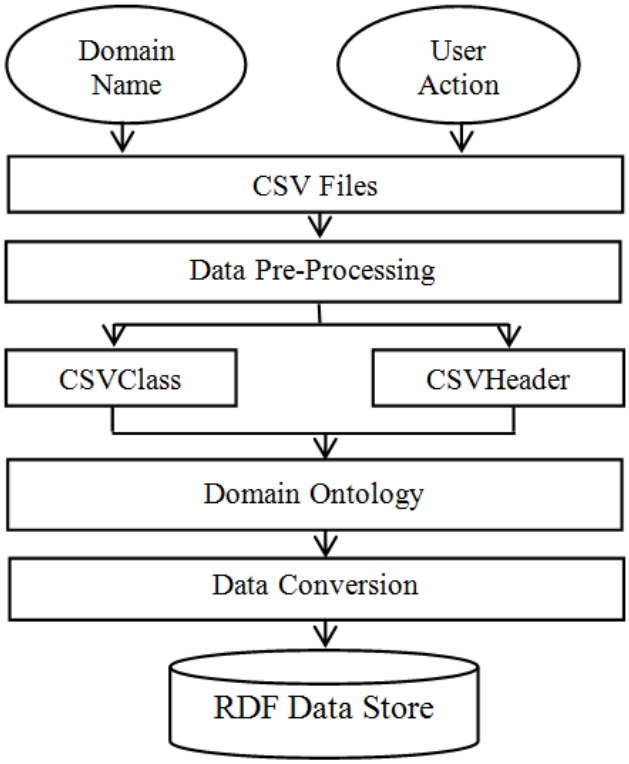


Figure 1. Design Architecture

In the proposed work, delimited characters are replaced by some non-delimited characters, duplicate header names are replaced by the character '_' followed by frequency number, which is

calculated based on the occurrence of the duplicate header name. The domain ontology is developed with the help of two canonical components – CSVClass and CSVHeader as shown in Figure 2. These components hold useful information regarding CSV file, headers and data. They also store required ontology class and class property. A CSVClass corresponds to an ontology class and a CSVHeader corresponds to a header and property of the class. CSVClass also holds the set of CSVHeaders and the file reference to the entire data. The data is retrieved from the disk on demand to reduce the memory overload while processing each file. CSVHeader on the other hand, holds the header name, index information, data-type and corresponding ontology property.

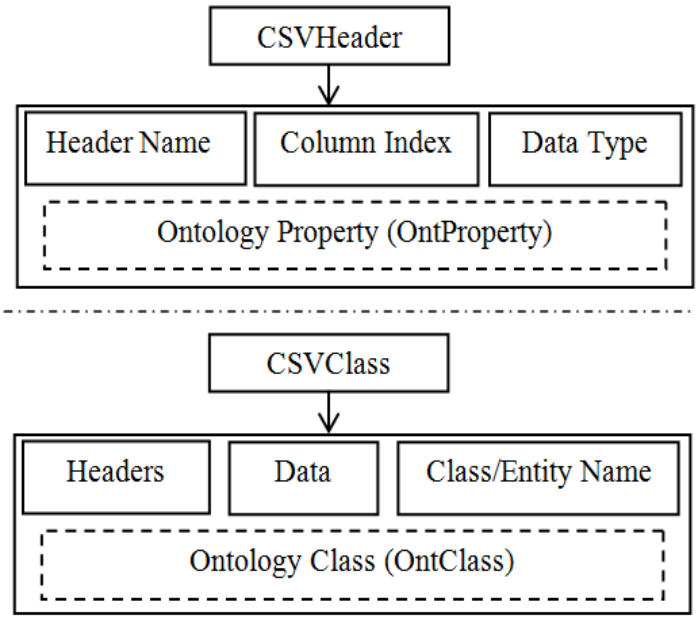


Figure 2. CSV to Canonical Components

As soon as the domain ontology is created, it is presented to the user for revising and making any changes. If user wants to change the class name, property name, property's data range then user defined data is stored in the canonical components. This is achieved while providing user interface to the user, as it helps to determine what input from the user needs to be applied before creating domain ontology. As soon as the domain ontology is created, the conversion process starts converting tabular data to RDF. The RDF model holds the entire RDF instances equivalent to CSV rows.

