


OPPCAT: Ontology population from tabular data

Journal of Information Science
1–15
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0165551519827892
journals.sagepub.com/home/jis


Ovunc Ozturk 

Department of Computer Engineering, Manisa Celal Bayar University, Manisa, Turkey

Abstract

In order to present large amount of information on the Web to both users and machines, it is urgently needed to structure Web data. E-commerce is one of the areas where increasing data bottlenecks on the Web inhibit data access. Ontological display of the product information enables better product comparison and search applications using the semantics of the product specifications and their corresponding values. In this article, we present a framework called OPPCAT, which is used for semi-automatic ontology population from tabular data in e-commerce stores and product catalogues. As a result, OPPCAT allows tabular data to be used for mass production of ontology content. First, we present the common patterns in tabular data which obstruct semi-automatic production of ontologies. Then, we suggest solutions which automatically fix these errors. Finally, we define an algorithm to build ontology content semi-automatically.

Keywords

e-commerce; ontology; ontology population; Semantic Web; tabular data

1. Introduction

The use of ontologies has become extremely popular for representing machine readable semantic knowledge. However, building ontological content from scratch is a resource-demanding, time-consuming and error-prone task. Therefore, the automatic or semi-automatic construction of ontological content is an emerging research area. Ontology construction, enrichment and adaptation are known as ontology learning [1]. For example, Kutiyawala et al. [2] uses ontology learning for creating a product ontology designed specifically for search and providing three methods to automatically extract product concepts for this ontology. The ontology population, which falls under the heading of ontology learning, is concerned with the task of adding new concept and relation instances to the ontology [3]. Ontology population requires an initial ontology that will be populated.

This article presents a framework, namely, OPPCAT, for extracting data out of tables in PDF product catalogues and e-commerce stores with the aim of building ontology content. We aim at building ontology content in a fast and effective way even by users who are unfamiliar with ontologies. Our work mainly deals with ontology population. It requires an initial ontology to be populated. However, it extends the ontology through the addition of new classes and properties. Therefore, we also categorise this work as ontology enrichment. Although this work is related to e-commerce domain, the proposed approach gives generic solutions for creating ontologies from Web data. Our approach has two main characteristic features. First, it is specifically targeted at populating the ontology with tabular data. Second, it especially focuses on the anomalies in tabular data that prevent building reliable ontological content. The anomalies that frequently encountered in tabular data are identified and automatically solved by the proposed algorithms.

Section 2 presents related work in this field; then, section 3 presents the most common anomalies in spreadsheet files and defines algorithms to solve these anomalies. Section 4 describes building ontology content semi-automatically using the normalised spreadsheet and uses OPPCAT in a real ontology building scenario. Our approach has been used to create ontological content about the automobiles by inexperienced users. Then, these users filled two surveys about the

Corresponding author:

Övünç Öztürk, Department of Computer Engineering, Manisa Celal Bayar University, 45140, Yunusemre, Manisa, Turkey.
Email: ovunc.ozturk@cbu.edu.tr

OPPCAT framework. Section 5 assesses the performance of OPPCAT framework based on the results of these surveys. Finally, section 6 concludes the article with a brief talk about possible future work.

2. Related work

There are some similar research works for creating ontologies from Web data. OntoGenie [4] parses the web pages to create knowledge instances for a given ontology using WordNet as a bridge for mapping between the ontologies and the web page terms. OntoSophie [5] is a system for semi-automatic population of ontologies with instances from unstructured text. It is based on supervised learning. This system learns extraction rules from annotated text and then applies those rules on new articles to populate the ontology. OntoSyphon [6] is a fully automated, unsupervised system that takes an ontology as input and uses the ontology to specify Web searches that identify possible semantic instances, relations and taxonomic information. The advantage of this approach is that the entire Web can be used as a corpus for instantiating entities in the ontology. M^2 [7] is a mapping language for converting data contained in arbitrary spreadsheets into Web Ontology Language (OWL) [8]. The language is implemented in MappingMaster, which is available as a plug-in for the Protégé ontology editor [9]. The disadvantage of the approach is that it is integrated into Protégé and requires users to be familiar with ontologies.

Our approach differs from aforementioned works in literature because it is specifically targeted at populating the ontology with tabular data. From this point, our work is most similar to the literature [10–14]. Holzinger et al. [10] extracts tabular data on the web pages and uses content spotters to derive meaning of the table, that is, to interpret the table structure in terms of product features represented as attribute name–value pairs. Nederstigt et al. [11] detailed about a framework capable of semi-automatic ontology population of tabular product information on the Web. It extracts the raw data from Web pages and then it maps the raw data to predefined OntoProduct ontology concepts. OntoProduct is fully compatible with GoodRelations [15], which is known as ‘the web vocabulary for e-commerce’. Ozacar [12] introduces a tool that produces structured interoperable data from product features, that is, attribute name–value pairs, on the Web. The tool extracts the product features using a website-specific template created by the user. The value of the extracted data is maximised using GoodRelations. The final output of the tool is GoodRelations snippets, which contain product features encoded in RDFa or Microdata.

The data.world project [13] aims to create a semantic-based publication platform for data sets, scalable to hundreds of thousands of heterogeneous users and millions of distinct data sets. When a user creates a data set, data.world generates a unique graph database instance for it. data.world handles each data set individually, giving each its own SPARQL endpoint. When users upload structured data in tabular formats that data.world supports, the data.world ingest pipeline automatically converts those data files into Resource Description Framework (RDF) using the CSVW specification, which provides a standardised model for virtualised tables modelled within an RDF graph structure.

Reasonable Ontology Templates (OTTRs) [14] are simple, but powerful, templates or macros for ontologies, represented in using a dedicated OWL vocabulary. Specifically, the implicit mapping between an OTTR’s parameters and its pattern may be exploited to generate various format descriptions and transformation specifications, for example, queries for extracting pattern instances and transformations between tabular input formats and OTTR pattern instances that may be processed by readily available desktop tools.

Unlike these approaches, our methodology especially focuses on the anomalies in tabular data that prevent building reliable ontological content. Based on our experience, we define and exemplify the anomalies we most frequently encounter in tabular data. Then, we define algorithms to automatically solve these anomalies. Finally, we build the ontological content semi-automatically using the normalised spreadsheet file. It is also good to note that the normalised spreadsheet files can be used with other approaches like Holzinger et al. [10] and Nederstigt et al. [11] for various needs of users.

