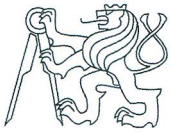
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF MASTER'S THESIS

Title:	Automatic ontology learning from semi-structured data
Student:	Bc. Filip Masri
Supervisor:	Ing. Jan Šedivý, CSc.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2017/18

Instructions

The purpose of this thesis is to automatically generate ontology from semi-structured data for e-commerce products. Review state of the art in the ontology learning. Compare the approaches and select a method for learning ontology from the Internet semi-structured content. Implement the chosen method and automatically generate ontology for mobile phones. Focus on the Internet e-commerce sites to acquire the content for learning. To evaluate the generated ontology use the "Gold Standard Evaluation of Ontology Learning Methods Through Ontology Transformation and Alignment" (http://ieeexplore.ieee.org/document/5601723). Use the "Mobile Phone Ontology" from the OPDM (http://www.ebusiness-unibw.org/ontologies/opdm/) as a gold standard.

References

Will be provided by the supervisor.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 9, 2017

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Master's thesis

Automatic ontology learning from semi-structured data

Bc. Filip Masri

Supervisor: Ing. Jan Šedivý, CSc.

4th May 2017

Acknowledgements

I would like to thank my thesis supervisor Ing. Jan Šedivý, CSc. for the continuous support, patience and motivation.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on 4th May 2017 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Masri, Filip. *Automatic ontology learning from semi-structured data.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Používání ontologií pro zachycení znalostí není žádnou novinkou. Důkazem tomu jsou veřejně dostupné ontologie, například z iniciativy Schema.org, které se hojně používají pro anotování webové obsahu. Ovšem, tyto ontologie bývají příliš obecné. Proto je potřeba systémů, které by generovaly ontologie zaměřené na specifičtější domény typu Mobilní telefony. Takové ontologie by poté mohly sloužit k rozšiřování obecnějších ontologií, jako je právě Schema.org.
Zde by mohla pomoci tato práce, která se zameřuje na vytváření ontologií z <table> elementů obsažených ve webových stránkách. Implementovaný systém využívá metody pro klasifikaci typu tabulky, detekci hlavičky, porozumění vztahům mezi buňkami v tabulce a vytváření finální ontologie v RDF/OWL formátu.
Výsledný přístup byl úspěšně aplikován na doménu mobilních telefonů. Jednotlivé ontologie byly vygenerovány z tabulek nalezených na stránkách amazon.com, buymobiles.net, gadgets.ndtv.com a snapdeal.com. Kromě této domény se daný systém dá využít i na další domény jako např. kamery, firmy, auta, basketbalový hráči.

Klíčová slova Ontologie, Znalostní databáze, Vytěžování vztahů z tabulek

Abstract

Publicly available ontologies, such as Schema.org, tend to be quite general. Therefore, demand for systems automatically generating domain specific ontologies has arose. The generated ontologies could later extend the general ones, for example in Schema.org.
This thesis focuses on building ontologies from <table> elements found in WEB pages. Methods were implemented for table type classification, header location, table understanding and creating final ontologies in RDF/OWL.
The implemented system has been successfully applied to mobile phones domain. Ontologies were generated from tables found on amazon.com, buymobiles.net, gadgets.ndtv.com and snapdeal.com. Moreover, the system is applicable to other domains, such as cameras, companies, cars and basketball players.

Keywords Ontologies, Knowledge databases, Table understanding

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Intelligent thinking machines represent the next challenging goal in the current information industry. These machines should be able to percept their surroundings and real world entities they keep interacting with. The relationship among these objects and their semantic context has to be presented to the machines in a suitable form, such as knowledge database, from which they can learn. The knowledge sources rely on creation of concepts describing different real-world entities and linking them together which is known as learning ontologies.

1.1 What is an ontology

A common understanding of an ontology has to be defined before diving deeper into the research of existing systems. The term ontology was specified by Thomas Gruber in 1993 [1] as "An ontology is a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents."
Therefore, the two building blocks of an ontology are concepts and relations. Concepts represent classes of entities and their individual members are called instances. Relations among concepts are called semantic relations. [2]
Moreover, ontologies can be created for different domains and serve as a foundation for a knowledge database which is populated with instances of given concepts. Such databases are used in information retrieval, question-answering systems and other applications. Examples of widely used semantic databases are:

- WordNet - lexical database of English - synonyms are grouped into sets of cognitive synonyms (synsets) [3]
- Freebase - large knowledge database acquired by Google [4]

- DBpedia - knowledge database extracting structured data from Wikipedia [5]

- YAGO - knowledge database extracted from multiple sources as WordNet, Wikipedia, etc. [6]

1.2 Insight into building ontology

Domain ontology building is a difficult task often performed by domain expert. Nevertheless, it is still a time consuming process which is the reason why automatic ontology learning has to be considered. The building process, both manual and automatic, has to take into consideration following characteristics, specified in [7]:

- Heterogenity - There are different data sources all over the World Wide Web (WEB) that potentially offer similar information. Integrating such data sources could lead to more accurate ontologies and could increase coverage of the ontologies. However, integrating ontologies is a difficult task consisting of subtasks as ontology merging, alignment and transformation. Result of an integration of two ontologies could be either relations linking concepts between them or a new global ontology.

- Uncertainty - Quality of learned ontologies heavily depends on the correctness of data it is derived from as well as it depends on the method used for ontology learning. The correctness of an ontology can be subjective and individual domain experts could argue about it.

- Reasoning - Deriving facts that are not expressed in ontology or in knowledge base explicitly is a challenging task that enriches existing databases or helps in better understanding of semantic context of given information. The knowledge databases should be build from consistent databases in order to reason in them.

- Scalability - Amount of available information keeps rapidly growing. Information can change, expand or appear in different formats. Therefore, ontologies that describe such information should be scalable and ready for change.

- Quality - measuring the quality of either manually or automatically created ontology is not an easy task. Quality of ontology depends on various factors such as its structure, hierarchy, used taxonomy, relations used among concepts, etc. Moreover, the quality also depends on the building approach which is domain specific. The resulting quality of the ontology has to be evaluated using a suitable method.

- Interactivity - Most of the ontologies are created manually by domain experts because its accuracy decreases with the increasing level of automation. Therefore, the automatic approaches require lots of pre-processing and post-processing steps.

1.3 Problem statement

The WEB is full of interesting information such as product specifications, personal information or company information. The published information tend to have some semantic meaning in the context of its domain. Moreover, the context can be captured by using annotations/tags. Annotating the content of a web page belongs to the field called Sematic Web [7] and is still in its infancy since not many ontologies were developed for the content annotation. In addition, the existing ontologies GoodRelations [8] and Schema.org [9] are too general and do not cover vast amount of domain specific information.
The lack of suitable ontologies leads to the demand of systems that would automatically generate domain specific ontologies that could be linked to the more general ones. These systems are very valuable and are a competitive advantage in the information industry. The application of such systems ranges from Semantic Web, knowledge graphs, search optimization systems, to question-answering, dialogue and recommendation systems.
Automatically generating ontologies is also called ontology learning and it is a non-trivial task. Obviously, learning can be performed on either unstructured, semi-structured or structured data. The first two data sources are suitable for ontology learning since large number of these sources are publicly available on the WEB.
Nevertheless, ontologies learned from unstructured text tend to be very inaccurate. Thus, tabular data were chosen since their content is semi-structured and the relation among its elements can be partially reconstructed.

1.4 Thesis outline

This section serves as a guidepost to the rest of the thesis. Chapters and their brief descriptions are following:

- Chapter 2 **Related work** - This chapter serves for familiarization with various systems and methods used in ontology learning that could be used in this thesis or could serve as an inspiration.
- Chapter 3 **New system design** - This chapter briefly describes problems the proposed system should deal with.
- Chapter 4 **WEB table representation** - This chapter is dedicated to internally representing the WEB table in the program memory.

- Chapter 5 **WEB table type classification** - This chapter mentions the existence of different WEB table types and means for their distinguishing.
- Chapter 6 **WEB table header recognition** - This chapter solves the problem of header location in WEB tables.
- Chapter 7 **Table understanding** - This chapter is dedicated to mining relations among table cells.
- Chapter 8 **Entity and Quantitative value recognition in table cells** - This chapter describes method for discovering important parts in the table content.
- Chapter 9 **Ontology generation** - This chapter gives insight into building the final ontology.
- Chapter 10 **Comparison created ontologies to Golden standard** - This chapter experiments with method for automatic ontology comparison to gold standard ontology.

Chapter 2

Related work

This thesis specializes on creating ontologies from semi-structured data. Domain ontology constructions are usually created by domain experts manually, which is tedious and time consuming. Therefore, automatic extraction and building ontology from existing information sources like WEB pages has been an emerging field of study and an urgent task [10].
Moreover, applications such as the Semantic Web or making web content directly queryable need an approach that would facilitate ontology creation. This simplification would lead to production of vast number and variety of ontologies required for future web applications [11].
Therefore, a research was done that would give an insight into the problematics. The structure of the research chapter is following. First, existing systems used for ontology learning were analyzed. However, some methods used by the systems are too complex or unclear since the systems are not open-source. Thus, other methods were explored for inspiration.
Finally, a framework called FLOPPY is mentioned. The framework uses a process called "conceptual extraction" for database population. The conceptual extraction takes a schema similar to an ontology and populates a knowledge database based on the ontology elements. Hence, adjusting the learned ontologies in this thesis to extraction concepts would lead to a knowledge database and is a great motivation.

2.1 Systems for automatic ontology learning from semi-structured data

Two systems for automatic ontology learning from semi-structured data were found among existing articles. Such systems are called MOGO and TARTAR. Both of them use <table> element for ontology learning. An ontology is reconstructed from the relations among individual cells in the table.

2.1.1 MOGO

First system is called MOGO (a Mini-Ontology GeneratOr) and is a component of the overall TANGO (Table ANalysis for Generating Ontologies) [12] project. TANGO focuses on concepts recognition in HTML (HyperText Markup Language) tables and links them to other related concepts. The concept recognition is done by MOGO component that transforms the table into a conceptual model, also called mini-ontology.
The component consists of three services. First is a service used for concept/-value recognition that evaluates a table column as a set of similar objects (using WordNet as lexical resource), detects hierarchy among header cells, detects similar value format in rows/columns, etc. The service uses quite complex operations as inserting category, deleting category, inserting subcategory, deleting subcategory, changing label, changing entry value and getting entry value that are iteratively applied to the table representation [13].
The second service discovers relationships among concepts by leveraging hyponyms and hypernyms. The last service discovers constraints by computing aggregations among value columns/rows, etc., and adds them to the final mini-ontology. Moreover, assignment of a name to the concept, derived from caption and surrounding elements, is also performed [11].
The evaluation was manually tested on 20 downloaded pages with tables and gave F-measure 90% in concept recognition, 77% in relationship discovery and 90% in constraint discovery.

2.1.2 TARTAR

Second framework is called TARTAR (Transforming ARbitrary TAbles into fRames) [14] and is a framework receiving HTML pages as input. Then, tabular data is retrieved and corresponding schema is created.
The process of creating schema is divided into several subtasks, depicted in figure 2.1. First, a table orientation is recognized by heuristic rules. Secondly, data cells are binded with header cells. Next, the table is transformed to a special structure, used in object-oriented programming, called F-Logic - Frame Logic [15] by creating Functional Table Model, showed in figure 2.2. The F-Logic is similar to the first service in MOGO that finds sets of similar objects by applying string similarity patterns. Finally, the F-Logic can be transformed to Resource Description Format (RDF) [16] / Web Ontology Language (OWL) [17] format.

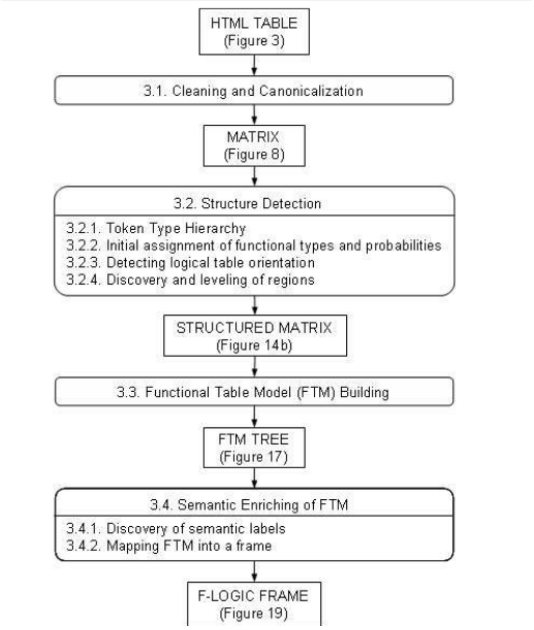


Figure 2.1: Workflow of TARTAR system.

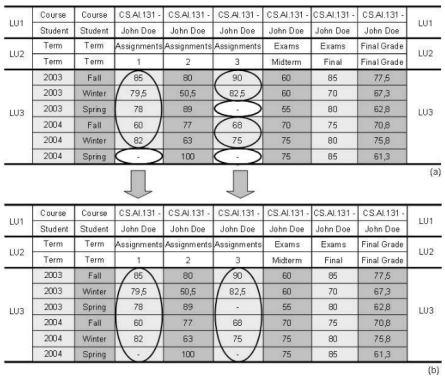


Figure 2.2: Example of clustering values in F-Logic in TARTAR.

The framework was tested on 158 tables crawled from a chosen domain. 85,44% of the tables were transformed correctly. The correctness of transforming the frames was compared to manually labeled frames.

2.2 Table understanding

A table understanding is a process of reconstructing knowledge and relations among concepts from tabular data. The table understanding methods are used in both MOGO and TARTAR. However, the approach used by MOGO is too

unclear and approach used in TARTAR is too complex. Thus, several articles were analyzed in this field [18] [19] [20] [21]. The names of the subsections are the official names of the papers.

2.2.1 Ontology Extraction from Tables on the Web

This article [18] mentions formal representation of generalized table structure based on the adjacency of cells and iterative structures. The representation is derived by applying simple rules defined in the article. Examples of the rules are shown in figure 2.3.
However, an interpretation of the table structure has to be manually defined by the user before deriving representation. By interpretation is meant the location of header cells determining a specific table type.
Such approach could be used for building hierarchy of the table header.

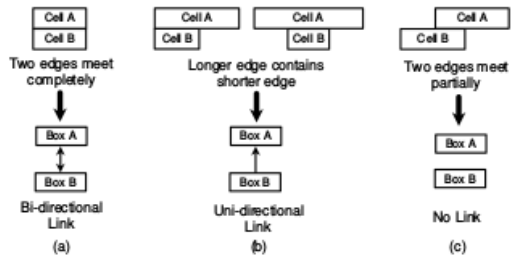


Figure 2.3: Rules used for defining relations among cells.