## 6. THE PROPOSED APPROACH

As we outlined above, the tabular data is structured into rows and columns consisting of headers and records. The first line in the CSV file is called header and the rest are called records separated by delimited character. The proposed approach is based on the idea of conceptual mapping between the CSV components and the ontology components. A CSV file is mapped to an ontology class. A record is always mapped to the instances of the class, headers are mapped to the properties, and the corresponding fields are mapped to the actual values. After all, the role of domain ontology is to supply required knowledge of the concepts and their terms.

Since, the CSV files are the members of the same domain, a CSV file cannot store records of multiple relations in one file. However, it can store reference data of related tables in separate

columns. Thus, they can still retain the relationship when they are exposed into a newer file format. For this, it requires identification of the kind of properties each CSV files contain. Thus, users can have multiple files belonging to a particular domain. In the above example, in section 4, Retail is a domain and Products, Product-Groups and Suppliers are concepts of Retail. In a domain, concepts may relate to each other. In other sense, they possesses link among them. Such relationship, if exist, needs to be retained during conversion. Consider again, the example mentioned above in the Table 1, from three tables Product, Product-Group and Supplier the domain ontology has to be generated. As shown in the Table 1, there are two header names in the Product table - 'Product Group Code' and 'Supplier ID', which are also the member of the headers of Product-Group and Supplier table respectively. When this set of CSV files is converted into RDF, it should be able to generate links between Product, Product-Group and Supplier tables. Now, the challenging task is in determining kind of property. A concept can have both datatype and object properties. Therefore, a header name could be either datatype or object property. Determining datatype and object properties is a tricky task. It needs looping through each file, headers and data values determining their range and identifying if any of the file holds any relationship. To understand how we achieve this, let us consider following generic tables. Let us consider three tables M1, M2 and M3 as shown in the Table 2. These three tables need to be converted to so call models in terms of Semantic Web technology retaining their header names as datatype property or object property. This is the utmost purpose of this work.

Table 2. Generic Tabular Data

**M1**

| a | b | d |
|---|---|---|
| a1 | b1 | d1 |
| a2 | b2 | d2 |
| a3 | b3 | d3 |

**M2**

| d | a | e |
|---|---|---|
| d1 | a1 | e1 |
| d2 | a1 | e2 |
| d3 | a2 | e3 |

**M3**

| g | h | e |
|---|---|---|
| g1 | h1 | e1 |
| g2 | h2 | e3 |
| g3 | h3 | e3 |

Above three tables have headers and data. Each of them has three header names such as (a, b, d), (d, a, e) and (g, h, e) for M1, M2 and M3 respectively. All of the above three tables contain a header field of another table. As we can see, M1 has the header name 'd' which is also a header name of 'M2', such that d∈M1 and d∈M2. We can say that 'd' as an object property, means a relationship holds between M1 and M2. For determining the range of an attribute, first, their data range has to be identified. That is, identification of the datatype of a header name is leading step. For every file and headers encountered, this step has been performed. Following sub-sections described how we have achieved this.

## 6.1. Process of Determining Datatype Property

In ontology, all the properties of a concept must have data range or datatype. Since, a property describes a particular behaviour of an instance; hence the value it contains must be valid and well formed. Therefore, in designing ontology, determination of property and its datatype is important. Now, CSV files do not contain any information regarding datatype of the headers. Though, they hold valid data, which means that, data into CSV file must be coming from well-designed data

store, such as relational databases. However, going back to its source data store is burdensome. We therefore, have added automatic datatype detection algorithm, which reduces the burden of providing datatype metadata. The algorithm is based on analysing non-empty data values of a single column for a header. Each data values are analysed based on predicting regular expression of a given datatype as well as values for each header are fetched and subsequently their size is checked. If the size equals to certain threshold then apply regex pattern-match to the data. If the size is greater than a threshold for all datatypes except for string or text, then it automatically treats as string or text datatype. For example, to satisfy a Boolean datatype the size should be in between 1 and 5, considering the possible values for Boolean as – 'T', 'F', '1', '0', 'True', 'False', 'Yes', and 'No'. If the size is less than or equal to 5 then apply regex pattern match for detecting whether the data is of Boolean type. The following definition has been derived for finding the datatype of a header.