3. Solving the anomalies in tabular data

This work involves building ontology content from the tabular data in e-commerce stores or product catalogues. We assume tabular data is represented in spreadsheets. Most of the product catalogues is in PDF format, therefore these catalogues should be converted to spreadsheets using programs such as Tabula [16]. Figure 1 shows parts of the technical specification table in Audi A1 brochure. Each column (except the first one) in the table corresponds to an individual of the ‘Automobile’ class. Each value in the first column corresponds to a property of the ‘Automobile’ class. Each value in column i and row j correspond to the value of property j of individual i (i and j are greater than 0). In some cases, it can be necessary to transpose the data in the table.

	1.0 TFSI	1.0 TFSI	1.4 TFSI*	1.6 TDI
Transmission	5-speed manual	7-speed S tronic	7-speed S tronic	5-speed manual
Cylinder	3-cylinder	3-cylinder	4-cylinder	4-cylinder
Displacement, cubic cm	999	999	1395	1598
Max. power ¹ , PS at rpm	95/5000-5500	95/5000-5500	150/5000	116/3500-400
Max. torque ¹ , Nm at rpm	160/1500-3500	160/1500-3500	250/1500-3500	250/1500-300

Figure 1. Audi A1 technical specifications table (partial).

	1.0 TFSI	1.0 TFSI	ENGINE DATA	SDV6	SDV6
Transmission	5-speed manual	7-speed S tronic	Power (PS)	249	275
Cylinder	3-cylinder	3-cylinder	Bore (mm)	84	84
Displacement, cubic cm	999	999	Stroke (mm)	90	90
Max. Power, PS at rpm	95/5000-5500	95/5000-5500	Compression ratio (:1)	16	16

(a) (b)

Figure 2. Different individuals with identical names: (a) Different individual with identical names example from Audi brochure (b) Different individual with identical names example from Range Rover brochure.

The following sections involve normalising the spreadsheet tables before transforming it to an ontology. We identify the patterns that produce errors while populating the ontology from spreadsheets automatically. We list these patterns and give an example of each pattern. Then, we list potential solutions that automatically solve the problem in the spreadsheet file.

3.1. Renaming different individuals with identical names

3.1.1. Definition. If we have different individuals with identical names in the spreadsheet file, these individuals should be renamed properly.

3.1.2. Example. In Figure 2(a), two instances, named as ‘1.0 TFSI (Turbo Fuel Stratified Injection)’ have different ‘Transmission’ values. Two examples are separated due to the difference of the attribute values. Another example is from Range Rover brochure (Figure 2(b)). Two different instances, named as ‘SDV6’ have different ‘engine power’ values.

3.1.3. Solution. Two individuals should be renamed properly. Appending the different property (‘Transmission’/‘engine power’) values to the end of the individual names using Algorithm 1 is one of the possible renaming strategies. After executing the algorithm, the first individual in Figure 2(a) is renamed as ‘1.0 TFSI 5-speed manual’, while the second one becomes ‘1.0 TFSI 7-speed S tronic’. In the same manner, the first individual in Figure 2(b) is renamed as ‘SDV6 249’, while the second one becomes ‘SDV6 275’. Time complexity of the algorithm is $O(n^3)$.

Algorithm 1. Renaming different individuals with identical names.

```

1: procedure RENAMESAMENAMEDINDIVIDUALS () ▷ Get the individual names in the first row, if there are individuals i, j
   having same name in this row ▷ Get the attribute values of i and j ▷ Append the name of the first attribute, which has
   different values for i and j, to the name of j
2:   firstRow ← getRow(0)
3:   for i ← 0, firstRow.length - 1 do
4:     for j ← i + 1, firstRow.length do
5:       if getCell(0, i).value = getCell(0, j).value then
6:         if getCell(0, i).value ≠ 0 then
7:           c1 ← getColumn(i)
8:           c2 ← getColumn(j)
9:           for k ← 1, c1.length do
10:            if getCell(k, i).value ≠ getCell(k, j).value then
11:              oldName ← getCell(0, j).value
12:              newName ← oldName + getCell(k, j).value
13:              newName ← oldName + getCell(k, i).value
14:              getCell(0, j).updateValue(newName)
15:              getCell(0, i).updateValue(newName)

```

1.2 69hp Hatchback - Convertible		A6 2.0 TFSI (132 kW)	
ENGINE		Model	A6 2.0 TFSI (132 kW)
No. of cylinders, arrangement	4, in line, transverse front	Engine type	4-cylinder in-line petrol engine with turbocharging and intercooling
Bore × stroke (mm)	70.8 × 78.9	Displacement in cc (valves per cylinder)	1984 (4)
Displacement (cm ³)	1242	Max output³ in kW at rpm	132/4000 - 6000
TRANSMISSION		Max. torque in Nm at rpm	320/1500 - 3900
Drive system	front wheel drive	Weights/capacities	Saloon Avant
Clutch control	hydraulic	Unladen weight ² in kg	1615 [1640] 1680 [1705]
Gearbox, no. of gears	5 forward + reverse Dualogic™ robotised 5 + R	Gross vehicle weight in kg	2120 [2145] 2235 [2260]
FUEL CONSUMPTION		Roof load/trailer nose weight in kg	100/85 [100/85] 100/85 [100/85]
MPG (L/100 KM)		Trailer load limit ³ in kg	750 [750] 750 [750]
urban cycle	49.6 (5.7) 50.4 (5.6)*		
extra-urban cycle	65.7 (4.3) 68.9 (4.1)*		
combined cycle	58.9 (4.8) 60.1 (4.7)*		

Figure 3. Different individuals located in the same column: (a) Example for different individuals located in the same column from Fiat brochure (b) Example for different individuals located in the same column from Audi brochure.

3.2. Separating different individuals located in the same column

3.2.1. Definition. In this case, we have two or more different individuals represented in one column as if they were the same. In fact, this column corresponds to more than one individual. This situation can be identified by columns having some cells that are split into n columns.