2.2.2 Data Extraction from Web Tables: the Devil is in the Details

This article [19] is a simplification of the article above 2.2.1. Nevertheless, attention is focused on relating data cells to header cells. Header cells contain names of described elements. Whereas, data cells contain specific value of the described element.
More specifically, all cells of a given rectangular table are divided into four regions, as presented in figure 2.4: stub (empty) cells, row-header cells, column-header cells, delta (data) cells. Then, data cells can be uniquely assigned to a column-header cell to the left and a row-header cell above, if exists. The assignment is called a row/column-header path.

	a	b	c	d	e	f	g	h
1			B					
2			B1			B2		
3			A1	A2	A3	A1	A2	A3
4	C	C1	D11	D12	D13	D14	D15	D16
5		C2	D21	D22	D23	D24	D25	D26

Figure 2.4: Representation of a table. Division of table to 4 categories.

2.2.3 Learning of Ontology from the Web-table

In contrast to the articles above 2.2.2 2.2.1 this article [21] describes the whole process starting with table header location and ending with ontology generation using RDF/OWL. The process is following:

1. **Table header location** - location of the header is critical when deriving ontology from a web table. However, no location approach is explained in the article.

2. **Table type detection** - one of five specified table types is chosen according to the table header location.

3. **Ontology generation** - creating ontology includes relationships as IS-A relationship, class-instance relationship, property relationship, range relationship and domain relationship. Examples of IS-A relationships are presented in figure 2.5.

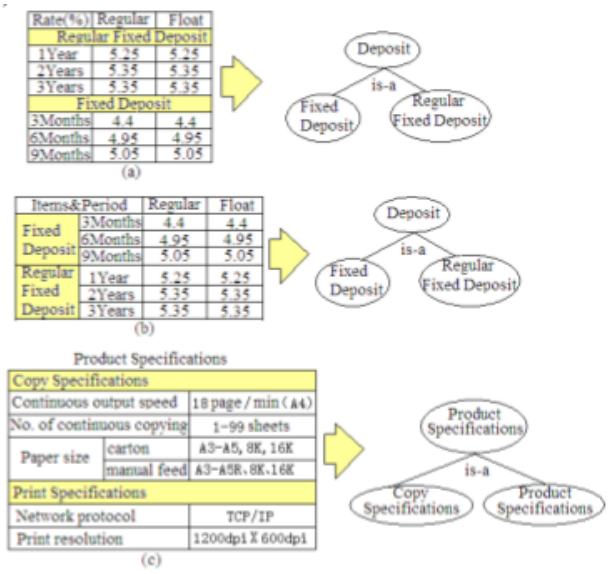


Figure 2.5: Generated ontologies from tables.

Such workflow seems to be reasonable when combined with approaches mentioned in articles 2.2.2 2.2.1.

2.2.4 Compositional Semantic Parsing on Semi-Structured Tables

This article [20] does not serve for building an ontology from a table. Instead, it performs question-answering on a chosen table. The table is converted to a logical form, question is parsed by a semantic parser and then an answer is induced by a knowledge graph created from the table. Table rows become row nodes, strings in table nodes become entity nodes and columns become directed edges from row nodes to the entity nodes of that column. The approach gives precision of 37,1% when answering questions.
Although, this approach was not used in this thesis it is a clever way for creating triples from a single table. An example of created knowledge graph based on triples is depicted in figure 2.6. However, such approach cannot be used for building ontologies from tables with more complex headers that do not consist of only first header row. The method can be rather used for populating instances belonging to specific classes and having only first header row.

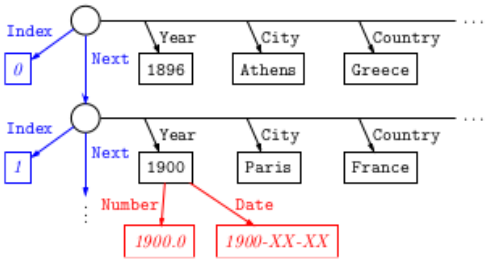


Figure 2.6: Representation of a knowledge graph.

2.3 Populating knowledge databases

This section describes a framework used for populating a product knowledge database. The aim of the thesis is to learn ontologies, not to populate databases with concept's instances. However, such ontologies should be later used for populating databases and thus an existing system was assessed.

2.3.1 FLOPPY

FLOPPY (a Framework for Large-scale Ontology Population of Product Information in E-commerce Store) [22] is a project focusing on e-commerce. It is a semi-automatic approach focusing on database population from WEB domains (Amazon, BestBuy, Walmart and Shopping.com), aided by user-defined

ontology annotations in the form of lexical representations and regular expressions. These annotations are used for looking up for instances values.
The semi-automatic feature is defined as having annotations provided by a user. Its advantage is that synonyms can be defined manually and thus the framework can be executed on different product domains.
However, the population specializes only on a product domain (distinguishing 24 product categories). A custom ontology called OntoProduct ontology has been created for its purposes, fully compatible with GoodRelations and Consumer Electronics Ontology (CEO) [8].

Chapter 3

New system design

Systems mentioned in the previous chapter are all proprietary. Therefore, it seemed challenging to build own system for building domain specific ontologies that could be later integrated with existing ontologies.
The considered parts are based on structure of existing systems assessed in the research. However, the inherent implementation of individual parts allows further adjustment to precisely fit our purposes.

3.1 System workflow

The design of the system consists of four main parts showed in figure 3.1. At the beginning, the system takes one or multiple web pages, having HTML tables, as input. These tables have to be extracted and transformed to an internal representation. The next step is to recognize the table types in order to filter out tables not suitable for building ontologies. It is done by applying machine learning method based on the computed features from the internal representation.
Also, annotation of rows/columns of the table as header/data parts is necessary for better table understanding. Table understanding is the process of building hierarchy among header elements and linking data cells to header cells. Finally, RDF ontologies are generated.

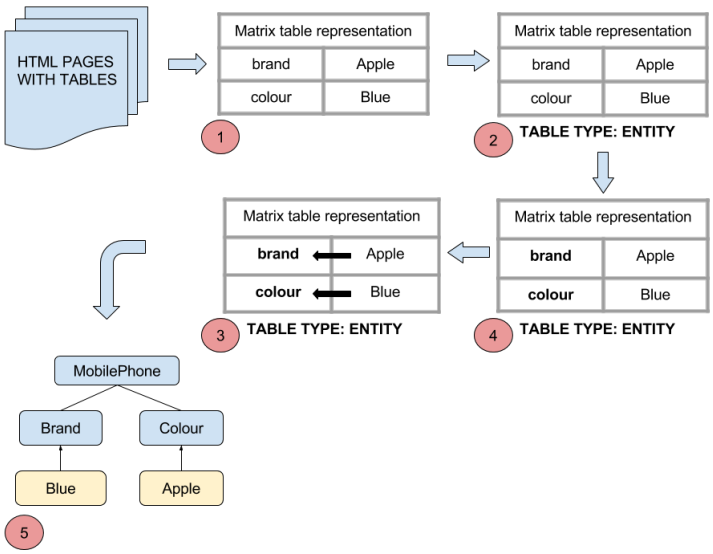


Figure 3.1: Workflow: 1.Matrix representation of a WEB table, 2.WEB table type classification, 3.WEB table header recognition, 4.Table understanding, 5.Ontology generation.

3.1.1 WEB table representation

The table has to be transformed to some internal representation because of two reasons. First is computation of row/column/global features for classification tasks. Second is representation of table cells as objects so the table cells can be divided into data/header cells. Moreover, relations can be specified among the table cells. The approach is presented in detail in chapter 4.

3.1.2 WEB table type classification

There are several types of tables on the WEB, such as ENTITY, RELATION, MATRIX, LAYOUT, OTHER tables. These types have to be distinguished since each table type requires an individual approach for table understanding. Nevertheless, only ENTITY, RELATION and MATRIX tables can be used for building ontologies. The problematics is further described in chapter 5.

3.1.3 WEB table header recognition

The location of the table header is an important step. The header recognition could be skipped if all header cells were marked with <th> element. Unluckily, that is not true. Thus, a classification method has to be chosen in order to

predict whether a table column/row is HEADER/DATA column/row. The discussion about the choice can be found in chapter 6.

3.1.4 Table understanding

The next process is to mine relations among entities from the table. The relations are derived from a table annotated with header location marks. More specifically, the reconstruction of relations uses heuristic rules, defined in chapter 7, resulting in a graph of entities. Obviously, table understanding depends a lot on a correct header location. Additionally, the table understanding is also tightly linked with entity recognition, specified in chapter 8.

3.1.5 Ontology generation

The final process is to generate an ontology based on the understanding of the table. Again, heuristic rules are used in order to "correctly" generate ontology. More about ontology generation is revealed in chapter 9.

Chapter 4

WEB table representation

The whole thesis is about working with HTML tables. Therefore, a method for internally representing a table in the program has to be chosen and implemented.
Most of the methods mentioned in chapter 2 were working with a matrix form. [14] [19]. That is because most of the HTML tables on the WEB represent data in a rectangular form. Thus, transforming the <table> element from a Document Object Model (DOM) tree to a matrix inside internal memory seems like a suitable approach.

4.1 Description of a HTML table

A HTML table allows web content creators to publish data in row and column cells. The elements that can be used to structure the WEB table (<table>, example is viewed in listing 4.1) are:

1. <caption> - represents the title of a table
2. <thead> - represents a block of cells forming a header
3. <tbody> - represents a block of cells forming the data content of the table
4. <tr> - represents a block of cells forming a row
5. <th> - represents a single header cell (the width and height of the cell is defined by colspan/rowspan attribute)
6. <td> - represents a single data cell (the width and height of the cell is defined by colspan/rowspan attribute)

Listing 4.1: Example of HTML table structure

```
<table>
        <thead>
                <tr>
                        <th>Header 1</th>
                        <th>Header 2</th>
                </tr>
        </thead>
        <tbody>
                <tr>
                        <td>Data 1</td>
                        <td>Data 2</td>
                </tr>
        </tbody>
</table>
```

4.2 Table preprocessing

Few pre-processing steps have to be performed once the <table> tag is extracted. One of the preprocessing steps is to remove empty rows. These are rows that do not have any content. The same is done with columns.
Another important preprocessing step is a conversion of the table caption to a first row of the table. Caption usually serves as the title of a table. Therefore, it should be involved in building the ontology. The caption is turned into <th> element with colspan set to the full width of the table. Including the caption as the main part of the header is done because the algorithm used for transforming to Matrix representation takes only <th> and <td> tags.

4.3 Converting DOM to a Matrix Form

After preprocessing, the table can be converted to the matrix representation. Two classes were created in program in order to represent the data correctly. The wrapping class internally containing the table is called **Table** class. The Table has a reference to the matrix of table cells. Each cell is represented by a class called **TableCell**. The TableCell contains reference to the real HTML element as well as it contains the text content and info about its 2D position in the table.
Additionally, colspan/rowspan of cells having value more than 1 (spanning multiple cells) has to be considered in the algorithm. Therefore, instances of TableCell's having such rowspan/colspan are copied to the neighboring cells according to the value in colspan/rowspan. The complete converting algorithm is described as code in algorithm 1.

Algorithm 1 Algorithm for converting DOM structure of HTML table to a Matrix Form

1: **for** $< tr >$ in $< table >$ **do**
2: **for** *element* (<th>/<td>) in $< tr >$ **do**
3:
4: matrix[i,j] = TableCell(*element*, i, j)
5: **if** *span* (rowspan/colspan value) in *element* > 1 **then**
6: copy the TableCell(*element*, i, j) to $span - 1$ following positions

Chapter 5

WEB table type classification

The previous chapter 4 was describing the internal program representation of a web table since it is the cornerstone of this thesis. Nevertheless, the representation does not state any information about the table orientation, type or visual structure. Especially, recognizing table type is very important because tables on the WEB can present various types of information.
The basic assumption is that tables present information referring to some entity or listing information about several similar entities. However, tables are also frequently used for managing the layout of elements on page. The layout tables obviously cannot be used for building a knowledge database. Therefore, an approach for classifying tabular elements is going to be described in this chapter. Such classification model is going to be later used when retrieving tables from pages that can be leveraged for deriving ontologies.

5.1 Related work

The chapter 2 describes systems or specific methods used for building ontology. WEB table type classification was not mentioned in the related work. Nevertheless, it is one of key parts of this thesis. Therefore, a brief summarization of related work to this topic is going to be mentioned in order to select an approach for implementation of this key step. Again, the names of following sections are the names of the original papers.

5.1.1 Detecting tables in HTML documents

This article [23] mentions the basic problem of tables used in HTML pages. The problem is that not all <table> elements indicate the presence of a genuine relational table. Therefore, two methods for GENUINE/NON-GENUINE table detection were used. First is rule-based method and second is a machine learning method.
The rule-based method takes the table structure and splits it according to

user-defined rules. The approach will not be assessed in-depth since it was outperformed by the machine learning approach.
The machine learning approach requires features that capture characteristic easing table types recognition. This paper presents 16 features - 7 layout features, 8 content type features and one word group feature. The layout features are for example average number of rows/columns and their deviations. Whereas, content type features compute histograms of image, hyperlink, form, etc. elements in the table. Thirdly, the word group features depict the textual content of a table on a word level.
The number of tables used for training and testing the method was 11 477. They served as an input to a Decision Tree (DT) classification model. A DT was chosen because it does not require assumptions of feature independence. A 9-fold cross-validation method was used for testing purposes, dividing the data into 9 parts. The final results give the best precision of 97.5% with corresponding recall 94.25%. The results overcome the rule-based method with its 79.46%.