*Definition 1: Let D be a data type and* $H = \{h_1, h_2, \ldots, h_n\}$ *be a set of headers. If* $V = \{v_1, v_2, \ldots, v_n\}$ *a set of non-empty values for each* $h_i \in H$ *then data-type for* $h_i$ *is D iff* $\forall v_i \in D$ *and* $SizeOf(v_i) \leq SizeOf(D)$.

Applying above definition following algorithm is derived to determine the datatype of a header.

**ALGORITHM 1:** GetDataRange

**Input**: A CSV Header 'h' and its dataset V = {$v_1$, $v_2$, ....., $v_n$}
**Output**: Data Range for 'h'

```
1:   Range ← NIL;
2:   D ← {d1, d2, ....., dn}; //a set of data-range
3:   for each d in D do
4:       matched ← True;
5:       Range ← d;
6:          P ← RegexPatternFor(d);
7:              for each v in V do
8:                  if !v.matches(P) OR SizeOf(v)>SizeOf(D) then
9:                      matched ← False;
10:                       Range ← NIL;
11:                   break;
12:                 end if
13:             end for
14:         if matched = True then
15:             break;
16:         end if
17:   end for
18: return Range;
```

## 6.2. Determining Object Property

There are two types of properties such as datatype properties and object properties. Datatype properties link instances to data values called literals whereas object properties link instances among instances of the same or different domain. So, determination of object property reveals the relations between concepts. Additionally, it identifies how an instance is related to another instance. In this work, we consider having object properties means 'has' relationship between instances. Due to this reason, loop over each file and their headers are extended. Consider tables described in Table 2, if a value for header 'd' is present in M1 and M2 both, then, it is said that two tables are in 'has' relationship. However, we need to determine which of these tables do actually hold the 'has' relationship. Since, the value for header 'd' is present in both M1 and M2.

*Definition 2*: *Let V be the set of all non-empty values for header h in a table T1 and V′ be the set of all non-empty values for header h in another table T2. We say that T1 contains some values of T2 or T1 has T2, iff, $V \neq V'$ or $V \subset V'$ (i.e., V must not equal to V′ or is a subset of V′) AND data-range of all values of header h is same in both T1 and T2.*

The following situations may be raised: all the values are same in both the headers, and yet they possess the 'has' relationship. This creates problem in determining the actual range of the property. Since, both source and target resources contain the same data and both of them could be the range of the property. For such cases, we present this conflict to the user.

Using above definition, the following algorithm has been derived to find the object range of a header.

**ALGORITHM 2:** GetObjectRange

```
   Input: A CSVClass 'c', Header 'h' and its dataset V = {v1, v2, ....., vn}
   Output: An object-range for 'h'
1:    Found ← False;
2:    Range ← NIL;
3:      for each c′ in getAllCSVClasses() do
4:        if c ≠ c′ then
5:          for each h′ in c′.getAllHeaders() do
6:            if h′.equals(h) then
7:              V1 ← c′.valuesForHeader(h′);
8:                if hasDuplicates(h(V)) AND hasUnique(h′(V1)) then
9:                  Range ← c′;
10:                 Found ← True;
11:                 break;
12:               end if
13:             end if
14:           end for
15:       if Found = True then break;
16:     end for
17:  return Range;
```

## 6.3. Generating Domain Ontology (Classes)

Once the datatype and object properties are determined, the ontology classes are created. The canonical components are used to create final ontology classes. From each CSVClass members, ontology classes are created. Each CSVHeader members are converted into the ontological properties, which are then assigned to the individual classes. Jena ontology framework is used to generate ontology, classes, and their properties. Following algorithm has been considered to generate the ontology of a CSV file through CSVClass.

**ALGORITHM 3:** GenerateOntologyClass

```
   Input: A CSVClass 'c' and a domain ontology 'Ont'
   Output: An ontology class or concept
1:  OntClass concept ← Ont.CreateClass(c.getClassName());
2:    for CSVHeader h in c.getHeaders() do
3:      DataRange D ← GetDataRange(h, c.valuesForHeader(h));
4:        DataTypeProperty P ← Ont.createDatatypeProperty(h.getHeaderName());
5:        P.addRange(D);
6:        P.addDomain(concept);
7:        ObjectRange O ← GetObjectRange(c, h, c.valuesForHeader(h));
```

```
8:          if O ≠ NULL then
9:             h.setRemovedProperty(P);
10:            concept.dropIndividual(P);
11:            ObjectProperty newP ← Ont.createObjectProperty(O.getName());
12:            newP.addRange(O);
13:            newP.addDomain(concept);
14:         end if
15:       end for
16:  return concept;
```