3.2.2. Example. Figure 3(a) partially shows the technical specification table in Fiat 500 brochure as an example. ‘1.2 69hp Hatchback-Convertible’ value represents two individuals, which are separated due to the difference of ‘Gearbox’ attribute values. Figure 3(b) shows another example from Audi A6 brochure. ‘A6 TFSI (132 kW)’ value represents two different individuals, which are separated due to their body types (‘saloon’ and ‘avant’).

3.2.3. Solution. The second column in Figure 3(a) should be separated into two and then the two individuals should be renamed as ‘1.2 69hp Hatchback-Convertible 5 forward reverse’ and ‘1.2 69hp Hatchback-Convertible dualogic robotised 5 + R’. In the same manner, the individuals in second column in Figure 3(b) should be renamed as ‘A6 TFSI (132 kW) Saloon’ and ‘A6 TFSI (132 kW) Avant’. This step is completed automatically using Algorithm 2, whose time complexity is $O(n^3)$.

Algorithm 2. Separating different individuals located in the same column.

```

1: procedure SEPARATEINDIVIDUALS () ▷ Remove all the merged regions. After unmerging, the original value in the region will be
   kept in the first cell and other cells will have an empty content ▷ Duplicate the value in the first cell to fill all of the blank
   cells with the original value ▷ To rename the two different individuals with identical names, Execute Algorithm 1
2:   for  $r \leftarrow 0$ , sheet.getNumMergedRegions do
3:     sheet.removeMergedRegion( $r$ )
4:     firstRow  $\leftarrow$  getRow(0)
5:     for  $j \leftarrow 1$ , firstRow.length do
6:       if getCell(0,  $j$ ).value = NULL then
7:          $c1 \leftarrow$  getColumn( $j$ )
8:         for  $k \leftarrow 1$ ,  $c1.length$  do
9:           if getCell( $k$ ,  $j$ ).value = NULL then
10:             $val \leftarrow$  getCell( $k$ ,  $j - 1$ ).value
11:            getCell( $k$ ,  $j$ ).updateValue( $val$ )
12:   RENAMESAMENAMEDINDIVIDUALS()

```

	Pop	Colour Therapy
Safety		
ABS + EBD	■	■
Driver and passenger airbag with Dual stage system	■	■
Side bags	■	■
Audio		
Radio/CD/MP3 player (6 speakers)	■	■
Hi-Fi	○	○

(a)

MAXIMUMOUTPUT	
DINhp / kW	445 / 327
ENGINE	
Capacity (cm ³)	4969
Cylinders / valves	V8 / 32
Valve mechanism	Dual VVT-E
Fuel type	Petrol, 95 octane or higher
Max. power (DIN hp / kW @rpm)	394 / 290 @ 6400
ELECTRIC MOTOR	
Type	Synchronous, permanent magnet

(b)

Figure 4. (a) Property hierarchy in Fiat 500 options table (partial). (b) Property hierarchy in Lexus LS technical specifications table (partial).

3.3. Defining top properties

3.3.1. Definition. If there is a hierarchy between properties in spreadsheet file, it is required to represent this hierarchy in the knowledge base. In our experience, we observed that all the values of the top properties in the spreadsheet files are usually empty.

3.3.2. Example. Figure 4(a) illustrates an example hierarchy. All the values of the top properties (*Safety*, *Audio*) in the spreadsheet are empty. Another example (Figure 4(b)) is taken from Lexus LS brochure. In this example, the top properties are maximum output, engine and electric motor.

3.3.3. Solution. If all of the values (except the first one) in a row are empty, then the property (the first value of the row) is a top property. If i_1 is the row index of a top property p and i_2 is the row index of the next top property, then all properties having index values between $[i_1 + 1, i_2 - 1]$ should be defined as subproperties of property p . Array L , which is the output of Algorithm 3, is used for defining top properties automatically in section 4. Time complexity of the algorithm is $O(n)$.

3.4. Renaming different properties with identical names

3.4.1. Definition. Two different subproperties with different top properties may have identical names. In this case, we should rename these subproperties properly.

3.4.2. Example. Figure 5(a) shows an example of this situation from 'Audi A3' options table. The first 'Milano leather, black' property qualifies 'standard seats', while the second 'Milano leather, black' property qualifies the 'sport seats'.

Algorithm 3. Defining top properties.

```

1: procedure DEFINETOPPROPERTIES () ▷ Check each row (except the first one) ▷ If all of the values (except the first one) in a
   row is empty then define the property in the row as a top property ▷ Output: Array L of row indexes of all top properties
2:   numberOf Rows ← getColumn(0).length
3:   for i ← 1, numberOf Rows do
4:     if ISEMPYROW (i) then
5:       L[m] ← i
6:       i ← i + 1
7: function ISEMPYROW (l)
8:   numberOf Columns ← getRow(0).length
9:   for j ← 1, numberOf Columns do
10:    if getCell(i, j).value ≠ N U L L then
11:      return false
12:    return true

```

	A3/A3 Spo Attraction	A3/A3 Spo Ambition	A3/A3 Spo Ambiente	S3/S3 Spo	
					CO₂ EMISSIONS* (g/km)
					Emissions level Euro V+
					Combined 249
Seat upholstery for standard seats, front					FUEL CONSUMPTION* mpg(ℓ/100km)
Milano leather, black	■	—	■	—	Combined 26.4 (10.7)
Milano leather, pashmina beige	■	—	■	—	BRAKES
					Front Ventilated discs (Brembo)**
Seat upholstery for sport seats, front					Rear Ventilated discs (Brembo)**
Rallye cloth, black/blue	—	□	—	—	SUSPENSION
Milano leather, black	■	■	■	—	LS460
					Front Multi link air suspension
					Rear Multi link air suspension

(a)
(b)

Figure 5. Different properties with identical names: (a) Example for different properties with identical names from Audi brochure (b) Example for different properties with identical names from Lexus brochure.

Figure 5(b) shows another example from Lexus LS brochure. The first ‘Combined’ property qualifies the ‘CO₂ Emissions’, while the second one qualifies the ‘Fuel Consumption’, and the first ‘Front’/‘Rear’ property qualifies the ‘Brakes’, while the second one qualifies the ‘Suspension’.