5.1.2 Building the Dresden Web Table Corpus: A Classification Approach

The table detection mentioned in the previous paper has been already greatly studied. In contrast, table layout classification has been paid only little attention. Table layout classification does not consider only genuine and non-genuine table type but rather Vertical/Horizontal listings, Matrix, Layout and Other types. Obviously, knowing the table type helps creating assumptions about the header location and the ultimate meaning of the table.
Similarly as in the previous paper, features characterizing the tables have to be extracted. The first group of features are global - they take the table as a whole. Part of the global features are table structure features such as maximum/average number of features per row/column or average number of characters per cell. The second part of global features are consistency and variation features including standard deviation of cells per row, etc. The last part of global features are ratio features of image, form, etc. elements in cells. Obviously, table structure and consistency + variation features refer to layout features used in the previous paper and the ratio features refer to content type features in the previous paper.
Moreover, there is a second group of features called local features. They consider only the first two rows, first two columns, last row and last column of a table as segments for local features. Local features consist of structural features such as average size of cells in the block, etc. and content ratio features - ratio of header, anchor, image, input, etc. elements in the blocks.
In total, 127 features are taken for each of 2022 labeled tables used for evaluation. The evaluation consists of 100 iterations whose results were averaged. In each iteration, the dataset was randomly split to 90% of training data and

10% of testing data.
The chosen classification models were: Decision Tree, Random Forest and Support Vector Machines (SVM). The results of a Random Forest are presented since it gives the best weighted results over all layout types - precision 85.13% and recall 82.06. [24]

5.2 Approach for WEB table type classification

Approaches mentioned in the research part 5.1 give good results when using machine learning methods. These methods use datasets for building classification models which are then used for prediction of the table type category.
The advantage of the classification model is its possibility of retraining on new data if current dataset does not sufficiently cover the variety of predicted classes. So obviously, the foundation of these methods is a good dataset. Unluckily, none of the ground-truth dataset concerning table types is published publicly on the web. Own datasets have to be created in order to evaluate the chosen methods. And building own datasets includes determining the table types to be recognized.
Finally, suitable classification models have to be chosen for prediction. In this thesis, Random Forest (RF) and Decision Tree (DT) method were chosen since they showed good results in the previous work.

5.2.1 Types of HTML tables

Five different categories of <table> elements have been found on the WEB. The categories are same as in [24]. Only Vertical/Horizontal listings were renamed to RELATION/ENTITY types.

1. ENTITY table contains information about one specific instance of some class.

OS	Android OS, v4.4.4 (KitKat)
RAM	1 GB
Item Weight	159 g
Product Dimensions	15 x 1 x 7.3 cm
Item model number	Desire 620G
Wireless communication technologies	Bluetooth
Connectivity technologies	GSM|(850/900/1800/1900 MHz)|WCDMA|3G|(900/2100 MHz)|WiFi 802.11 b/g/n|WiFi|WiFi Hotspot|Bluetooth|3.5mm Headphone Jack|microUSB v2.0|GPS|FM Radio
Special features	Accelerometer; Photo/video editor; proximity; Camera-Geo tagging
Other camera features	5 MP
Form factor	Touchscreen
Weight	160 Grams
Colour	Milkyway Grey
Battery Power Rating	2100
Whats in the box	Handset, Charger and User Manual

Figure 5.1: Example of ENTITY table.

2. RELATION table contains a list of instances of a shared class.

Rank	Country (or dependent territory)	Population	Date	% of world population	Source
1	China[Note 2]	1,381,590,000	February 17, 2017	18.5%	Official population clock
2	India	1,312,100,000	February 17, 2017	17.5%	Official population clock
3	United States[Note 3]	324,542,000	February 17, 2017	4.34%	Official population clock
4	Indonesia	260,581,000	July 1, 2016	3.48%	UN Projection
5	Brazil	207,109,000	February 17, 2017	2.77%	Official population clock
6	Pakistan	196,387,000	February 17, 2017	2.62%	Official population clock

Figure 5.2: Example of RELATION table.

3. MATRIX table has header both in a column and row. Therefore, data cells have relations to either of the headers.

	Pitching																	
	G	GS	W	L	SV	BS	HLD	CG	SHO	IP	H	R	ER	HR	BB	K	ERA	WHIP	BAA
Home	14	14	5	6	0	0	0	0	0	89.1	88	35	35	7	26	80	3.53	1.28	.262
Away	18	18	7	5	0	0	0	1	0	114.0	102	53	51	17	28	87	4.03	1.14	.238

Figure 5.3: Example of MATRIX table.

4. LAYOUT table serves for structuring the data on a page.

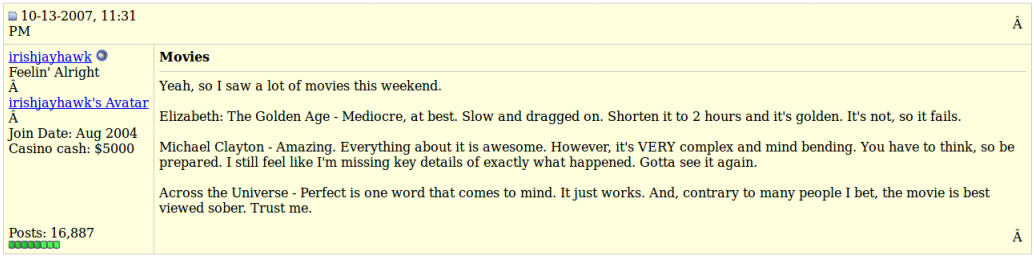


Figure 5.4: Example of LAYOUT table.

5. OTHER is a category for unknown types.

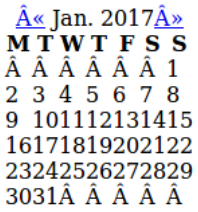


Figure 5.5: Example of OTHER table.

5.2.2 Datasets

A dataset of labeled <table> elements is needed in order to use such machine learning approach. Nevertheless, there was not found a publicly available ground-truth dataset. The only dataset found was WDC Web Table Corpus 2015 (WDC dataset) [25] that was gathered by crawling the WEB for web tables. The discovered tables were labeled by a similar machine learning method as the one proposed here. Thus, the dataset is not ground-truth and has to be manually verified.
Another dataset was created from mobile phones domains since it is the target domain of this thesis.

5.2.2.1 Common WEB crawl dataset

The WDC Web Table Corpus 2015 project published JSON (JavaScript Object Notation) files containing links to pages that have tabular data together with the table position inside the page. Additionally, probable type (predicted by machine learning algorithm [24]) of the table was provided corresponding to one of the five categories specified in 5.2.1. Thus, 1168 tables were downloaded and their table type was manually checked and corrected, as presented in table 5.1, in order to have a good quality dataset. Still, 1168 is only a small part of the whole corpus but it would be time consuming to validate all the tables in the corpus.

Moreover, multiple similar tables were taken from one WEB domain. As a result, the dataset contains groups of tables that are structurally identical but have different textual content.

Table type	Tables count
ENTITY	526
RELATION	420
LAYOUT	142
MATRIX	24
OTHER	56

Table 5.1: Table types count in the WDC dataset.

5.2.2.2 Product domains dataset

Another dataset was created as a collection of tables containing specifications of mobile phones found on relevant WEB domains (english/czech WEB domains). This Product domain dataset (Product dataset) was used only as a test dataset. Ten pages were taken from each domain and tables containing mobile phone specifications were extracted.
The dataset was also enriched by tables from Wikipedia and also by a set of 5 RELATION tables. The RELATION tables contain finance data and country instances. The enrichment was done in order to assess the general use of the method on various domains.
Importantly, different domains contain different number of pages because the specifications in WEB domains were split into multiple sub-tables. The overview of number of tables for each domain is in table 5.2.

Domain	Tables count
amazon.com	20
buymobiles.net	10
datart.cz	80
electroworld.cz	115
expert.cz	10
gadgetsndtv.com	61
heureka.cz	10
mall.cz	10
snapdeal.com	138
wikipedia.org	10
relational	5

Table 5.2: Distribution of tables in the Product dataset.

5.2.3 Features used for classification

Some features have to be assigned to each table in order to classify it. In this task two different groups of features are considered. First group is called Global features and second group is called Local features.
Global features consider characteristics of the whole table and all the rows/-columns together. More precisely, there are 10 features taking all table cells, 4 features taking all columns cells and 4 features taking all rows cells. There are 18 global features in total.
Local features consider only characteristics of the first two rows/columns and the last row/column. That means 6 blocks, each having 18 features. In total, it aggregates to 108 local features.
So there are 126 global and local features used for this classification task. Most of the 126 features were taken from [24].

5.2.3.1 Global features

Global features can be divided into three sub-categories: features for all cells, features for row blocks and features for column blocks.

Features for all cells

First three features are strictly numerical and would belong to the table structure category, mentioned in the research. Next features consider different content type categories and compute their histograms. These features belong to the content ratio features category. Such content type categories are: IMAGE, FORM, HYPERLINK, ALHABETIC, NUMERIC, EMPTY, OTHER cells. The summary of the features is below:

1. Average cell length
2. Maximum cell length
3. Standard deviation of the cell length
4. Percentage of image cells
5. Percentage of form cells
6. Percentage of hyperlink cells
7. Percentage of predominantly alphabetic cells
8. Percentage of predominantly numeric cells
9. Percentage of empty cells
10. Percentage of other cells

Features for row and column blocks

These features take either row/column blocks and compute their characteristics. All of the features belong to the table structure group and are following:

1. Average number of cells per blocks
2. Maximum number of cells per blocks
3. Standard deviation of number of cells per blocks
4. Cumulative length consistency

The definition of Cumulative length consistency (CLC) is following - it computes the consistency of the table segment entries with respect to their size. In other words, segments where cells content lengths are below twice the size of the average cell keep a value of the CLC according to the equation. Whereas, segments with cells that have content twice and more longer have a constant value 1 distinguish them from segments with values closer to average. In detail, for each cell c, the size s is taken as the number of characters and the cumulative length consistency (CLC) is computed per row or column as follows, where s_{avg_i} is the average cell size of table segment (i.e. column or row) i:

$$CLC_i = \sum_c 0.5 - x_c, \text{ where } x_c = min(\frac{\mid s_c - s_{avg_i} \mid}{s_{avg_i}}, 1) \tag{5.1}$$

These consistency scores are averaged across all rows (CLC_i) and columns (CLC_i).

5.2.3.2 Local features

Computing local features for all rows/columns would lead to unknown (high) number of features since dimensions of tables differ. Therefore, local features are computed only for first two rows/columns and the last row/column, as viewed in figure 5.6. These rows/columns are most relevant when deciding about the table type and should be sufficient.

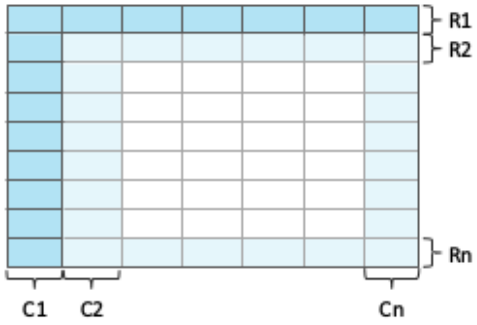


Figure 5.6: Rows/columns used for computing local features. The figure was borrowed from [24].

The first two features are of a table structure category according to categories mentioned in section 5.1.2. The other features belong to the content ratio category. Such block features are:

1. Average cell length
2. Standard deviation of cell length
3. Percentage of cells having <th> tag
4. Percentage of cells having <a> tag
5. Percentage of cells having <img> tag
6. Percentage of cells having <input> tag
7. Percentage of cells having <select> tag
8. Percentage of cells having font tags - <b>, , <u>, <i>, <font>
9. Percentage of cells having
 tag
10. Percentage of cells containing colon - ":"
11. Percentage of cells containing number
12. Percentage of cells containing only number
13. Percentage of cells containing non-empty cells
14. Percentage of cells having <ul> tag
15. Percentage of cells having <ol> tag
16. Percentage of cells containing comma - ","
17. Percentage of cells containing brackets - "(" or ")"
18. Percentage of cells containing affirmative/negative - Yes/No (English), Ano/Ne (Czech) - ENTITY table types usually contain these

5.2.4 Results of WEB table type classification

The DT and RF methods are going to be evaluated. The datasets specified in 5.2.2 are taken and 126 features are computed for each table, taking both global and local features. Then, the following approach was used for evaluating the methods:

1. Split WDC dataset by 10-fold stratified cross-validation (10f-scv)
2. Perform training and testing of the model according to the splits and give average results on individual classes
3. Train model on the whole WDC dataset
4. Test model on the Product dataset and give results for individual domains

The above mentioned process uses 10f-scv in order to select model that is stable on different parts of data. Moreover, the evaluation is divided into two parts. First, the models are trained on the WDC dataset and tested on an independent part of the same dataset. This process is performed in order to estimate classification scores of all five table types.
Next, the model is trained on the whole WDC dataset and then tested on the Product dataset. This step is performed in order to evaluate the model on tables from a target domain specified in assignment of the thesis. Based on these two tests, the final model should give good results on all the five table types as well as on the product domain. Keeping these constraints should enable the use of the model on various domains in the future.
Finally, precision of the models tested on WDC dataset is counted as an average precision of classification of individual classes. In contrast, the precision on all tables averages predictions of tables from all categories together. Therefore it is higher then average precision of classes since majority of tables are either ENTITY or RELATION with high precision scores.
Testing on the Product dataset assumes precision as the number of correctly predicted table types on a single domain.

5.2.4.1 Decision tree

The choice of a DT heavily depends on its power of feature selection. Indeed, the first splits in the tree select the most important features within the dataset. Consequently, a series of several decision rules can result into a strong classification statement describing a specific table type. On the other hand, the DT can be over-fitted by exploring features values that would rarely occur outside of the training dataset. Therefore, additional pruning of the DT was performed by iteratively limiting the maximum number of leaf nodes. The pruned tree with best classification scores is presented.

Computed scores are presented for WDC dataset in table 5.3 and for Product dataset in 5.4.

P (%) \ Method	**DT UNPRUN**	**DT PRUN (101 LN)**
P - All tables	84.4	85.4
P - LAYOUT tables	69	71.7
P - OTHER tables	63.3	65.7
P - RELATION tables	85.7	85.5
P - MATRIX tables	55	55
P - ENTITY class	91.1	92
P - AVG of classes	**72.8**	**74.1**

Table 5.3: Results of DT table type classification on WDC dataset. LN - leaf nodes, P - precision, PRUN - pruned, UNPRUN - unpruned.

P (%) \ Method	**DT UNPRUN**	**DT PRUN (101 LN)**
P - buymobiles	100	100
P - heureka	100	100
P - expert	90	90
P - gadgets	88.5	100
P - electroworld	75.7	93.9
P - mall	100	100
P - amazon	100	100
P - wikipedia	40	50
P - relational	60	60
P - datart	0	10
P - snapdeal	86.2	57.2
P - AVG of domains	**76.4**	**78.3**

Table 5.4: Results of DT table type classification on Product dataset. LN - leaf nodes, P - precision, PRUN - pruned, UNPRUN - unpruned.

5.2.4.2 Random forest

RF method arose from the DT method. RF takes multiple DT's that are trained on random samples of the training data. Next, the built trees vote about the classification of a new sample. The multiplicity of deep DT's trained on different parts of the same dataset together with final averaging should avoid over-fitting.
Obviously, the number of trees is a parameter that has to be set. Thus, 100 trees were chosen as it was selected in the related work. Additionally, max_features parameter, that defines the number of possible features when looking for the best split, was set to either values "auto" or "log2(n_features)".

On the contrary, no experiments were done with pruning since RF should avoid overfitting.
Computed scores are presented for WDC dataset in table 5.5 and for Product dataset in table 5.6.