## 6.4. Generating RDF Resources

The datatype properties, object properties and ontology classes are generated using the Algorithm 1, 2 and 3 respectively. In the first phase, from tabular data, the RDF dataset has been generated. The RDF dataset do not contain any links among other resources during the execution of first phase. For each CSV file to RDF, an asynchronous job is allocated, which a job tracker controls. The job itself is a converter, which takes only CSVClass and domain ontology as parameters. The datasets are stored into RDF files as soon as they are created, and removed from in-memory models to release the memory occupied by the large datasets. The conversion job is also released from the memory as soon as it finishes converting data and writing RDF dataset into file.

## 6.5. Linking Resources

Linking resources of CSV files is the ultimate goal of this work. The links are generated among the resources of different concepts in the same domain. For this, help of 'removedProperty' of CSVHeader has been taken. A 'removedProperty' is a property which is common to both domain and range concepts. For example, for the table M1, as discussed above, the 'removedProperty' is 'd' and the original property is 'hasM2'. For 'hasM2' the range would be in M2, which is looked up using 'removedProperty' 'd' in M2. For linking, again, the looping through each RDF resource is needed, as shown in the Algorithm 4. For every encountered resource, check is made if there is a need of adding relationship. If so, referenced RDF model is searched for the target resource and the link is established.

**ALGORITHM 4:** LookUpLinks

```
    Input: A CSVClass 'c' and a domain ontology 'Ont'
    Output: Generate links in the RDF model of 'c'
1:      for each CSVHeader h in c.getHeaders() do
2:        If h.IncludeRelationship() then
3:          CSVClass range ← h.GetRangeClass();
4:            Model  rangeModel ← range.GetRDFModelFromDisk();
5:            Iterator I1 ← c.model.getResources (h.removedProperty);
6:            while I1.hasNext() do
7:              Resource R1 ← I1.nextResource();
8:              Property P1 ← R1.GetProperty(h.removedProperty);
9:               Iterator I2 ← rangeModel.getResources(h.removedProperty);
10:                while I2.hasNext() do
11:                  Resource R2 ← I2.nextResource();
12:                  Property P2 ← R2.GetProperty(h.removedProperty);
13:                  if P1.Equals(P2) then
14:                    R1.addProperty(h.property, R2);
15:                    break;
16:                  end if
17:                end while
```

```
18:                 end while
19:             end if
20:     end for
```

## 7. RESULT

The experiments have been made using Lahman's Baseball Database [29]. The Baseball Database contains statistics data on pitching, hitting and fielding from major baseball league from 1871 to 2014. It includes data from major baseball leagues such as American and National, Players League, and Federal League. The database was created by Sean Lahman, aiming in making baseball statistics freely available to the researchers and public users. Data are found in the following formats: MS Access, SQL Server, and Comma Delimited Versions (CSV). Comma delimited data has been collected for the five consecutive years – 2010, 2011, 2012, 2013 and 2014. The database consists of the following main entities: Master, Batting, Pitching and Fielding which contains player information, batting, pitching and fielding statistics respectively.

Table 3. Characteristics of Input Data

| Dataset Version | No of Files | Total Size (MB) | Total no. of Rows |
|---|---|---|---|
| Version 2010 | 24 | 11.9 | 155030 |
| Version 2011 | 21 | 10.4 | 149558 |
| Version 2012 | 21 | 10.5 | 156738 |
| Version 2013 | 21 | 10.0 | 159794 |
| Version 2014 | 21 | 10.5 | 174517 |

In addition, the database is supplemented by 20 other entities, which contain various statistical data such as yearly statistics and standings, post-season batting statistics, franchise information, and awards won by players etc. Table 3 shows the basic characteristics of the baseball CSV datasets. The complete conversion process is automatic. However, we provide user interface for editing datatype and object properties. Users can also edit the whole domain ontology in an editor window. The editor window allows user to change the concept name, properties name, data type of the properties, and can change the target object range of a property. The dataset have been tested and verified before processing to ensure that they do not contain any unsupported data. Starting from data pre-processing, processing and conversion to RDF, following phases have been performed.