3.4.3. Solution. In this case, we append the name of top property to the name of its subproperty using Algorithm 4. Time complexity of the algorithm is $O(n^2)$.

3.5. Separating different properties located in the same row

3.5.1. Definition. An attribute value can be represented as two values in different measurement units within the same cell. The secondary value is usually represented in parentheses.

3.5.2. Example. In Figure 6(a), ‘Maximum speed mph (kph)’ is a property represented by two different measurement units. In the same manner, ‘Acceleration’, ‘Maximum speed’, ‘Urban’ and ‘Extra Urban’ properties in Figure 6(b) are also represented by two different measurement units.

3.5.3. Solution. In this case, the property values should be defined in separate rows as ‘Maximum speed mph’ and ‘Maximum speed kph’ (Algorithm 5). In this solution, the secondary unit name must be represented in parentheses (e.g. ‘kph’). The ‘Units’ array contains all the values in the UN/CEFACT Common Codes List [17]. If the property name contains more than one unit name from the ‘Units’ array (e.g. ‘mph’ and ‘kph’) and the property values contain parentheses (e.g. ‘94(151)’), then we add a new row below. We store the primary property values in the original row and the

Algorithm 4. Renaming different properties with identical names.

```

1: procedure RENAMESAMENAMEDPROPERTIES () ▷ Check if column 0 contains recurring property names ▷ If so, then find
   top properties of each subproperty ▷ Append the names of top properties to the names of corresponding subproperties ▷
   Use function ISEMPYROW(i) in Algorithm 3 to find the top property
2:   numberOfRows ← getColumn(0).length
3:   for i ← 1, numberOfRows - 1 do
4:     for j ← i + 1, numberOfRows do
5:       p1 ← getCell(i, 0), p2 ← getCell(j, 0)
6:       if p1 = p2 then
7:         topi ← FINDTOPPROPERTY (i)
8:         topj ← FINDTOPPROPERTY (j)
9:         if topi ≠ -1 and topj ≠ -1 then
10:          getCell(i, 0).updateValue(p1 + getCell(topi, 0).value)
11:          (j, 0).updateValue(p2 + getCell(topj, 0).value)
12: function FINDTOPPROPERTY (i)
13:   while i ≥ 0 do
14:     if ISEMPYROW (i) then
15:       return i
16:     i ← i - 1
17:   return -1

```

PERFORMANCE		Performance	3.0L TDV6	PERFORMANCE	
Aerodynamic 5/Cx	0.715	Maximum speed km/h (mph)	209 (130)	Aerodynamic 5/Cx	0.715
Maximum speed mph(kph)	94(151)	Fuel Economy		Maximum speed mph	94
0-62 mph (0-100 km/h)	14.5	Urban l/100km (mpg)	7.8 (36.2)	Maximum speed kph	151
		Extra Urban l/100 km (mpg)	6.4 (44.1)	0-62 mph (0-100 km/h)	14.5
(a)		(b)		(c)	

Figure 6. Different properties located in the same row: (a) Example for different properties located in the same row from Mazda, (b) Example for different properties located in the same row from Land Rover and (c) The normalized property hierarchy using Algorithm 5.

secondary values in the new row. Figure 6(c) represents the result after processing the table in Figure 6(a). Time complexity of the Algorithm 5 is $O(n^2)$.

3.6. Removing non-alphanumeric characters from the spreadsheet file

3.6.1. Definition. In order to comply with the naming conventions of ontology editors such as Protégé, non-alphanumeric characters should be removed from the spreadsheet file.

3.6.2. Example. Figure 7 shows two spreadsheet files which may cause ontology editor errors due to the non-alphanumeric characters (in red box) it contains.

3.6.3. Solution. The removal procedure (Algorithm 6) contains the following operations in order: (1) small capitalisation; (2) replacing inch ('), euro (?), percentage (%) and degree (°) symbols with text; (3) removing non-alphanumeric characters; (4) replacing consecutive underscore characters with a single underscore; (5) removing the underscore characters which are at the beginning or at the end of the name; and (6) appending an underscore character in front of the names which starts with a digit.

Time complexity of the algorithm is $O(n^3)$.

4. Building the ontology

This process contains semi-automatic steps that gets a spreadsheet file and produces an ontology file. The aim is the mass production of the ontology content even by inexperienced users in a fast and efficient way. Figure 8 shows the interface

Algorithm 5. Separating different properties located in the same row.

```

1: procedure SEPARATEPROPERTIES () ▷ If the property name at row i contains more than one unit name, then insert a new
row at i ▷ insertRow(i) inserts a new row at i and moves the original row to i + 1 ▷ Split the value in the original row at
i + 1 and fill row i with primary values and row i + 1 with secondary values ▷ Rename the properties
2:   numberOf Rows ← getColumn(0).length
3:   for i ← 1, numberOf Rows do
4:     if HASMULTIPLEVALUES(i) and getCell(i, 1).value.contains(' ') then
5:       sheet.insertRow(i)
6:       for j ← 0, numberOf Columns do
7:         value ← getCell(i + 1, j).value
8:         parts ← value.split(' ')
9:         getCell(i, j).updateValue(parts[0])
10:        if j = 0 then getCell(i + 1, j).updateValue(parts[0] + parts[1])
11:        else getCell(i + 1, j).updateValue(parts[1].remove(' '))
12: function HASMULTIPLEVALUES(l) ▷ the units are stored in the Units array
13:   value ← getCell(i, 0).value
14:   counter ← 0
15:   for j ← 0, numberOf Units do unit ← Units[j]
16:     if value.contains(unit) then counter ++;
17:     if counter = 2 then return true
18:   return false

```

		Angle of approach (unladen)	–	28°
Automatic climate control	140	Angle of departure (unladen)	–	28°
Dualogic™ robotised gearbox	407	Tilt angle (unladen)	35°	35°
Dualogic™ controls on steering wheel	5BH	Wading depth – mm	600	800
Bi-xenon headlights	230	Ramp breakover angle (4x4 only)	–	25°

(a)
(b)

Figure 7. Non-alphanumeric characters in the names of concepts and individuals: (a) Example for non-alphanumeric characters in the names of concepts and individuals from Fiat brochure and (b) Example for non-alphanumeric characters in the names of concepts and individuals from Ford brochure.

of the application. User fills the textbox with the name of the class to which the individuals are mapped. The nodes of the tree in this figure are the property names in the first column of the spreadsheet file.