P (%) \ Method	**RF 100 auto**	**RF 100 log2**
P - All tables	84.4	89.7
P - LAYOUT tables	71.9	67.6
P - OTHER tables	68	67.7
P - RELATION tables	95.7	94.8
P - MATRIX tables	50	50
P - ENTITY tables	95.6	95.8
P - AVG of classes	**76.2**	**75.2**

Table 5.5: Results of RF table type classification on WDC dataset. 100 - number of trees in RF, auto - automatic max_features parameter selection, log2 - max_features value set to log2(n_features), P - precision.

P (%) \ Method	**RF 100 auto**	**RF 100 log2**
P - buymobiles	100	100
P - heureka	0	0
P - expert	100	70
P - gadgetsndtv	100	100
P - electroworld	100	100
P - mall	100	100
P - amazon	100	100
P - wikipedia	0	10
P - relational	100	100
P - datart	2.5	0
P - snapdeal	30.4	17.4
P - AVG of domains	**66.6**	**63.4**

Table 5.6: Results of RF table type classification on Product dataset. 100 - number of trees in RF, auto - automatic max_features parameter selection, log2 - max_features value set to log2(n_features), P - precision.

5.2.4.3 Comparison of results

Both DT and RF seem to have similar precision when tested on part of the WDC dataset, around 74% and 76%. Both giving around 50% precision on the MATRIX table type since its distribution in the dataset is minimal.
Whereas, testing on the Product dataset results into 78% and 66%. The interpretation is following - the tables from datart, heureka and snapdeal have

no highlighting (<th>, <b>, etc.) of the header cells in the first column. Therefore, the tables are incorrectly but logically classified as RELATION tables. The same class is assigned to the wikipedia tables because of its complex header including image. The resolution of the wikipedia issue would be to expand the WDC dataset by sufficient amount of tables with varying structure in order to cover unknown table structures.
Generally, the RF method is more stable (due to multiplicity of decision trees) when classifying tables from one domain. According to the results, 7 out of 11 (having precision 1.0) WEB domains were classified correctly. In contrast, DT gives better average results but classifies correctly only 4 out of 11 WEB domains.
Examples of incorrectly classified tables are given in figures 5.7, 5.8, 5.9, 5.10.

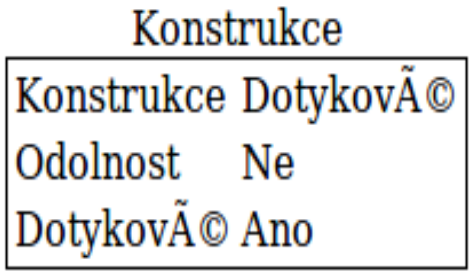
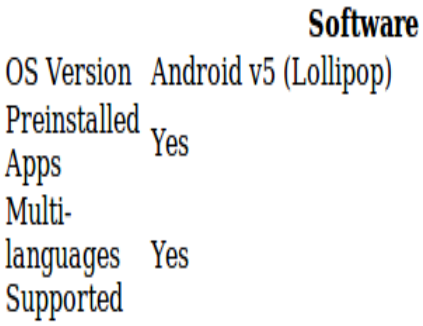


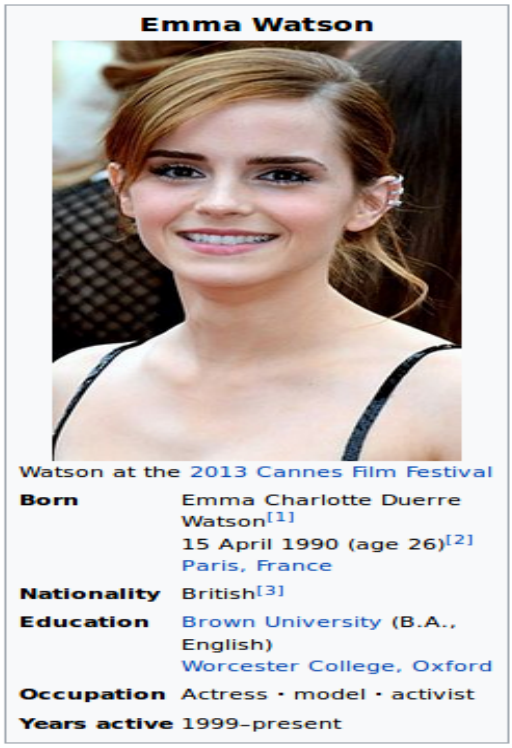
Figure 5.7: Table from datart.cz

Figure 5.8: Table from heureka.cz



Software	
OS Version	Android v5 (Lollipop)
Preinstalled Apps	Yes
Multi-languages Supported	Yes

Figure 5.9: Table from snapdeal.com



Emma Watson

Watson at the 2013 Cannes Film Festival

Born	Emma Charlotte Duerre Watson[1] 15 April 1990 (age 26)[2] Paris, France
Nationality	British[3]
Education	Brown University (B.A., English) Worcester College, Oxford
Occupation	Actress · model · activist
Years active	1999–present

Figure 5.10: Table from wikipedia.org

Performance verification

The selected approach is very similar to the approach mentioned in [24]. Therefore, corresponding results are going to be compared.
The sizes of datasets differ - 2022 (paper) vs. 1168 (thesis) tables. The difference in features is 127 (paper) vs. 126 (thesis) features. The distribution of classes in the datasets is presented in figure 5.7.

Table type	% of classes in Thesis	% of classes in Paper
ENTITY	45	22
RELATION	35.9	15.3
LAYOUT	12.1	54.4
MATRIX	2.2	1.66
OTHER	4.8	7

Table 5.7: Distribution of table classes in datasets.

The table 5.8 presents results on individual classes both in the paper and in the thesis. The paper outperforms the thesis in LAYOUT tables since they

comprised more than half of the paper's dataset. However, the thesis gives better results in classification of the ENTITY and RELATION tables that are the key tables to be used in building knowledge databases.
Obviously, the results were computed on different datasets, with not completely similar features. Thus, the results serve for general comparison.

Table type	**P (%) - Thesis**	**P (%) - Paper**
ENTITY	95.6	85
RELATION	96.7	68
LAYOUT	71.9	93
MATRIX	50	36
OTHER	68	76
P - AVG of classes	**76.2**	**72**

Table 5.8: Performance verification of WEB table type classification results. P - precision.

Chapter 6

WEB table header recognition

The last important step before Table understanding is to distinguish header and data cells. The distinguishing leads to discovery of relations among cells. These relations are key parts when learning ontology from tabular data.
Nevertheless, the task of header recognition is not so trivial due to the lack of annotations of header cells, mentioned in chapter 3. Header can be classical text, sometimes with different font, sometimes with colon or spanning multiple cells. Thus, a classification method will be used for classifying table rows/columns as DATA/HEADER.

6.1 Related work

There are not many articles contributing to this topic. Mainly because the <th> element enables annotation of the header cells. This element should be the key element used for understanding the table. Nevertheless, not all tables on the WEB use the <th> tag.
Therefore a paper exploiting the machine learning method for header classification has been studied [26]. Again the name of the following section is the original name of the paper.

6.1.1 Table Header Detection and Classification

This paper describes two methods for table header classification. First, a heuristic method is used. Second, a machine learning method is tested on two datasets. The size of the datasets is 130 and 120 tables. These tables are gathered from the CiteSeerX document repository [27]. A comparison of these two methods is necessary when choosing an approach in this thesis.
The heuristic approach takes pairs of corresponding rows. A similarity score is computed for these rows. Then, the first local extrema (first valley) is chosen as the separating header line while iterating in top-bottom/bottom-top block table pass.

In detail, the row pair similarity is following. A similarity score is computed for corresponding cells (cells that overlap horizontally for rows). The similarity score consists of weighted font size score, character number score, overlap score, alignment score. Then, the scores for cell pairs in compared rows are averaged and supplemented with a row score considering number of cells in the two rows. This approach reaches up to 60% of precision on the two selected datasets.
Next, a machine learning was proposed. At the beginning, relevant features are mentioned. There are two groups of features - single row features and neighboring row features. The single row features contain number of cells, average number of characters, percentage of alphabetical/numerical/symbolic cells, etc. Whereas, neighboring row features are percentage of spanning cells, average alignment, percentage of same data-types in cells, etc.
These features serve as input to three types of classification models - Random Forest (RF, 100 trees), Support Vector Machines (SVM), Logistic regression (LR). The results of 10-fold cross-validation are presented in table 6.1.

Method	**P - rows (%)**	**P - columns (%)**
RF	97	98
SVM	92	86
LR	84	94

Table 6.1: Result of column/row header detection of selected methods. P - precision.

According to the paper, RF gives best performance and is able to automatically choose the most useful features. Moreover, it has bagging mechanism that selects training samples which could reduce variance and avoid over-fitting. The SVM may probably be affected by the unbalanced number of header and data cases. The low performance of logistic regression may be caused by relative limited size of datasets [26].

6.2 Approach for WEB table header classification

The machine learning approach mentioned in the research part gives excellent results. Therefore, machine learning methods - Random Forest and Decision Tree were chosen for this task. Again, the foundation of these methods is a good dataset. Therefore, the same datasets mentioned in 5.2.2 are used.

6.2.1 Dataset

The same datasets (section 5.2.2.1, 5.2.2.2) that were used for table type classification will be used for header classification task. Each table has label HEADER or DATA for each column and row. First, all tables will be used

for training the model. Second, only ENTITY tables will be used for training the model since entity tables are going to be used for building the ontologies. The results will be compared.

6.2.2 Features used for classification

Some features have to be assigned to each row/column (block). Such features can be divided into two categories: Single block features and neighboring block features.

6.2.2.1 Single block features

Each row/column is assinged 16 single block features. Such features are:

1. Block index in the table structure
2. Number of block cells
3. Average cell length among block cells
4. Total number of characters among block cells
5. Percentage of numeric characters among block cells
6. Percentage of alphabetic characters among block cells
7. Percentage of special characters among block cells
8. Percentage of punctuation characters among block cells
9. Percentage of emphasized cells among block cells - containing tags <th>, <i>, <b>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>
10. Average cell string similarity ratio - all cell strings are encoded into string that containts only categorical characters as digit, alphabetical, punctuation and symbolic. Cells are then compared to each other according to [28]. Average similarity ratio is returned.
11. Standard deviation of cell length among block cells
12. Percentage of cells with hyperlinks among block cells
13. Percentage of cells with colon among block cells - ":"
14. Percentage of cells with comma among block cells - ","
15. Percentage of cells with brackets among block cells - "(" or ")"
16. Percentage of cells with affirmative/negative words among block cells - Yes/No (English), Ano/Ne (Czech)

6.2.2.2 Neighboring block features

Neighboring block features could help in capturing differences among data/-header clusters. Each block gets extra features that compare current block against the next block in the table structure. The comparison is made against the row below when computing for rows, against column to the right when computing for columns. Such features are:

1. Percentage of spanning cells - a cell spanning over multiple cells in the lower row or in the column to the right
2. Number of cells difference - abs(number of cells upper – number of cells lower) / (number of cells upper + number of cells lower). Header rows often have missing cells, similar to the "number of cells" for single rows.
3. Percentage of same cell data type. Here data type refers to alphabetical, digit and symbol.

6.2.3 Results of table header classification

Two classification methods were tried for this task. First is DT and second is RF. The methods work with 19 features, taking both single block features and neighboring block features. Following approach was used for evaluating the methods:

1. Split WDC dataset by 10-fold stratified cross-validation (10f-scv)
2. Perform training and testing of the model according to the splits and give average results for accuracy on individual classes
3. Split WDC dataset having only ENTITY tables (ENTITY WDC Dataset) by 10-fold stratified cross-validation (10f-scv)
4. Perform training and testing of the model according to the splits and give average results for accuracy on individual classes
5. Train model on the whole WDC dataset
6. Test model on the whole Product dataset and give results for individual domains
7. Train model on the ENTITY WDC Dataset
8. Test model on the ENTITY Product dataset and give results for individual domains

The approach above uses 10f-scv in order to select model that is stable on different parts of data. Moreover, the evaluation is divided into two parts. First, the models are trained on the WDC dataset/ENTITY WDC dataset and tested on an independent part of the same dataset. This process is performed in order to estimate classification scores of the HEADER/DATA blocks on

all types of tables. Further, focusing on the ENTITY tables should improve understanding the product specification tables.
Next, the model is trained on the whole WDC dataset/ENTITY WDC dataset and then tested on the Product dataset. This step is performed in order to evaluate the model on tables from a target domain specified in assignment of the thesis.
As a result, training on the ENTITY WDC dataset is expected to give better results since it specializes on the recognition of HEADER/DATA blocks in ENTITY tables. Training on the whole WDC dataset contains noise created by blocks of LAYOUT, OTHER, MATRIX and RELATION tables.

6.2.3.1 Decision tree

The reasons for selecting DT were the same as in section 5.2.4.1. Computed scores are presented for WDC dataset in table 6.2 and for Product dataset in table 6.3.

P (%) \ Method	**DT E**	**DT E PRUN (48 LN)**	**DT A**	**DT A PRUN (48 LN)**
P - All blocks	99.2	99.3	98.2	98.5
P - HEADER	95.5	95.5	87.3	84.8
P - DATA	99.6	99.7	98.9	99.4
P - AVG of classes	**97.5**	**97.6**	**93.1**	**92.1**

Table 6.2: Results of DT table header classification on WDC dataset. E - ENTITY tables only, A - All tables, PRUN - pruned, LN - leaf nodes, P - precision.

P (%) \ Method	**DT E**	**DT E PRUN (48 LN)**	**DT A**	**DT A PRUN (48 LN)**
P - buymobiles	100	100	80	100
P - heureka	100	0	0	0
P - expert	100	100	20	100
P - gadgets	83.8	73.5	69.1	83.8
P - electroworld	66.1	63.2	77	95.4
P - mall	100	100	0	0
P - amazon	95	95	80	95
P - wikipedia	30	10	70	60
P - datart	77.6	20.9	26.9	69.4
P - snapdeal	81.4	78.6	58.8	77.1
P - AVG of domains	**83.4**	**64.1**	**48.2**	**68.1**

Table 6.3: Results of DT table header classification on Product dataset. E - ENTITY tables only, A - All tables, PRUN - pruned, LN - leaf nodes, P - precision.

6.2.3.2 Random forest

The reasons for selecting RF were the same as in section 5.2.4.2. 100 tree estimators were used as mentioned the paper [26]. Computed scores are presented for WDC dataset in table 6.4 and for Product dataset in table 6.5.

P (%) \ Method	**RF E**	**RF A**
P - All blocks	99.5	98.9
P - HEADER	95.9	87.1
P - DATA	99.9	99.6
P - AVG of classes	**97.9**	**93.4**

Table 6.4: Results of RF table header classification on WDC dataset. E - Entity tables only, A - All tables, P - precision.