### 7.1. Data Pre-Processing

Data pre-processing is needed to remove some unwanted characters or strings. In CSV file, most of the column data are quoted by some characters, such as, double-quote or single-quote. The baseball dataset contained quoted characters only in few datasets. Proceeding with such data produces unpredictable errors, which could stop the conversion process. The delimited text files should not contain the delimited text at the end of row line. Presence of this will lead to errors. The following pre-processing has been incorporated to reduce the error:

1. Removal of unsupported quoted characters.
2. Removal of delimited text at the end of each row line.
3. To make sure of all the files contain same delimited text and quoted text.
4. Correction of related header names in each CSV file.

## 7.2. CSV to Direct RDF

First of all the, CSV files have been converted into RDF without any datatype detection and linking among resources. That means, all the headers are directly converted into RDF properties serving all of them as string datatype. Table 4 shows result & statistics for converting CSV data to RDF. The result in Table 4 shows that for each CSV dataset maximum time to generate RDF dataset is 4 minutes for the CSV data whose files size is 12MB. Each RDF dataset contains near about 150K subjects having 2900K RDF triples.

## 7.3. Datatype Detection

Data type detection is performed for each CSV file. For each CSV header, the nature of corresponding column data have been analysed and their data is detected. In the review process, user might want to decide to change the datatype of certain properties. This facility has been provided, as we list entire ontology in the editor window.

Table 4. Result & statistics for converting CSV data to RDF

| Dataset Version | No. of RDF Statements | Conversion Time (Min) | Total no. of RDF Resources |
|---|---|---|---|
| Version 2010 | 2817935 | 3.35 | 155030 |
| Version 2011 | 2878556 | 2.63 | 149558 |
| Version 2012 | 2959789 | 2.98 | 156738 |
| Version 2013 | 2816041 | 3.40 | 153177 |
| Version 2014 | 2901396 | 2.70 | 166619 |

By default the editor window shows the detected data type along with additional data types for users to select. When the revised ontology is finalized, the properties are processed again, so that users input can be applied. Figure 3 shows the time taken for detecting data type and building the ontology for each dataset.

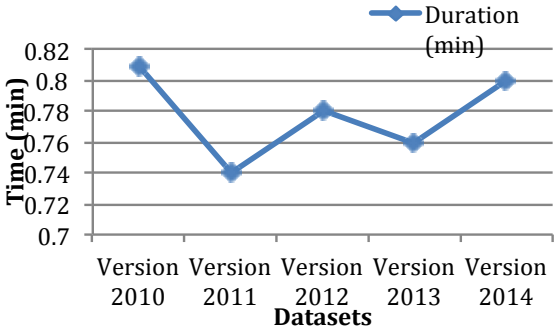


Figure 3. Time to detect datatype

## 7.4. Linking CSV Files

Each CSV files may contain links to one another. For linking such CSV files, we used Algorithm 4, determining the object range and the domain. Links are generated among CSV files in the same domain. Link to outside resources have been omitted to avoid long duration in processing each resource and looking up similar or related resources in the web.

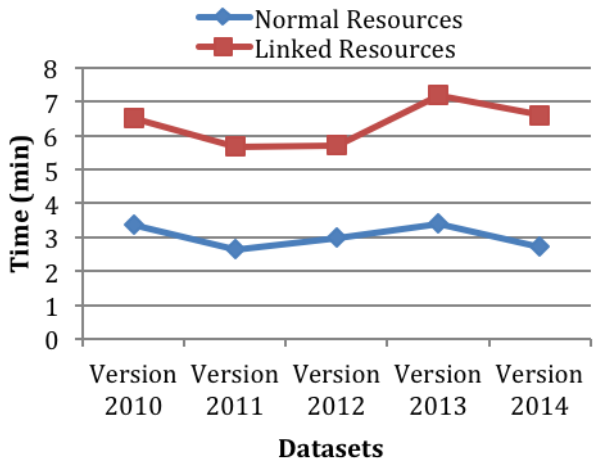


Figure 4. Comparing data extraction time between normal resources and linked resources

Though, loading whole CSV data into memory caused out of memory exception, as every files contained data more than few megabytes. To handle this situation, we stored the RDF models corresponding to each CSV files into disk and later loaded from the disk on demand. Figure 4 shows a comparison between the resources generated with links and without links. For linking resources, the total execution time is higher than the normal conversion process. The ontology generated by this approach depicts all class names along with their properties. User can edit the ontology in the editor window and can enter comment for each and every property. Using this ontology we have generated around 1260K RDF resources and around 14050K RDF triples. Jena Semantic Web framework [30] has been used for developing ontology. Ontologies are tested using Protégé, an Ontology Editor and Knowledge Based Framework [31]. The experiment has been performed on a 64-bit, 2 GHz Intel Core i7 processor having 4GB of RAM running on Mac OS X 10.10.2.