User checks the object properties in the tree. Please note that the datatype properties are relations between instances of classes and RDF literals and XML (eXtensible Markup Language) Schema datatypes. However, object properties are relations between instances of two classes. All the subproperties of a property should have the same metaclass. In other words, if a top property is a datatype property, then all of its subproperties should be datatype properties. In the same manner, if a top property is an object-type property, then all of its subproperties should be object-type properties.

At the final stage all the classes, properties and individuals in the ontology are generated automatically by Algorithm 7 in Appendix 1. The value entered in the textbox is parameter class, the unchecked property names (in Figure 8) are the values of array *P datatype* and checked values are the values of array *P objecttype*. *topPropertyIndices* stores the indices of all top properties, which are defined by array *L* in Algorithm 3. Algorithm 7 executes the following steps in order:

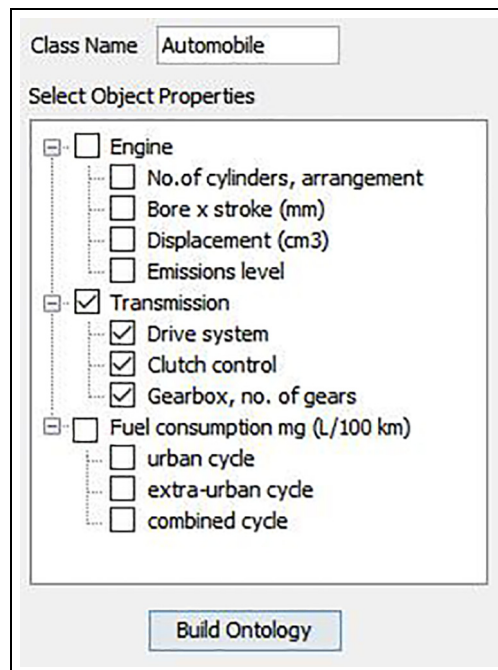
- Creates a class for the value entered by the user (e.g. ‘Automobile’ in Figure 8);
- Creates a new OWL datatype property for each datatype property (unchecked properties like ‘urban cycle’ in Figure 8);
- Creates a new OWL object property for each object-type property (e.g. ‘Transmission’, ‘Drive System’, ‘Clutch Control’ and ‘Gearbox, no. of gears’ in Figure 8);
- For each object-type property, if this property is not a top property, then it creates a new class for the range of the property;
- For each value in row 0, it creates an individual of the main class;

Algorithm 6. Removing non-alphanumeric characters.

```

1: procedure REMOVENONALPHANUMERICCHARACTERS ()
2:   numberOf Rows  $\leftarrow$  getColumn(0).length
3:   numberOf Columns  $\leftarrow$  getRow(0).length
4:   for  $i \leftarrow 0, \text{numberOf Rows}$  do
5:     for  $j \leftarrow 0, \text{numberOf Columns}$  do
6:       value  $\leftarrow$  getCell( $i, j$ ).value
7:       value  $\leftarrow$  value.toLowerCase()
8:       value  $\leftarrow$  value.replace(' ', 'inch')
9:       value  $\leftarrow$  value.replace('£', 'euro')
10:      value  $\leftarrow$  value.replace('%', 'percentage')
11:      value  $\leftarrow$  value.replace('°', 'degree')
12:      for  $m \leftarrow 0, \text{value.length}$  do
13:        if Character.isLetterOrDigit(value( $m$ )) = false then
14:          value  $\leftarrow$  value.replace(value( $m$ ), '')
15:      value  $\leftarrow$  value.replace(' ', '')
16:      if value.startsWith('ith(') then
17:        value  $\leftarrow$  value.substring(1, value.length)
18:      if value.endsWith('ith(') then
19:        value  $\leftarrow$  value.substring(0, value.length - 1)
20:      if Character.isDigit(value(0)) = true then
21:        value  $\leftarrow$  ' ' + value

```

**Figure 8.** Building the ontology file semi-automatically.

- For each property value v ($\text{cell}(m, n)$, where $m, n > 0$), if the corresponding property $p(\text{cell}(m, 0))$ is an object-type property, then it defines v as an individual of the class which represents the range of p and fill the property p value of i ($\text{cell}(0, n)$) with the value v ;
- For each property value v ($\text{cell}(m, n)$, where $m, n > 0$), if the corresponding property $p(\text{cell}(m, 0))$ is an datatype property, then it fills the property p value of i ($\text{cell}(0, n)$) with the value v ;
- Defines top properties for each value in $\text{topPropertyIndices}$.

Table 1. Project statistics.

No. of people	10
Project duration	1 month
Amount of work	331 man-hour
No. of brochures to be processed	267
No. of ontologies in the output	266 model ontology, 29 brand ontology, 1 domain ontology
No. of classes in the output	29
No. of properties in the output	30,171
No. of individuals in the output	1696 (automobiles), 818 (automobile options)
No. of triples in the output	191,340
Maximum depth of class hierarchy	3
Maximum depth of class hierarchy	2

The following procedures in Algorithm 7 simply produce OWL code corresponding triples below (the main class is represented by variable *class*);

- *createClass(c)* → *c* a owl: Class
- *createDatatypeProperty(p)* → *p* a owl: DatatypeProperty, *p* rdfs:domain class
- *createObjecttypeProperty(p, obj_c)* → *p* a owl: ObjectProperty, *p* rdfs:domain class, *p* rdfs:range obj_c
- *createIndividual(i, c)* → *i* a *c*
- *defineSubProperty(sub, top)* → *sub* rdfs: subPropertyOf *top*
- *assignDataPropertyValue(i, p, v)* → *i* *p* 'v'^xsd: string
- *assignObjectPropertyValue(i_sub, p, i_obj)* → *i_sub* *p* *i_obj*

OPPCAT is used in a real-life project, which involves creating ontological content about the automobiles. The tables in PDF catalogues of various automobile brands in Europe have been extracted using Tabula. Then, technical specifications and options of cars in spreadsheet files are imported into ontologies in accordance with OPPCAT methodology. In this study, the reasons for the choice of OPPCAT methodology are the challenging time constraint (1 month) and the lack of experienced project staff. Table 1 represents some statistics about the project (training hours was taken into account when calculating the ‘amount of work’).