P (%) \ Method	**RF E**	**RF A**
P - buymobiles	100	100
P - heureka	100	0
P - expert	100	40
P - gadgets	83.8	72.1
P - electroworld	94.3	93.1
P - mall	100	100
P - amazon	95	95
P - wikipedia	80	80
P - datart	28.4	13.4
P - snapdeal	80	53.6
P - AVG of domains	**86.1**	**64.7**

Table 6.5: Results of RF table header classification on Product dataset. E - Entity tables only, A - All tables, P - precision.

6.2.3.3 Comparison of the results

Both DT and RF seem to have similar average precision on classes when tested on part of the ENTITY WDC dataset, around 97.5%.
In contrast, testing on the whole WDC dataset decreases the precision of HEADER block classification to 87%. Clearly, excellent HEADER block recognition is critical for precise ontology generation, so 87% is insufficient. Precision of the DATA block stays the same since it comprises much bigger part of the dataset (table usually has only one or two HEADER blocks but multiple DATA blocks).
Similarly, DT and RF methods gave similar results on the Product dataset, 84% and 86%.

Performance verification
The classification approach was inspired by paper [26]. This paper takes 175 tables parsed from PDF (Portable Document Format) files, 11 single block features, 8 neighbor block features. In comparison, the approach proposed in this thesis takes 16 single block features and 3 neighbor block features.
The RF proposed in the paper gives precision of 91% (the result from the paper was averaged from correctly classified HEADER/DATA results). The RF used in this thesis reached precision 93.4% when trained and tested on all types of tables. The features used in both works slightly differ, as well as the datasets. Therefore, these results are considered only for general comparison.

Chapter 7

Table understanding

Table understanding is the process of deriving relations among cells in tables. These relations are the key factor needed for generating ontologies. More specifically, the cells are divided into two categories. First are header cells, marked by WEB table header classification. Second are data cells, giving actual values of the properties specified in header.
Obviously, the need is to create a hierarchy between header cells. Consequently, link the data cells to correct header elements.

7.1 Hierarchy of a header

Reconstructing the hierarchy of the table header is the first step to be done. The reconstruction can be done by applying simple rules defined in figure 7.1. Similar to the ones defined in section 2.2.1.

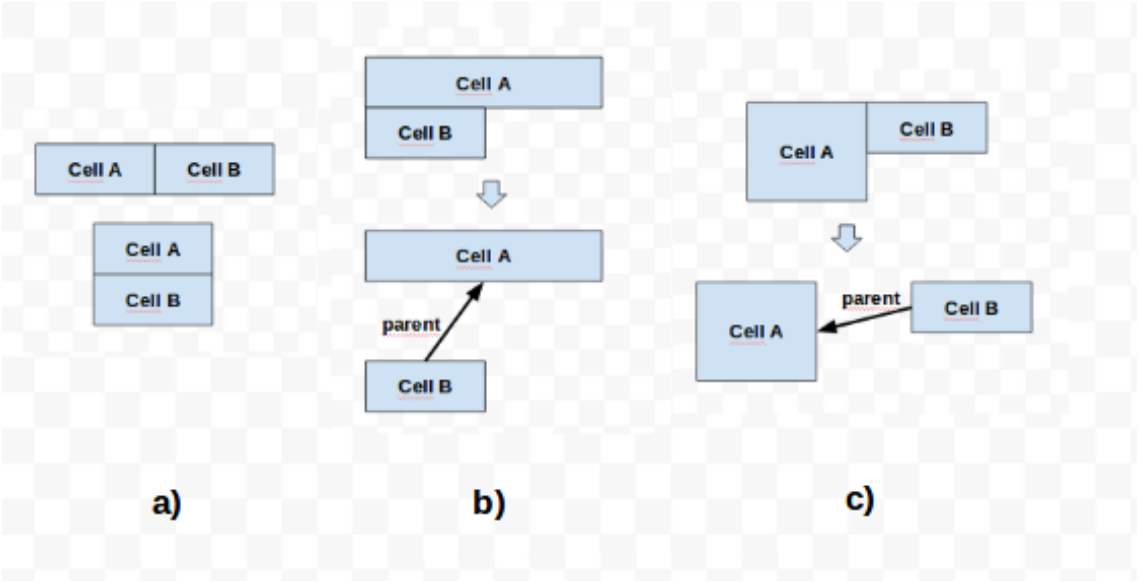


Figure 7.1: Rules for reconstructing relations.

The rule **a)** says that two neighboring cells do not have a specific relation if their height/width (colspan/rowspan value) is the same. The rules **b), c)** define that cell above/"to the left" is a parent of the other cell if its height/width is bigger.
Moreover, partial overlaps are considered to be the same category as **b), c)** in this work. The section 2.2.1 categorizes it as **a)**.
Then, the algorithm for the reconstruction is formally described in algorithm 2:

Algorithm 2 Algorithm for recognizing header hierarchy

```
1:  for cell in headerCells do
2:
3:      for leftCell in cell.cellsToTheLeft do
4:
5:          if leftCell is higher than cell then
6:              Append leftCell to parents of cell
7:              BREAK
8:
9:      for upperCell in cell.cellsAbove do
10:
11:         if upperCell is wider than cell then
12:             Append upperCell to parents of cell
13:             BREAK
```

7.2 Matching data cell to header cells

There exists a relation between a data and header cell in each table. The question is whether it can be recognized by a machine approach. One method, presented in [19], links the cells by row/column-header paths. The approach is very easy. Moreover, it works for most tables found on the WEB.
In detail, the path from the data cell is searched for the upper/left header cell, if they exist. Additionally, sequence of ancestors of the left header cell is retrieved. Then, it is checked whether it contains the upper header cell. If yes, the upper header cell is already an ancestor and thus cannot serve as direct parent of the data cell. The same check is done vice versa.

Then, the algorithm is formally described in algorithm 3:

Algorithm 3 Algorithm for matching data cell to header cell

```
1:  for cell in dataCells do
2:
3:      leftParent = NULL
4:      upperParent = NULL
5:      for leftCell in cell.cellsToTheLeft do
6:
7:          if leftCell is header then
8:              leftParent = leftCell
9:              BREAK
10:
11:     for upperCell in cell.cellsAbove do
12:
13:         if upperCell is header then
14:             upperParent = upperCell
15:             BREAK
16:
17:     if leftParent != NULL and leftParent not in upperCell.ancestors
    then
18:         Append leftParent to cell parents
19:
20:     if upperParent != NULL and upperParent not in leftCell.ancestors
    then
21:         Append upperParent to cell parents
```

7.3 Result of the decomposition

A tree-like structure of relations among individual cells in a table can be generated by applying algorithms mentioned above. Example of such result is presented in figure 7.2.

Battery	
battery type	Li-Ion
battery capacity	2500 mAh

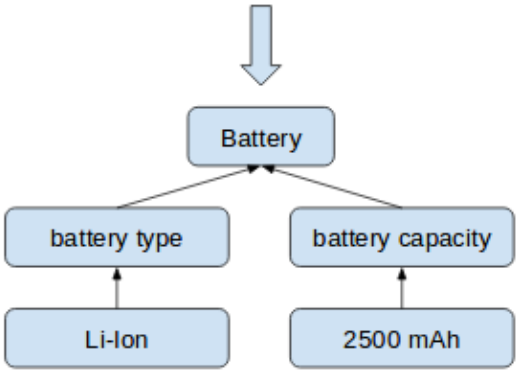


Figure 7.2: Resulting tree-like structure after decomposition.

Chapter 8

Entity and Quantitative value recognition in table cells

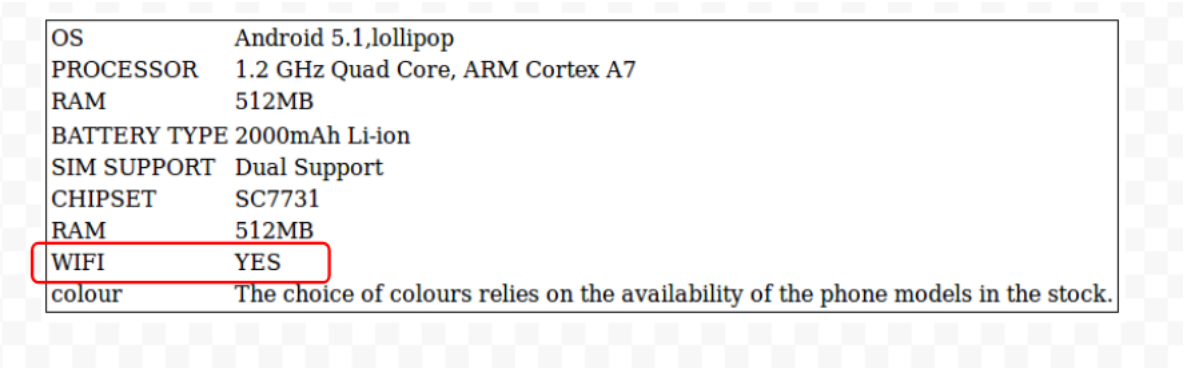
Another important step is to analyze the content of individual cells. The cell's content can contain unknown entities as well as numerical values supplemented by a unit of measurement (Quantitative value).
There are basically three categories of elements to be recognized in ENTITY tables, like the one in figure 8.1. These categories are: entities in header cell, entities in data cell, numeric value content. The categories are going to be shown on a specific table that defines parameters of a some mobile phone product. Lets assume that the header rows/columns would be correctly classified. So the first column would be marked as header column.

OS	Android 5.1,lollipop
PROCESSOR	1.2 GHz Quad Core, ARM Cortex A7
RAM	512MB
BATTERY TYPE	2000mAh Li-ion
SIM SUPPORT	Dual Support
CHIPSET	SC7731
RAM	512MB
WIFI	YES
colour	The choice of colours relies on the availability of the phone models in the stock.

Figure 8.1: Specifications table referring to some mobile phone.

8.1 Entity recognition in header cells

It is quite frequent that some parameters of ENTITY tables are marked by affirmative/negative words. These parameters specify existence of a part of the described entity, also call a meronymy. Thus, the header cell can be marked as being an entity. In figure 8.2, "Wifi" would be an entity.

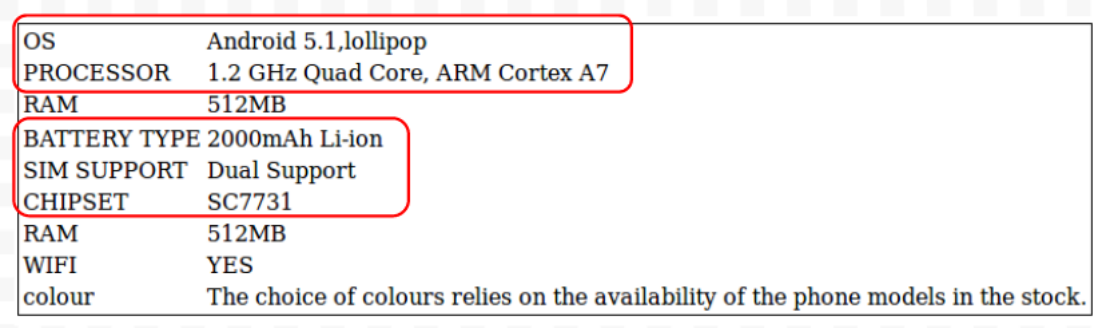


OS	Android 5.1,lollipop
PROCESSOR	1.2 GHz Quad Core, ARM Cortex A7
RAM	512MB
BATTERY TYPE	2000mAh Li-ion
SIM SUPPORT	Dual Support
CHIPSET	SC7731
RAM	512MB
WIFI	YES
colour	The choice of colours relies on the availability of the phone models in the stock.

Figure 8.2: Parameter expressed by affirmative/negative words.

8.2 Entity recognition in data cells

In other cases, the entities can be specified in data cells. Multiple entities can be specified in one cell. However, they are usually separated by a comma. Then, the entities in the data cell are instances of the concept mentioned in its parent cell.
In the example presented in figure 8.3 "Android 5.1" would be correctly linked to its parent entity "OS". Thus, Android 5.1 would be an instance of an Operating System entity.



OS	Android 5.1,lollipop
PROCESSOR	1.2 GHz Quad Core, ARM Cortex A7
RAM	512MB
BATTERY TYPE	2000mAh Li-ion
SIM SUPPORT	Dual Support
CHIPSET	SC7731
RAM	512MB
WIFI	YES
colour	The choice of colours relies on the availability of the phone models in the stock.

Figure 8.3: Entities recognition in data cells.

8.3 Quantitative value recognition in data cells

Additionally, numerical values and their corresponding units of measurement can be parsed from the data cells. The parsing is done by regular expressions. Considering only cells that begin with a number and are followed by a single word that should denote the unit. Also, only numerical values without unit are taken, as shown in figure 8.4.

OS	Android 5.1,lollipop
PROCESSOR	1.2 GHz Quad Core, ARM Cortex A7
RAM	512MB
BATTERY TYPE	2000mAh Li-ion
SIM SUPPORT	Dual Support
CHIPSET	SC7731
RAM	512MB
WIFI	YES
colour	The choice of colours relies on the availability of the phone models in the stock.

Figure 8.4: Parsing numerical values with units of measument.

8.4 Natural language as data cells content

Some data cells describe the parameters in longer sentences. These sentences, as in figure 8.5, are difficult to parse for their numerical and unit values since there could be lots of misunderstanding. Lets consider value "this phone is sold with 4GB memory, not 8GB". The parser would find both values but one of them is not correct.
Also, there are no entities that could be recognized by splitting the sentence by commas. Therefore, a filter is applied to all the comma-split parts. Such filter computes number of predefined stop-words (a list of stop-words for several languages) and filters out parts that have more stop-words (indicating natural language) than some threshold (set to 2 in the thesis).

OS	Android 5.1,lollipop
PROCESSOR	1.2 GHz Quad Core, ARM Cortex A7
RAM	512MB
BATTERY TYPE	2000mAh Li-ion
SIM SUPPORT	Dual Support
CHIPSET	SC7731
RAM	512MB
WIFI	YES
colour	The choice of colours relies on the availability of the phone models in the stock.

Figure 8.5: Example of natural language content.

Chapter 9

Ontology generation

The ontology generation takes the tree-like structure of a table decomposed to relations and generates a file according to syntactical and structural rules. The thesis specializes on a product domain. Therefore, an ontology defining Mobile phones was taken from GoodRelation ontologies [8] as a reference model, also called a gold standard ontology. Hence, the final ontology will consist of the same building blocks as the gold standard ontology. Moreover, the generated ontologies can be later linked to relevant classes defined in ontology standards, such as Schema.org [9], GoodRelation [8], etc.