## 8. LIMITATIONS AND IMPROVEMENTS

Datasets resulted in rich set of RDF data. However, it needs further optimisation in storing and processing large size of tabular data. At the moment, the data from each file has been stored in memory, which causes slow performance while loading very large size of CSV files. Further, to generate links among resources, the source CSV file should contain exactly the same header name as in the target file. Otherwise, it cannot detect object properties. This part can be improved by analysing the kind of data on both the columns of source and target files. However, we plan to include this in future work. We can improve this work by programmatically looking for pre-existing ontology for the CSV files. Since, it is always recommended to reuse existing ontology to avoid heterogeneousness concepts for the same domain. There are approaches for finding relevant ontologies according to one's need. One appropriate solution for this is to use registries and libraries [32] to discover domain specific ontologies. Another, straightforward solution for finding ontologies is by using Google search engine by providing "<domain name> filetype:owl" text into the search bar. For example, we can find library related ontologies using the text "library filetype:owl".

## 9. CONCLUSION AND FUTURE WORKS

Data plays an important role in our day-to-day life. In the web, everything is dominated by data. However, in the present day, data suffers from machine understandable and analysable. Many attempts have been made to represent data into structured format and resolve their issues. But, many times the semantic relations among data are not considered. Data is not stored into properly structured format. Semantic Web brought amazing technologies for making data understandable by both machine and human. The important part is the semantic of data, which is highly exposed. Not only semantic, it also exhibits the relations among different concepts. In this article, we have extracted the CSV data into RDF using domain ontology for a set of CSV files. The useful resources are extracted and every member of CSV file is converted into members of domain ontology. This way, the data extraction job made easy, as the domain ontology provides legitimate directions on converting data. We believe such approach will help in converting data from related sources. In near future, we have planned to include efficiently storing tabular data of large size while converting, and reducing the time in linking phases.

## REFERENCES


[1] Han L, Finin T, Parr C, Sachs J and Joshi A, (2006) "RDF123: a mechanism to transform spreadsheets to RDF", Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI), AAAI Press, Menlo Park.

[2] Ermilov I, Auer S and Stadler C, (2013) "Csv2rdf: User-driven csv to rdf mass conversion framework", ISEM 13: 04-06.

[3] Bizer C, Heath T, and Berners-Lee T, (2009) "Linked data-the story so far", International journal on semantic web and information systems 5.3: 1-22.

[4] Taye MM, (2010) "Understanding semantic web and ontologies: Theory and applications", Journal of Computing 2 (6): 182-192.

[5] Euzenat J, Le Bach T, Barrasa J, Bouquet P, De Bo J, Dieng R, Ehrig M, Hauswirth M, Jarrar M, Lara R, Maynard D, Napoli A, Stamou G, Stuckenschmidt H, Shvaiko P, Tessaris S, Van Acker S, and Zaihrayeu I, (2004) "D2. 2.3: State of the art on ontology alignment", Knowledge Web 2-3.

[6] Uschold M and Gruninger M, (1996) "Ontologies: Principles, methods and applications", The knowledge engineering review 11.02: 93-136.

[7] Lin J, Fox MS, and Bilgic T, (1996) "A requirement ontology for engineering design", Concurrent Engineering 4.3: 279-291.

[8] Lee JH and Suh HW, (2008) "Ontology-based multi-layered knowledge framework for product lifecycle management", Concurrent Engineering 16.4: 301-311.

[9] Dutra M, Ghodous P, Kuhn O and Tri NM, (2010) "A generic and synchronous ontology-based architecture for collaborative design", Concurrent Engineering 18 (1): 65-74.

[10] Happel HJ, and Seedorf S, (2006) "Applications of ontologies in software engineering", Proc. of Workshop on Sematic Web Enabled Software Engineering : 5-9.