5. Evaluation

We evaluate the usability of the OPPCAT method using two questionnaires: Computer System Usability Questionnaire (CSUQ) [18] and System Usability Scale (SUS) [19]. The reason for choosing these questionnaires is that these two approaches have a higher accuracy with an increasing sample size than the other questionnaires.

Figure 9 shows the results of the study by Tullis and Stetson [20]. They analyse the effectiveness of four standard usability questionnaires: SUS, QUIS (Questionnaire for User Interaction Satisfaction) [21], CSUQ and Words (adapted from Microsoft’s Product Reaction Cards [22]). As one would expect, the accuracy of the analysis increases as the sample size gets larger. SUS and CSUQ reach asymptotes of 90%–100% at a sample size of 12.

In this work, we have 10 participants; therefore, the percentage of the reliability of the results is about 75%. All of the participants were undergraduate students with no experience on building ontology or ontology-based systems. The usability test was conducted as follows; first, the participants received a 60-min introduction of applying the method. After this introduction, each participant applied the OPPCAT methodology using catalogues of the one (or two) automobile brand. At the end, each participant filled out a CSUQ and an SUS Questionnaire. The following sections present the results of the questionnaires and discussion of these results.

5.1. CSUQ evaluation results

The CSUQ contains 19 questions and users rate them from 1 to 7, where 1 is ‘strongly agree’ and 7 is ‘strongly disagree’. The three internal subscales of CSUQ (Table 2) are ‘System Usefulness’, ‘Information Quality’ and ‘Interface

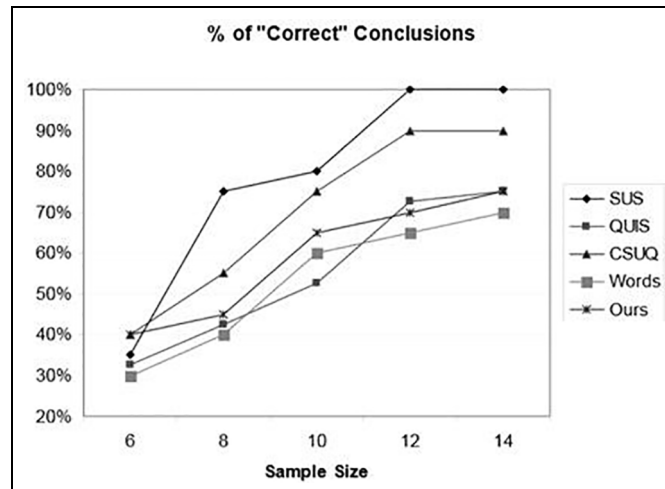


Figure 9. Comparison of the significance of the different usability questionnaires.

Quality'. The first eight questions on CSUQ assess 'System Usefulness'. These questions refer to the users' perception of the ease of use, learnability, speed of performance and effectiveness in completing tasks and subjective feeling. Questions 9–15 can be used as a means of assessing the participants' satisfaction with the quality of the information associated with the system. 'Information Quality' includes the users' beliefs regarding error messages and error handling, information clarity, understandability and utility more generally. Questions 16–18 provide a score for the 'Interface Quality'.

The results of the CSUQ are shown in Table 3, where N is the number of responses, AVG is the average value, DEV is the deviation value, MED is the median, MAX is the maximum and the MIN is the minimum. Most of the participants appreciated that the method was easy-to-apply and has a clear value and purpose. They found it to be a time-saving, fast and useful, which contributes to lowering the cost of data entry and improves productivity. They mentioned that this method provides clear and reliable results with standard PDF catalogues. Participants who had catalogues of two different brands felt comfortable about using the method for different brands without any extra knowledge. The majority of participants enjoyed to use the interface and found it clear and pleasant. In general, the participants were satisfied with the functions and capabilities of the system. The quality of the error messages and the documentation was ranked positive in average, but some participants criticised the inadequate and very generic error messages.

We further calculated the overall score and the three factor scores for 'System Usefulness', 'Information Quality' and 'Interface Quality' for all participants as illustrated in Table 4. The overall assessment of the participants about the usability of the method was positive (82.33%). The results show that the 'System Usefulness' ranked highest in all scores (83.93%). Finally, 'Information Quality' and 'Interface Quality' ranked about 80%.

5.2. SUS evaluation results

SUS is one of the popular questionnaires, used for the assessment of usability (Table 5). It is described as 'quick and dirty' usability scale. The participant rates each question with a 5-point scale, where 5 is 'strongly agree' and 1 is 'strongly disagree'.

The score of positive questions (1, 3, 5, 7 and 9) is the rating minus 1. The score of negative questions (2, 4, 6, 8 and 10) is 5 minus the rating. To obtain the overall score on a scale of 0–100, you add up the score values and multiply the sum by 2.5. The score values of the different participants are presented in Table 6. The final average SUS score is 81.75.

Table 7 provides the SUS scores with their corresponding adjective [23] and acceptability [24] ratings. The results show that the proposed method has a 'good' and 'acceptable' level of usability. Likewise, the results in Table 6 show that all participants rated usability above 62.5.

Table 2. The three internal subscales of CSUQ.

Score name	Average to responses to
Overall	Questions 1–19
System Usefulness	Questions 1–8
Information Quality	Questions 9–15
Interface Quality	Questions 16–18

CSUQ: Computer System Usability Questionnaire.

Table 3. CSUQ evaluation results.