9.1 Building blocks

The building blocks consist of a mix of standards used for describing ontologies. The basic standard for creating ontologies is called RDF (Resource Description Framework). Later, several standards such as RDFS, OWL, etc., were created as extensions of RDF.

9.1.1 Resource description framework + schema

The RDF data model [16] is similar to classical conceptual modeling approaches (such as entity–relationship or class diagrams). It is based on making statements about resources (in particular "web resources") in the form of subject–predicate–object expressions (triples). The subject denotes the resource, and the predicate denotes a relationship between the subject and the object.
Additionally, RDFS (Resource description framework schema) defines more constructs built on the limited vocabulary of RDF.
RDFS is a set of classes and properties, presented in table 9.1, using the RDF extensible knowledge representation data model. Its main purpose is the description of new ontologies intended to structure RDF resources.

Name	Description
rdfs:Class	Classes define set of similar objects with some properties.
rdf:Property	Property describes a relation between subject resources and object resources.
rdfs:type	Type defines that the source (subject) is an instance of a specific class.
rdfs:range	Range is used to state that the values of a property are instances of one or more classes.
rdfs:domain	Domain is used to state that any resource that has a given property is an instance of one or more classes.

Table 9.1: RDF/RDFS building blocks.

9.1.2 Web ontology language

The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs [17]. OWL building blocks are described in table 9.2.

Name	Description
owl:Class	An owl:Class defines a group of individuals that belong together because they share some properties. It is a subclass of the rdfs:Class.
owl:DatatypeProperty	A datatype property is one of two main categories of properties. It links individuals to data values, and is defined as an instance of the built-in OWL class owl:DatatypeProperty. An owl:DatatypeProperty is a subclass of the RDF class rdf:Property.
owl:ObjectProperty	An object property is one of two main categories of properties. It links individuals to individuals, and is defined as an instance of the built-in OWL class owl:ObjectProperty. This defines a property with the restriction that its values should be individuals. An owl:ObjectProperty is a subclass of the RDF class rdf:Property.

Table 9.2: OWL building blocks.

9.1.3 GoodRelations

The goal of GoodRelations [8] is to define a data structure for e-commerce that is suited for product entities specified on WEB. GoodRelations defines several classes that can be used to denote the products and offers on WEB pages. Two of them, `http://purl.org/goodrelations/v1#ProductOrService` and `http://purl.org/goodrelations/v1#QuantitativeValue` are described in table 9.3.

Name	Description
ProductOrService	ProductOrService is the superclass of all classes describing products or services types. Example of such subclasses is "Mobile Phone".
QuantitativeValue	Some properties can have numerical values, further defined by units of measurement. QuantitativeValue class was created for this reason.

Table 9.3: GoodRelation building blocks.

9.1.4 XML schema definition

XSD (XML Schema Definition), a recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. Its strength is the definition of datatypes - the ability to define element and attribute content as containing values such as integers and dates rather than arbitrary text [29].

boolean Boolean datatype (`http://www.w3.org/2001/XMLSchema#boolean`) is used in GoodRelation ontology for defining existence of features belonging to a specific product. Such triple would be (subject:MobilePhone predicate:wifi object:True) - saying a mobile phone has a WiFi.

9.2 Process for generating the ontology

Now, the table has been decomposed to a tree-like structure, entities and numerical values were recognized, building blocks were presented. Knowing all these parts, the tree-like structure and recognized content can be transformed to an ontology by assembling different parts of building blocks.

9.2.1 Create wrapping entity

First, a wrapping entity (the class being described by the ontology) has to be created. The entity contains the properties defined in the table. One way of defining the wrapping entity is to derive it from the contents of the table, as specified in [30] as a protagonist recognition problem. In this thesis, the wrapping entity is given as an input from the user when executing the table extraction. The statement about the wrapping entity is following.

```
:MobilePhone rdf:type owl:Class
```

9.2.2 Adding instances

Next, the entities found in the data cells are added to the ontology, as depicted in figure 9.1. These entities are added as instances of the concept specified in header cell it is being linked to.

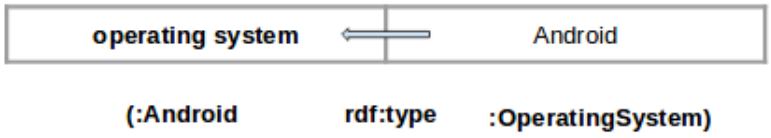


Figure 9.1: Example of statement for adding instances to the ontology.

9.2.3 Entities recognized in header cells

According to MobilePhone ontology in GoodRealation repository, meronyms (being part of - "wheel is a meronymy of automobile") are added as owl:DatatypeProperty properties with a boolean range.
However, this attitude decreases the extensibility of the ontology. Normally, the meronymy entities should be created as classes having a relationship to the wrapping entity. This way, multiple concepts could share relations to these entities.
Thus, two approaches shown in figures 9.2, 9.3 can be chosen.

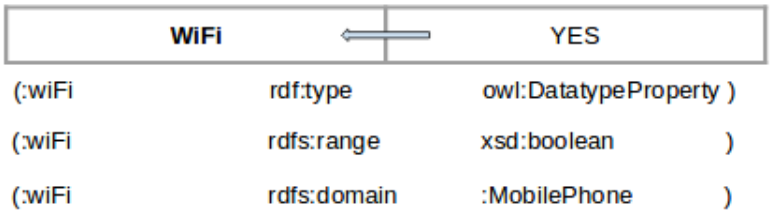


Figure 9.2: Statement according to the GoodRelation ontology.

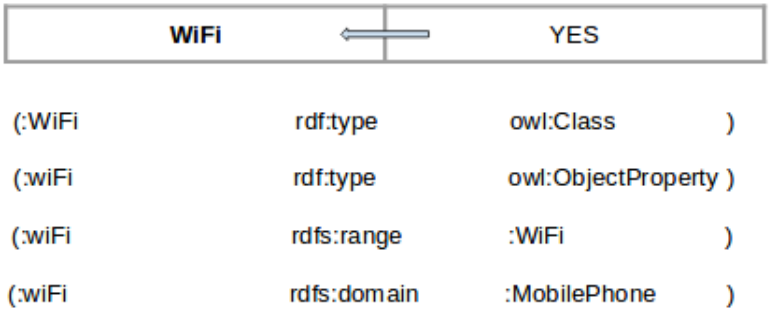


Figure 9.3: Statement enabling sharing links to entities.

9.2.4 Header properties without instance

Other header cells, not as the ones mentioned above, are analyzed in order to append them to the ontology. Such cells can refer to either Class element or Property element.

9.2.4.1 Class

The existence of instance "Blue" in data cell determines the "Colour" to be an owl:Class. The task now is to link the "Colour" class to its parent element. According to the figure 9.4, "Colour" has a parent "Appearance" taken from the tree-like structure depicting relations among table cells. Therefore, a relation has to be defined between the classes "Colour" and "Apperance". The easiest way is to create a new property "colour" declaring existance of class "Colour" belonging to class "Appearance".

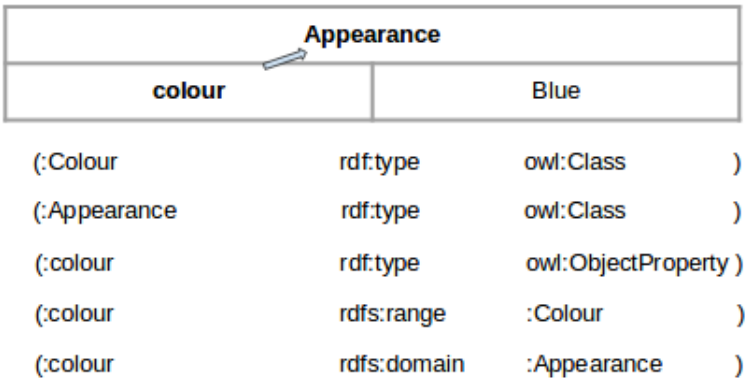


Figure 9.4: Statements linking two classes in the header hierarchy.

9.2.4.2 Property

Some header cells are linked by data cells having quantitative values (number and additionally a unit). These header cells are of type owl:ObjectProperty and their values can be gr:QuantitativeValue. Moreover, possible regular expressions and units are added as commentaries to the ontology.
Finally, these cells are linked to the wrapping entity or to their parent that must be a owl:Class in the header hierarchy, as shown in figure 9.5.

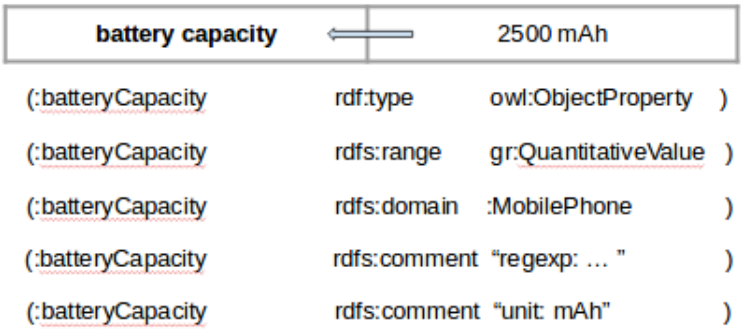


Figure 9.5: Statements capturing properties with numerical values.

9.3 Generated ontologies

The previous methods were applied to four product domains used in the testing Product dataset. Thus, 10 pages with mobile phones parameters were analyzed from each domain and corresponding ontologies were generated. These domains were amazon.com (amazon), buymobiles (buymobiles), gadgets.ndtv.com (gadgets), snapdeal.com (snapdeal).
First, the ontologies were generated according to rules where meronyms are captured by a DatatypeProperty. Their distribution of structural elements is presented in table 9.4.

Ontology / Elements	**amazon**	**buymobiles**	**gadgets**	**snapdeal**
Classes	49	29	19	51
Object properties	63	40	26	54
Datatype properties	12	4	28	26
Individuals	174	107	56	126
Levels of hierarchy	1	1	1	2

Table 9.4: Used RDF elements in individual ontologies (meronym as datatype).

Next, the ontologies were generated according to rules where meronyms are defined as individual classes, as presented in table 9.4.

Ontology / Elements	**amazon**	**buymobiles**	**gadgets**	**snapdeal**
Classes	62	34	42	71
Object properties	75	43	47	71
Individuals	170	107	55	125
Levels of hierarchy	1	1	1	2

Table 9.5: Used RDF elements in individual ontologies (meronym as class).

The levels of hierarchy parameter depicts the depth of the generated graph. In other words, number of ancestor nodes from the leaf nodes. The visualization of the graphs (capturing mainly the levels of hierarchy since individual elements are hardly readable) is presented in figures 9.6, 9.7, 9.8, 9.9. The graphs were generated by WebVOWL [31] - a Web-based Visualization of Ontologies.

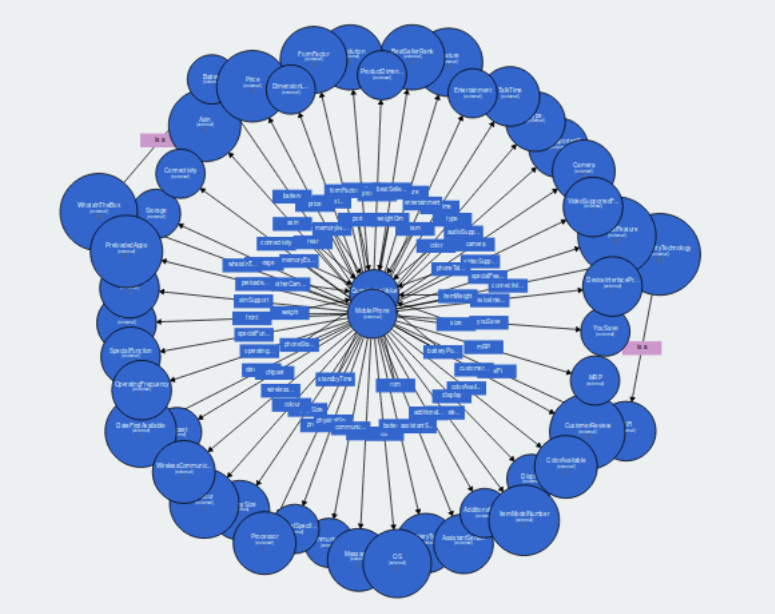


Figure 9.6: Graph of amazon ontology with meronyms as classes. The circle in the middle is MobilePhone entity, surrounded by found predicates and classes.

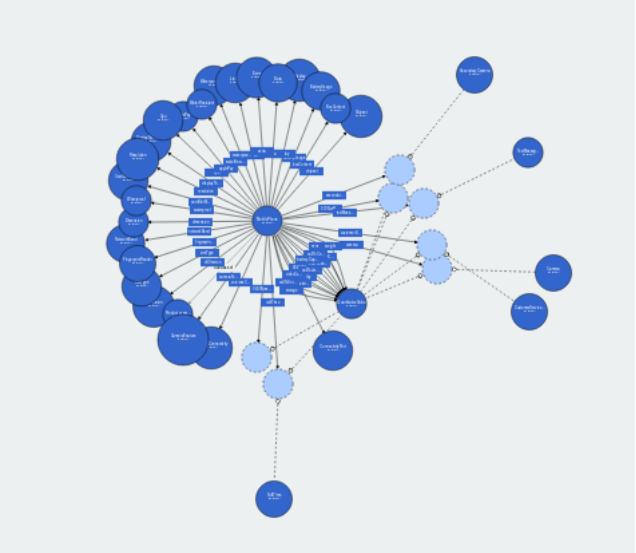


Figure 9.7: Graph of buymobiles ontology with meronyms as classes. The circle in the middle is MobilePhone entity, surrounded by found predicates and classes.

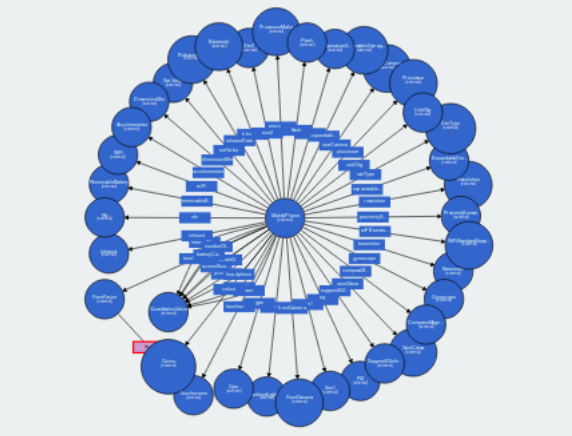


Figure 9.8: Graph of gadgets ontology with meronyms as classes. The circle in the middle is MobilePhone entity, surrounded by found predicates and classes.