[11] Anantharangachar R, Ramani S, and Rajagopalan S, (2013) "Ontology Guided Information Extraction from Unstructured Text", International Journal of Web & Semantic Technology (IJWesT) Vol.4, No.1, January 2013.

[12] Lebo T and Williams GT, (2010) "Converting governmental datasets into linked data", Proceedings of the 6th International Conference on Semantic Systems: 38.

[13] Langegger A and Wöß W, (2009) "XLWrap–querying and integrating arbitrary spreadsheets with SPARQL", In: Springer Berlin Heidelberg. 8th International Semantic Web Conference. Chantilly, VA, USA, p. 359-374.

[14] Mulwad V, Finin T and Joshi A, (2011) "Automatically generating government linked data from tables", Working notes of AAAI Fall Symposium on Open Government Knowledge: AI Opportunities and Challenges 4 (3).

[15] Mulwad V, Finin T, Syed Z and Joshi A, (2010) "Using Linked Data to Interpret Tables", COLD 665.

[16] Mulwad V, Finin T, Syed Z, and Joshi A, (2010) "T2LD: Interpreting and Representing Tables as Linked Data", In 9th International Semantic Web Conference ISWC : 25-28.

[17] Spanos DE, Stavrou P and Mitrou N, (2012) "Bringing relational databases into the semantic web: A survey", Semantic Web 3 (2): 169-209.

[18] Lin L, Xu Z and Ding Y, (2013) "OWL Ontology Extraction from Relational Databases via Database Reverse Engineering", Journal of Software 8 (11): 2749-2760.

[19] Telnarova Z, (2010) "Relational database as a source of ontology creation", Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on IEEE: 135-139.

[20] Dadjoo M and Kheirkhah E, (2015) "An Approach For Transforming of Relational Databases to OWL Ontology", International Journal of Web & Semantic Technology (IJWesT) Vol.6, No.1, January 2015.

[21] Dimou A, Sande MV, Colpaert P, Verborgh R, Mannens E, and Walle RVd, (2014) "RML: a generic language for integrated RDF mappings of heterogeneous data", In Proceedings of the 7th Workshop on Linked Data on the Web.

[22] Muñoz E, Hogan A and Mileo A, (2014) "Using linked data to mine RDF from wikipedia's tables", In Proceedings of the 7th ACM international conference on Web search and data mining: 533-542.

[23] Petrou I, Meimaris M and Papastefanatos G, (2014) "Towards a methodology for publishing Linked Open Statistical Data", eJournal of eDemocracy & Open Government 6 (1).

[24] Sharma K, Marjit U and Biswas U, (2014) "Linking Library Data: A Linked Data Based Approach", PLANNER – 2014, Capacity Building in Library and Information Services, Dibrugarh University, Assam (39).

[25] Lange C, (2009) "Krextor–an extensible XML→ RDF extraction framework", Scripting and Development for the Semantic Web (SFSW) 449: 38.

[26] Butler MH, Gilbert J, Seaborne A and Smathers K, (2004) "Data conversion, extraction and record linkage using XML and RDF tools in Project SIMILE", HP Labs, Bristol, UK.

[27] Battle S, (2006) "Gloze: XML to RDF and back again", In Jena User Conference, May.: http://jena. hpl. hp. com/juc2006/proceedings.

[28] Sharma K, Marjit U and Biswas U, (2013) "Exposing MARC 21 Format for Bibliographic Data As Linked Data With Provenance", Journal of Library Metadata 13 (2-3): 212-229.

[29] Lahman S, (2014) "Lahman's Baseball Database", In Baseball Archive: Dataset versions 2010-2014. [Cited 2015 July 20]. Available from: http://seanlahman.com/.

[30] McBride B, (2001) "Jena: Implementing the RDF Model and Syntax Specification", In SemWeb.

[31] Knublauch H, Fergerson RW, Noy NF and Musen MA, (2004) "The Protégé OWL plugin: An open development environment for semantic web applications", The Semantic Web–ISWC 2004, Springer Berlin Heidelberg: 229-243.

[32] Alexander P, (2011) "Finding Ontologies", In The MMI Guides: Navigating the World of Marine Metadata. [Cited 2015 July 20]. Available from: http://marinemetadata.org/guides/vocabs/ont/existing/finding.