Question	N	AVG	DEV	MED	MAX	MIN
Overall, I am satisfied with how easy it is to use this method	10	5.7	0.67	6.0	7	5
It was simple to use this method	10	5.8	1.03	6.0	7	4
I can effectively complete my work using this method	10	6.0	0.82	6.0	7	5
I am able to complete my work quickly using this method	10	5.4	1.17	5.5	7	4
I am able to efficiently complete my work using this method	10	5.8	0.63	6.0	7	5
I feel comfortable using this method	10	6.1	0.99	6.0	7	4
It was easy to learn to use this method	10	6.4	0.70	6.5	7	5
I believe I became productive quickly using this method	10	5.8	0.79	6.0	7	5
The method gives error messages that clearly tell me how to fix problems	10	5.2	1.23	5.5	7	3
Whenever I make a mistake using the method, I recover easily and quickly	10	4.9	1.10	5.0	6	3
The information (such as online help, on-screen messages and other documentation) provided with this method is clear	10	6.1	0.57	6.0	7	5
It is easy to find the information I needed	10	6.0	0.82	6.0	7	5
The information provided for the method is easy to understand	10	5.8	0.79	6.0	7	5
The information is effective in helping me complete the tasks and scenarios	10	5.9	0.74	6.0	7	4
The organisation of information is clear	10	5.6	1.26	6.0	7	3
The interfaces of the related tools are pleasant	10	5.8	0.92	6.0	7	4
I like using the interfaces of the related tools	10	5.5	0.97	6.0	7	4
This method has all the functions and capabilities I expect it to have	10	5.5	0.71	6.0	6	4
Overall, I am satisfied with this method	10	6.2	0.63	6.0	7	5

CSUQ: Computer System Usability Questionnaire.

Table 4. CSUQ single participant evaluation scores.

Respondent no.	Overall (%)	System Usefulness (%)	Information Quality (%)	Interface Quality (%)	Average (%)	Score value (%)
1	5.47	5.50	5.29	5.33	5.40	77.12
2	5.95	6.63	5.43	5.00	5.75	82.15
3	5.53	5.63	5.14	6.00	5.57	79.62
4	5.84	6.25	5.43	5.67	5.80	82.81
5	5.16	5.38	5.29	4.33	5.04	71.97
6	5.84	6.25	5.57	5.33	5.75	82.13
7	5.89	5.75	6.00	6.00	5.91	84.45
8	6.32	6.50	6.29	6.00	6.28	89.65
9	6.16	5.50	6.57	6.67	6.22	88.91
10	5.47	5.38	5.43	5.67	5.49	78.37
Average	5.76	5.88	5.64	5.60		
Score value	82.33	83.93	80.61	80.00		

Table 5. Items of the SUS questionnaire.

No.	Question
1	I think that I would like to use this method frequently
2	I found the method unnecessarily complex
3	I thought the method was easy to use
4	I think that I would need the support of a technical person to be able to use this method
5	I found the various functions in this method were well integrated
9	I thought there was too much inconsistency in this method
7	I would imagine that most people would learn to use this method very quickly
8	I found the method very cumbersome to use
9	I felt very confident using the method
10	I needed to learn a lot of things before I could get going with this method

SUS: System Usability Scale.

Table 6. SUS evaluation results.

Question no.	Responder 1	Responder 2	Responder 3	Responder 4	Responder 5	Responder 6	Responder 7	Responder 8	Responder 9	Responder 10
1	5	4	4	3	3	4	5	4	4	4
2	1	1	2	1	2	2	1	3	1	2
3	5	3	5	5	4	4	5	4	5	4
4	3	3	2	3	3	2	1	2	3	1
5	5	4	4	4	4	3	5	4	5	4
6	1	1	1	2	3	2	1	2	1	1
7	5	3	5	5	4	5	5	4	5	4
8	1	4	1	3	2	1	1	2	1	1
9	5	4	4	5	5	4	5	4	5	5
10	2	1	1	2	5	2	1	2	2	1
Sum	37	28	35	31	25	31	40	29	36	35
Score value	92.5	70	87.5	77.5	62.5	77.5	100	72.5	90	87.5

SUS: System Usability Scale.

Table 7. SUS scores with their corresponding adjective and acceptability ratings.

SUS scores	Adjective ratings	Acceptability ratings
89–100	Best imaginable	Acceptable
84–88	Excellent	
71–83	Good	Marginal Unacceptable
50–70	OK	
32–49	Poor	
20–31	Awful	
0–19	Worst imaginable	

SUS: System Usability Scale.

6. Conclusion and future work

In most of the cases, the tabular product data on the Web and in the product catalogues are detailed and reliable. This work is a pragmatic approach for building ontological content from the tabular data in these sources. It provides a methodology and tool to help the inexperienced users enrich an ontology. The process contains semi- or fully automatic steps. Although there are similar studies that aim at populating ontologies using tabular data, the main difference of the approach presented in this article is that it takes anomalies in tabular data into consideration to build more reliable ontological content. The proposed anomaly detection and correction system eliminates: (1) different individuals with

identical names, (2) different individuals located in the same column, (3) properties with empty values, (4) different properties with identical names, (5) different properties located in the same row and (6) non-alphanumeric characters in the tabular data.

A possible future work is to infer missing property values using data mining techniques on the existing property instances. We can also define further and more complex anomalies, and give algorithmic solutions for them. Another possibility is to build ontological content in a fully automatic fashion.

Appendix

Algorithm 7. Building the ontology content (time complexity is $O(n^2)$).

```

1: procedure BUILDONTOLOGY (class, P_datatype[0..n] P_objecttype[0..m] topPropertyIndices)
2:   class ← ToCamelCase(class)
3:   class[0] ← Character.upperCase(class[0])
4:   createClass(class)
5:   for i ← 1, n do
6:     P_datatype[i] ← ToCamelCase(P_datatype[i])
7:     createDatatypeProperty(P_datatype[i])
8:   for i ← 0, m do
9:     P_objecttype[i] ← ToCamelCase(P_objecttype[i])
10:    obj class ← P_objecttype[i]
11:    obj class[0] ← Character.upperCase(obj class[0])
12:    createClass(obj class)
13:    createObjecttypeProperty(P_objecttype[i] obj class)
14:  numberOf Rows ← getColumn(0).length
15:  numberOf Columns ← getRow(0).length
16:  for i ← 1, numberOf Columns do
17:    i name ← ToCamelCase(getCell(0, i).value)
18:    createIndividual(i name, class)
19:    for j ← 1, numberOf Rows do
20:      p ← ToCamelCase(getCell(j, 0).value)
21:      value ← getCell(j, i).value
22:      if p in P_objecttype[0..m] then
23:        obj class ← p name
24:        obj class[0] ← Character.upperCase(obj class[0])
25:        obj value ← ToCamelCase(value)
26:        createIndividual(obj value, obj class)
27:        assignObjectPropertyValue(i name, p name, obj value)
28:      else assignDataPropertyValue(i name, p name, value)
29:  for i ← 0, topPropertyIndices.length do
30:    currentIndex ← topPropertyIndices[i]
31:    nextIndex ← topPropertyIndices[i + 1]
32:    topProperty ← ToCamelCase(getCell(currentIndex, 0).value)
33:    for j ← currentIndex + 1, nextIndex do
34:      subProperty ← ToCamelCase(getCell(j, 0).value)
35:      defineSubProperty(subProperty, topProperty)
36:  function TOCAMELCASE(name)
37:    for j ← 0, name.length do
38:      if name[j] = 's' then
39:        name[j + 1] ← Character.upperCase(name[j + 1])
40:    name ← name.replace('s', '')
41:    return name

```


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship and/or publication of this article.