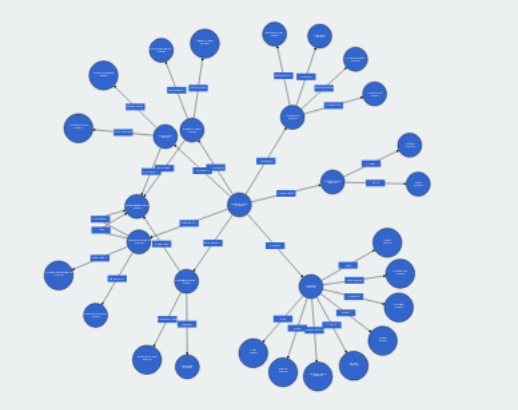


Figure 9.9: Graph of snapdeal ontology with meronyms as classes. The circle in the middle is MobilePhone entity. The first level nodes are categories of parameters such as Display, Connectivity, MemoryStorage, BatteryPower, HardwareConnectivity and General. Next level contains found classes and predicates.

Chapter 10

Comparison of created ontologies to Gold standard

An evaluation of the system has to be done from the view of the quality of learned ontologies. This evaluation would consider the lexical and structural quality of the ontology. Nevertheless, such task is very difficult and subjective since opinions on the structural quality can differ.
Yet, an evaluation method was chosen [32] in order to compare the learned ontology to a gold standard ontology and see if the best ontology can be selected according to the experimental results. Such evaluation method had to be implemented from scratch since no open-source publicly available methods were found.
After implementation, the MobilePhone ontology, a part of GoodRelation repository, was used as the gold standard.

10.1 Gold standard ontology

Generally, the gold standard ontologies are assumed to represent well and accurately the significant knowledge of the domain. However, the structure of the ontology relies only on its authors and their methodology.
An effort was made for discovery of a public ontology with a suitable taxonomy and structure. During the search, only MobilePhone ontology was found, being a part of the GoodRelation project.
On one hand, the ontology contains correct names of parameters associated with the mobile phone object. In other words, the ontology uses correct taxonomy.
On the other hand, the structure of the ontology is arguable. Mainly the use of DatatypeProperty for defining meronyms, mentioned in 9.2.3, leads to low extendability of the ontology.
Therefore two versions of this ontology were considered. The first one is the

original MobilePhone ontology. The second one is an ontology where some properties were transformed to actual classes.

10.1.1 Parts of the ontology

The ontology can be divided into several parts. The parts are Classes and their Individuals, ObjectProperties and DatatypeProperties.

Classes and Individuals There are two main classes in the ontology to be recognized - MobilePhone and MediaFormat. MobilePhone is the base class of the ontology. It is a subclass of the ProductOrService class, defined in the GoodRealtion ontology.
MediaFormat is a supplementary class with instances like JPEG, MP3, MP4, .etc.

Datatype properties There are 29 datatype properties in the ontology. These properties define meronyms of the mobile phone product - parts/features of the mobile phone. The existence of the parts are specified by boolean datatype. Example of the properties are obk:bluetooth, obk:cardSlot, obk:gps, etc.

Object properties There are 16 object properties that define properties having a numerical range. Examples of these properties are obk:cpuClockSpeed, obk:cpuCores, obk:ramSize, etc.

10.2 The distributional method for alignment (DMA)

The gold standard ontology and the learned ontology are going to be aligned in this section. The DMA method is going to be used for alignment. There are also other methods that use only string similarity matching of the concept names. However, DMA should outperform the string matching since it transforms the concepts and properties of the compared ontologies into probability distributions defining wider context of the aligned elements. In this thesis, the simple string matching is used only as a supplementary method. The DMA consists of three main steps: transformation, matching and evaluation.

10.2.1 Transformation

Each concept or a property from an ontology should be represented in some form in order to compare it to elements of the other ontology. In this method, a distributional representation is obtained.
First, a common term space is created by extracting terms that appear in

labels, name of concepts, comments, concepts instances, domain and range properties of the learned and gold standard ontology. These surrounding elements define the context of the element.
Then, each element in the ontology is represented by a term vector, as the one in figure 10.1, consisting of the terms from the common term space. Each term in the term vector is assigned a frequency number according to the occurrence count in the element's context.
Finally, the vector is normalized by the term space size. The whole process in visualized in figure 10.2.

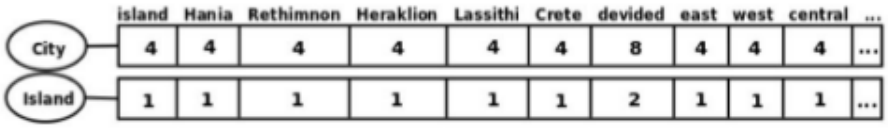


Figure 10.1: The distributional representation of a concept. Figure was borrowed from [32].

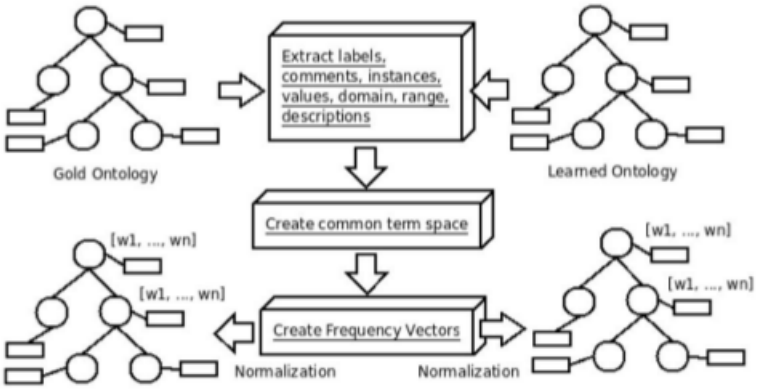


Figure 10.2: Workflow of DMA. Figure was taken from [32].

10.2.2 Matching

Each pair of elements belonging to the same category (either Concept or Property) from the learned and golden standard ontology has to be assigned a dissimilarity score (how "close" the concepts are). For this purpose, a probability metric can be used. The metric chosen in this thesis is called Total Variational Distance (TVD), defined in equation 10.1.

$$TVD = \frac{1}{2} \sum_{i} |p(i) - q(i)| \tag{10.1}$$

The p and q in the equation represent elements from the learned/gold standard ontology.
The DMA considers a one-to-one matching of the elements from the two ontologies. Therefore, the number of pairs is the minimum number of elements from either ontologies.
As a results, the final matching consists of pairs that have minimal dissimilarity distance ($SimDist_i$). In aggregation, all the pairs should minimize the value of the equation 10.2. N represents all possible pairs. M represents the final pairs.

$$argmin_N\{\sum_i^M SimDist_i\} \tag{10.2}$$

Nevertheless, sometimes multiple pairs can have the same dissimilarity score. In these cases, the pair with highest bi-gram string similarity is selected. In following equation 10.3, $s1$ is the first string, $s2$ is the second string to be compared. $pairs$ is a function returning all pairs of successive letters in the string ("FRANCE" => ["FR", "RA", "AN", "NC", "CE"]).

$$bigram(s1, s2) = \frac{2 * |pairs(s1) \bigcap pairs(s2)|}{|pairs(s1)| + |pairs(s2)|} \tag{10.3}$$

10.2.2.1 Evaluation

The correct matching is the most important step when performing evaluation and heavily affects the results. After matching, an approach has to be chosen that captures the similarity of the matching pairs both from lexical and structural view. Therefore, specific precision (equation 10.4), recall (equation 10.5) and F-measure (equation 10.6) scores were chosen that consider the lexical and structural characteristics:

$$P = \frac{1}{M}\sum_i^M (1 - SimDist_i)PCP_i \tag{10.4}$$

$$R = \frac{1}{M}\sum_i^M (1 - SimDist_i)PCR_i \tag{10.5}$$

$$F = \frac{(\beta^2 + 1)P * R}{\beta^2 R + P} \tag{10.6}$$

The precision and recall depend on Probabilistic Cotopy Precision (PCP) and Probabilistic Cotopy Recall (PCR) factors that compute the error ratio of the structure similarity of the compared ontologies.
For a pair of matching concepts C_L in the learned ontology and a concept C_G in the gold ontology, PCP_i and PCR_i (where i is the identifier of the matching pair) are defined by using the Cotopy Set of the concepts.

In detail, the Cotopy Set of a concept CS(C) is the set of all its direct and indirect super- and subconcepts and its direct properties, including the concept C itself.
Then, PCP_i is the number of concepts in the cotopy set of C_L correctly matched to concepts (correct matching is defined in alignment files that are further described at the end of this text) in the cotopy set of C_G, divided by the number of concepts in the cotopy set of C_L, as showed in equation 10.7

$$PCP_i = \frac{||CS(C_L) \bigcap CS(C_G)||}{||CS(C_L)||} \tag{10.7}$$

PCR_i is the number of concepts in the cotopy set of C_L matched to concepts in the cotopy set of C_G, divided by the number of concepts participating in the cotopy set of C_G, as showed in equation 10.8.

$$PCR_i = \frac{||CS(C_L) \bigcap CS(C_G)||}{||CS(C_G)||} \tag{10.8}$$

The correctness of the matched pairs used in PCR_i and PCP_i has to be defined in a separate alignment file. This file contains pairs of elements from different ontologies that can be substituted one for another, as viewed in figure 10.3.

```
<entity1 rdf:resource="http://purl.org/opdm/mobilephone#talkTime"/>
<entity2 rdf:resource="http://localhost/mobilephone#phoneTalkTime"/>
```

Figure 10.3: Example of an alignment.

10.3 Results of comparing ontologies

First, the DMA method described above is going to be tested on a benchmark ontology. This has to be done in verify performance of the implemented method to the original results mentioned in [32]. After ensuring about the quality, the method is going to be launched on ontologies mentioned in section 9.3.

10.3.1 Testing on benchmark ontologies

The benchmark ontology and its modifications were taken from The Ontology Alignment Evaluation Initiative (OAEI). The initiative assesses strength/weaknesses of current alignment systems, compares techniques used in the systems and helps in improving the systems.
The borrowed benchmark ontologies describe Bibliographic references and contain 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals.

Several tests were performed in order to compare the quality of the implemented algorithm to the original results. These test were:

1. Test the ontology against itself (ID=101)
2. Test the ontology against a totally irrelevant one (food ontology)(ID=102)
3. Test the ontology against its generalization in OWL Lite (some constraints are modified)(ID=103)
4. Test the ontology with its restriction in OWL Lite (unavailable constraints have been discarded)(ID=104)
5. Each label or identifier is replaced by a random one (ID=201)
6. Each label or identifier is replaced by a random one. Comments (rdfs:comment and dc:description) have been suppressed as well. (ID=202)
7. Different naming conventions (Uppercasing, underscore, dash, etc.) are used for labels. Comments have been suppressed (ID=204)
8. Labels are replaced by synonyms. Comments have been suppressed (ID=205)
9. The complete ontology is translated to French (ID=206)
10. Each label or identifiers translated to French (ID=207)
11. Combination of methods above (ID=208)
12. Combination of methods above (ID=209)
13. Combination of methods above (ID=210)

The resulting scores of benchmark tests are summarized in table 10.1.

Scores / Tasks	P	R	F-measure	Alignments	F-measure original
101	1.0	1.0	1.0	97/97 (1.0)	1.0
102	0.0	0.0	0.0	0/0 (0.0)	0.0
103	0.98	0.98	0.98	97/97 (1.0)	0.9
104	0.98	0.98	0.98	97/97 (1.0)	0.8
201	0.94	0.94	0.94	93/97 (0.96)	0.94
202	0.05	0.05	0.05	5/97 (0.05)	0.04
204	0.98	0.98	0.98	97/97 (1.0)	0.96
205	0.96	0.96	0.96	94/97 (0.97)	0.96
206	0.98	0.98	0.98	96/97 (0.99)	0.96
207	0.98	0.98	0.98	96/97 (0.99)	0.96
208	0.82	0.82	0.82	81/97 (0.84)	0.04
209	0.32	0.32	0.32	32/97 (0.33)	0.04
210	0.3	0.3	0.3	30/97 (0.31)	0.04

Table 10.1: Results of tests on benchmark tasks. P - precision, R - recall, Alignments - number of matched alignments from the alignment file, F-measure original - scores from the original paper.

The F-measure of task 101 was correctly evaluated as 1.0 since it is compared to itself. Similarly F-measure of 102 is 0.0 because it is compared to an irrelevant one.
Removing tags owl:unionOf, owl:oneOf, etc. in 103, 104 does not affect correct matching, only creates small difference in the context of elements leading to a small increase in scores.
Changing the names of the identifiers in 201 decreases the scores. However, the comments that partially form the context keep it above 90%. Removing the comments lowers the F-measure to 5% in 202.
Next, the naming conventions are changed and comments are suppressed in 204. The naming conventions should not much affect the results since language processing methods (all chars to lowercase, removing non-alphabetical and non-numerical characters, etc.) are used when creating the bag of words. Further, replacing labels by synonyms in 205 leads to confusion in pair matching (94 out of 97 pairs were correctly matched) since no semantic word similarity methods are used.
Interesting are tasks 206, 207 that have been translated to French. However, the comments have not been left out which keeps the quality of pair matching high. Also, many French words are similar to the English ones.
Mixing all the techniques above leads to general decrease in the results.
According to the results table, the implemented algorithm shows very similar results on the F-measure as the original article. That is a sign of correct implementation. Nevertheless, some steps such as text-preprocessing might have been done differently and could be the reason why some scores, for example

tasks 208-210, slightly differ.

10.3.2 Testing on product ontologies

The correctness of the implementation of the proposed method was proved above. Now, the task is to take the generated mobile phone ontologies in section 9.3 and compare them to the gold standard MobilePhone ontology.
The DMA method is going to be used for evaluation. Nevertheless, some rules have to be set before executing the tests. First of all, some test are going to consider the comment tags as a part of a context of an element and some will not. The reason is that the generated ontologies do not contain such tags.
Moreover, the ontologies can define meronyms in different ways, as mentioned in section 9.2.3. Both types of the generated ontologies are going to be assessed.
Finally, it has to be mentioned that the class MobilePhone in generated mobile phone ontologies had to be linked to a parent class ProductOrService (`http://purl.org/goodrelations/v1#ProductOrService`) in order to mirror a similar structure as the one used in gold standard ontology.

In total, four different computations were done for each ontology:

1. MER_AS_DATA - this is the ontology where meronyms were defined as datatype properties.

2. MER_AS_DATA_TRANS - this is the ontology where meronyms were defined as datatype properties. Moreover, names of correctly matched concepts in alignment file were replaced by their linked concepts from the gold standard ontology. This was done in order to eliminate differences in precision, recall and F-measure caused by correctly matched pairs that are expressed by synonyms.