ORCID iD

Ovunc Ozturk  <https://orcid.org/0000-0001-7127-7902>

References

- [1] Maedche A and Staab S. Ontology learning. In: Staab S and Studer R. (eds) *Handbook on ontologies*. Berlin: Springer, 2004, pp. 173–190.
- [2] Kutiyawala A, Verma P and Yan Z. Towards a simplified ontology for better e-commerce search. In: *Proceedings of SIGIR workshop on e-commerce*, Ann Arbor, MI, 8–12 July 2018.
- [3] Petasis G, Karkaletsis V, Paliouras G et al. Ontology population and enrichment: state of the art. In: Paliouras G, Spyropoulos CD and Tsatsaronis G. (eds) *Knowledge-driven multimedia information extraction and ontology evolution*. Berlin; Heidelberg: Springer, 2011, vol. 6050, pp. 134–166.
- [4] Patel C, Supekar K and Lee Y. Ontogenie: extracting ontology instances from WWW. In: *Proceedings of human language technology for the semantic web and web services (ISWC)*, Sanibel Island, FL, 20–23 October 2003.
- [5] Celjuska D and Vargas-vera M. Ontosophic: a semi-automatic system for ontology population from text. In: *International conference on natural language processing (ICON)*, Hyderabad, India, 19–22 December 2004.
- [6] McDowell L and Cafarella M. Ontology-driven, unsupervised instance population. *Web Semant: Sci Serv Agents World Wide Web* 2008; 6(3): 218–236.
- [7] O'Connor MJ, Halaschek-Wiener C and Musen MA. Mapping master: a flexible approach for mapping spreadsheets to owl. In: *The semantic web (ISWC)*, Shanghai, China, 7–11 November 2010, pp. 194–208. Berlin; Heidelberg: Springer.
- [8] Smith MK, Welty C and McGuinness DL. Owl web ontology language guide, 2004, www.heppnetz.de/ontologies/vso/ns
- [9] Noy NF, Ferguson RW and Musen MA. The knowledge model of protege-2000: combining interoperability and flexibility. In: Dieng R and Corby O. (eds) *Knowledge engineering and knowledge management methods, models, and tools*. Berlin: Springer, 2000, vol. 1937, pp. 17–32.
- [10] Holzinger W, Krupl B and Herzog M. Using ontologies for extracting product features from web pages. In: *The semantic web (ISWC)*, Athens, GA, 5–9 November 2006, pp. 286–299. Berlin; Heidelberg: Springer.
- [11] Nederstigt LJ, Aanen SS, Vandic D et al. Floppies: a framework for large-scale ontology population of product information from tabular data in e-commerce stores. *Decis Support Syst* 2014; 59: 296–311.
- [12] Ozacar T. A tool for producing structured interoperable data from product features on the web. *Inform Syst* 2016; 56: 36–54.
- [13] Jacob B and Ortiz J. Data.world: a platform for global-scale semantic publishing. In: *Proceedings of the ISWC posters & demonstrations and industry tracks co-located with 16th international semantic web conference (ISWC)*, Vienna, 23–25 October 2017.
- [14] Skjaveland MG, Forssell H, Kluwer JW et al. Pattern-based ontology design and instantiation with reasonable ontology templates. In: *Proceedings of the 8th workshop on ontology design and patterns (WOP)*, Vienna, 21 October 2017.
- [15] Hepp M. GoodRelations: an ontology for describing products and services offers on the web. In: *Proceedings of the 16th international conference on knowledge engineering: practice and patterns (EKAW)*, Aci Trezza, 29 September–2 October, pp. 329–346. Berlin; Heidelberg: Springer.
- [16] Aristaran M, Tigas M and Merrill J. Introducing tabula, 2013, <https://source.opennews.org/en-US/articles/introducing-tabula/>
- [17] Group UICM. *United Nations Economic Commission for Europe, UN/CEFACT Common Codes for Units of Measurement*. http://wiki.goodrelations-vocabulary.org/Documentation/UN/CEFACT_Common_Codes, (2006, accessed 18 February 2019).
- [18] Lewis JR (1995). IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human – Computer Interaction* 7(1): 57–78.
- [19] Brooke J. SUS: a quick and dirty usability scale. In: Jordan PW, Thomas B, McClelland IL et al. (eds) *Usability evaluation in industry*. London: Taylor and Francis, 1996, pp. 189–194.
- [20] Tullis TS and Stetson JN. A comparison of questionnaires for assessing website usability. In: *Proceedings of Usability Professionals Association (UPA) conference*, Minneapolis, MN, 7–11 June 2004.
- [21] Harper BD and Norman KL. Improving user satisfaction: the questionnaire for user interaction satisfaction version 5.5. In: *Proceedings of the 1st annual Mid-Atlantic human factors conference*, Virginia Beach, VA, February 1993, pp. 224–228.
- [22] Benedek JMT. Measuring desirability: new methods for evaluating desirability in a usability lab setting. In: *Proceedings of Usability Professionals Association (UPA) conference*, London, 2–6 September 2002.
- [23] Miller JT, Bangor A and Kortum PT. Determining what individual SUS scores mean: adding an adjective rating scale. *J Usability Stud* 2009; 4(3): 114–123.
- [24] Bangor A, Kortum PT and Miller JT. An empirical evaluation of the System Usability Scale. *Int J Hum-Comput Int* 2008; 24(6): 574–594.