3. MER_AS_CLASS - this is the ontology where meronyms were defined as classes.

4. MER_AS_CLASS_COMM - this is the ontology where meronyms were defined as classes. Moreover, it is the only task where comments of the elements were fetched into the context.

The resulting scores of ontology tests are summarized in table 10.2. The results are followed by a discussion.

Scores / Tasks	P	R	F-measure	Alignments
amazon MER_AS_DATA	0.61	0.64	0.62	13/21 (0.62)
amazon MER_AS_DATA_TRANS	0.63	0.66	**0.65**	18/21 (0.86)
amazon MER_AS_CLASS	0.58	0.65	0.61	22/33 (0.66)
amazon MER_AS_CLASS_COMM	0.56	0.64	0.60	18/33 (0.55)
buymobiles MER_AS_DATA	0.53	0.56	0.55	6/11 (0.55)
buymobiles MER_AS_DATA_TRANS	0.54	0.57	0.55	7/11 (0.64)
buymobiles MER_AS_CLASS	0.52	0.57	0.54	11/15 (0.73)
buymobiles MER_AS_CLASS_COMM	0.49	0.55	0.52	5/15 (0.33)
gadgets MER_AS_DATA	0.60	0.62	0.61	13/17 (0.76)
gadgets MER_AS_DATA_TRANS	0.61	0.62	0.62	14/17 (0.82)
gadgets MER_AS_CLASS	0.58	0.63	0.60	17/26 (0.65)
gadgets MER_AS_CLASS_COMM	0.55	0.61	0.58	13/26 (0.5)
snapdeal MER_AS_DATA	0.49	0.63	0.55	17/21 (0.76)
snapdeal MER_AS_DATA_TRANS	0.49	0.64	0.55	21/21 (1.0)
snapdeal MER_AS_CLASS	**0.48**	**0.66**	0.55	27/38 (0.71)
snapdeal MER_AS_CLASS_COMM	0.48	0.65	0.55	26/38 (0.68)

Table 10.2: Results of tests on mobile phone ontologies. P - precision, R - recall, Alignments - number of matched alignments from the alignment file (%).

Upper-bound for recall constrained by structure The recall is defined in range between 0 and 1. However, an upper-bound can be computed when considering the number of alignments. Lets consider amazon ontology. 107 matching pairs were found by the DMA, 22 out of them were defined in the alignment file (when considering meronyms as datatype properties). Thus 22 pairs would have PCR_i equal to 1 when considering the same level of hierarchy. The taxonomic dissimilarity would be 1 when considering translation. The other 85 pairs must have at least two matching ancestors ProductOrService and MobilePhone. The PCR_i would be 0.66 for these pairs since only the name of the paired elements would not match. The taxonomic similarity is omitted in this case (set to 1 since looking for upper-boundary). Thus, the upper bound for recall constrained by structure results to $(22 * 1 + 85 * 0.66)/107 = 0.73$. Clearly the recall cannot be improved, only if more alignments would be found.

Comparison of ontologies based on the meronyms definition The meronyms defined as either classes or datatype properties seem to give similar results. Some generated ontologies (like snapdeal) where more elements are defined as instances of classes can profit from the transformation of the gold standard ontology.

Levels of hierarchy The snapdeal ontology is the only one with 2 levels of hierarchy. Meaning that the leaf nodes are not linked directly to the MobilePhone entity but there is an extra node in-between. This structural difference in comparison to the gold standard ontology is visible in the results. The snapdeal ontology has a high recall when aligned to other ontologies. In contrast, the precision (decreased by the more complex structure) is the lowest among all.
Also, instances cannot be forgotten when talking about levels of hierarchy. Instances are individuals of a specific class. Therefore, the class it is being instance of is added to the sequence of its ancestors. So, the instances have +1 longer path to the root entity. It is very important since the gold standard ontology has only 4 individuals. Whereas, the generated ontologies tend to have up to 100 individuals. Therefore, some classes of the gold standard ontology are paired with instances of the learned ontology having longer ancestors path. This happened with the buymobiles ontology. The level of the hierarchy is 1 as is the level of the gold standard ontology. However, buymobiles has small number of classes but high number of instances which are paired to the classes of reference ontology.

Translation of correctly matched elements The translation was performed in order to asses how much the word similarity of correctly matched elements influences the final scores. In each ontology, at least four translations had to be performed (such as translating "phoneTalkTime" from learned ontology to "talkTime" in gold standard ontology).
Nevertheless, the improvement caused by uniting the names of parameters ranges only in 1-3%. That is understandable, for example consider a bag of words with 500 words. And one word would be correctly translated. Then, in TVD (dissimilarity measure), two differences ($|p(i) - q(i)|$) would be reduced to zero since the two words were translated. So the change would be 0.004 (2/500) and is really low.
Moreover, the generated ontologies do not contain comments. Therefore, the lexical changes can be done only on names of elements which is a very limited space.

Number of matches in alignment file Obviously, the quality of the ontology increases with higher number of correctly paired elements. Therefore, amazon and snapdeal give higher recall because there are more aligned elements than in other ontologies.

Overall size of the ontology Amazon is the largest of ontologies. The advantage of some big ontologies is a higher probability of finding a matching alignment. And the higher number of correctly pairs elements leads to better scores, as mentioned on the previous paragraph.

Which is the best ontology? Selecting the best generated ontology is arguable. First of all, the evaluation algorithm very much depends on the quality of the gold standard ontology. And the quality emerges from the way how human beings percept relations among entities and their hierarchy level in the complete entity space. Thus, it can be quite subjective.
Also, the algorithm works on one-to-one matching of elements in ontologies. Therefore, the number of pairs to be matched is the lower count of elements in both ontologies. So the final scores of the ontology are computed from: 1. correctly matched elements, 2. paired elements that have low dissimilarity values and complete the missing positions in the pairs to be matched. So obviously, the algorithm does not encounter the amount of uncertain elements that do not fit in the pairs to be matched. This way, an ontology with many alignments but lots of uncertain (like amazon) could be selected as the best ontology.
Thirdly, the DMA does not capture the semantic similarity of words. Instead, it uses context of the elements. That would be fine if the ontologies would be well commented. Nevertheless, that is not the case of automatically generated ontologies.

Conclusion

Building an ontology learning system is a non-trivial task. However, it is an essential building block in case of mining and understanding knowledge from different data sources. This semantically understandable knowledge can be used for building knowledge graphs that are nowadays the key part of intelligent applications. Such applications understand customers needs, especially chatbots, question-answering systems and recommendation systems.

More specifically, this thesis brought an insight into building ontologies from tabular data in HTML documents. These HTML documents are the input to the implemented ontology learning system. First, the documents are searched for relevant tables which are detected by a machine learning classifier. Such classifier receives table represented by features, computed from the internal matrix representation of the WEB table, and predicts the table type as ENTITY, RELATION, LAYOUT, etc. Nevertheless, the implemented system specializes only on building ontologies from ENTITY tables. Therefore, the classifier was applied to 11 sites with ENTITY tables and correctly predicted table types in 7 of them.

After, the table cells had to be labeled as data/header cells. The labeling had to be done to correctly understand relations in the table. Therefore, a machine learning method was trained for classifying table cells. The header classification correctly labeled 86% of various tables found in sites with mobile phones. An excellent precision is necessary, because it was found out that table understanding heavily depends on correctly labeled structure.

Furthermore, several methods were presented that transform the labeled table into a tree graph. The graph, together with trivial named entity recognition, is an input for ontology generation algorithm. Such algorithm constructs the ontology by applying several predefined rules that assemble structural blocks, borrowed from standards, such as RDF, OWL, etc.

Finally, ontologies were generated for four mobile phone sites such as amazon, snapdeal, gadgets and buymobiles. These four ontologies were compared to the gold standard MobilePhone ontology by a chosen alignment method

(DMA) that captures the taxonomic and structural similarity. As a result, amazon ontology was selected as the most similar to the reference ontology with its 65% F-measure score. Nevertheless, the results heavily depend on the quality of the gold standard ontology, as well as on the used alignment method. Especially, the quality of both gold standard ontology and generated ontologies is very subjective. Thus, more approaches for building ontologies are mentioned in the thesis, because it is always up to the domain expert to decide which ontology is more suitable.

As a result, the proposed system can be applied for generating fuzzy ontologies for different domains like products, companies, basketball players, etc. These ontologies can serve as a draft for domain ontology expert that creates the final ontology. In such case, an automatic approach for merging the fuzzy ontologies to a less fuzzy ontology is suggested as a future step. The merged ontology would consist of elements shared among the input ontologies. Thus, it would filter out elements with uncertain names and values.

Another possible application can be found in conceptual extraction. The generated ontologies can serve as extraction concepts whose taxonomy is used for key-value pairs extraction in given input documents. Then, database population can be performed from these key-value pairs according to the ontology structure.

Bibliography

[1] Gruber, T. R.; et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, volume 5, no. 2, 1993: pp. 199–220.

[2] Medelyan, O.; Witten, I. H.; et al. Automatic construction of lexicons, taxonomies, ontologies, and other knowledge structures. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, volume 3, no. 4, 2013: pp. 257–279.

[3] Miller, G. A. WordNet: a lexical database for English. *Communications of the ACM*, volume 38, no. 11, 1995: pp. 39–41.

[4] Bollacker, K.; Evans, C.; et al. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, AcM, 2008, pp. 1247–1250.

[5] Auer, S.; Bizer, C.; et al. Dbpedia: A nucleus for a web of open data. In *The semantic web*, Springer, 2007, pp. 722–735.

[6] Suchanek, F. M.; Kasneci, G.; et al. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, ACM, 2007, pp. 697–706.

[7] Semantic Web refers to W3C's vision of the Web of linked data. `https://www.w3.org/standards/semanticweb/`, accessed: 2017-03-05.

[8] Hepp, M. GoodRelations is a vocabulary for publishing the details of products and services. `http://www.heppnetz.de/projects/goodrelations/`, accessed: 2017-03-05.

[9] Ronallo, J. HTML5 Microdata and Schema. org. *Code4Lib Journal*, volume 16, 2012.

[10] Wang, S.; Zeng, Y.; et al. Ontology extraction and integration from semi-structured data. In *International Conference on Active Media Technology*, Springer, 2011, pp. 39–48.

[11] Lynn, S.; Embley, D. W. Automatic Generation of Ontologies from Canonicalized Web Tables. *submitted manuscript*, 2008.

[12] Tijerino, Y. A.; Embley, D. W.; et al. Towards Ontology Generation from Tables. *World Wide Web*, volume 8, no. 3, 2005: pp. 261–285.

[13] Wang, X. Tabular abstraction, editing, and formatting. 1996.

[14] Pivk, A.; Cimiano, P.; et al. Transforming arbitrary tables into logical form with TARTAR. *Data & Knowledge Engineering*, volume 60, no. 3, 2007: pp. 567–595.

[15] Kifer, M.; Lausen, G. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In *ACM SIGMOD Record*, volume 18, ACM, 1989, pp. 134–146.

[16] Group, R. W. RDF is a standard model for data interchange on the Web. `https://www.w3.org/RDF/`.

[17] Group, O. W. The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. `https://www.w3.org/OWL/`.

[18] Tanaka, M.; Ishida, T. Ontology extraction from tables on the web. In *Applications and the Internet, 2006. SAINT 2006. International Symposium on*, IEEE, 2006, pp. 7–pp.

[19] Nagy, G.; Seth, S.; et al. Data extraction from web tables: The devil is in the details. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, IEEE, 2011, pp. 242–246.

[20] Pasupat, P.; Liang, P. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*, 2015.

[21] Cha, S.-i.; Ma, Z.-m.; et al. Learning of ontology from the web-table. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 3, IEEE, 2011, pp. 1454–1458.

[22] Nederstigt, L. J.; Aanen, S. S.; et al. FLOPPIES: a framework for large-scale ontology population of product information from tabular data in e-commerce stores. *Decision Support Systems*, volume 59, 2014: pp. 296–311.

[23] Wang, Y.; Hu, J. Detecting tables in html documents. In *International Workshop on Document Analysis Systems*, Springer, 2002, pp. 249–260.

[24] Eberius, J.; Braunschweig, K.; et al. Building the dresden web table corpus: A classification approach. In *Big Data Computing (BDC), 2015 IEEE/ACM 2nd International Symposium on*, IEEE, 2015, pp. 41–50.

[25] Lehmberg, O.; Ritze, D.; et al. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, International World Wide Web Conferences Steering Committee, 2016, pp. 75–76.

[26] Fang, J.; Mitra, P.; et al. Table Header Detection and Classification. In *AAAI*, 2012, pp. 599–605.

[27] CiteSeerx is an evolving scientific literature digital library and search engine. `http://csxstatic.ist.psu.edu/about`, accessed: 2017-03-28.

[28] Black, P. E. Ratcliff/Obershelp pattern recognition. *Dictionary of Algorithms and Data Structures*, volume 17, 2004.

[29] Group, X. S. W. XML Schemas express shared vocabularies and allow machines to carry out rules made by people. `https://www.w3.org/XML/Schema`.

[30] Crestan, E.; Pantel, P. Web-scale knowledge extraction from semi-structured tables. In *Proceedings of the 19th international conference on World wide web*, ACM, 2010, pp. 1081–1082.

[31] Lohmann, S.; Negru, S.; et al. Visualizing Ontologies with VOWL. *Semantic Web*, volume 7, no. 4, 2016: pp. 399–419, doi:10.3233/SW-150200. Available from: `http://dx.doi.org/10.3233/SW-150200`

[32] Zavitsanos, E.; Paliouras, G.; et al. Gold standard evaluation of ontology learning methods through ontology transformation and alignment. *IEEE Transactions on Knowledge and Data Engineering*, volume 23, no. 11, 2011: pp. 1635–1648.

Appendix A

Contents of CD

Attached CD contains datasets, generated ontologies, alignment files, python scripts and other files needed for completing the work.

- root
  - MachineLearning
    - ProductDataset - contains HTML tables and computed features for different product domains
    - WDC Dataset - contains HTML tables and computed features for tables from WDC dataset
    - FeaturesGenerator.py - a script for downloading HTML tables and generating features
    - HeaderDetection.ipynb - a script for training and testing header detection classifier
    - LayoutDetection.ipynb - a script for training and testing layout detection classifier
    - sample.gz - file containing information about WDC dataset
    - header_detection.plk - trained model
    - layout_detection.plk - trained model
  - OntologyLearning - contains scripts and programs used for building the ontologies
  - ComparingOntology
    - OntologiesAndAlignments - contains built ontologies, gold standard ontology and alignment files
    - ScoringOntologyAgainstGoldStandard.ipynb - script for scoring learned ontologies against final ontologies
  - Thesis - sources of the text