

Department of Computer Science, Systems and Communication

PhD program Computer Sciences

Cycle XXXII

# **ENABLING TABULAR DATA UNDERSTANDING BY HUMANS AND MACHINES THROUGH SEMANTIC INTERPRETATION**

Cremaschi Marco

052538

Tutor: Prof. Andrea Maurino

Supervisor: Prof. Flavio Maria De Paoli

Coordinator: Prof. Leonardo Mariani

**2018/2019**



## Abstract

A significant number of documents, reports and Web pages –an analysis reports 233M relational tables within the Common Crawl repository of 1.81 billion documents– makes use of tables to convey information that cannot be easily processed by humans, and understood by computers. To address this issue, we propose a new approach that allows computers to interpret the semantics of a table, and provides humans with a more accessible representation of the data contained in a table. To achieve the objective, the general problem has been broken down into three sub-problems: (i) define a method to provide a semantic interpretation of table data; (ii) define a descriptive model that allows computers to understand, access and share table data via Web services; and (iii) define processes, techniques and algorithms to generate natural language representation of the table data.

Regarding sub-problem (i), the semantic representation of a data has been obtained through the application of table interpretation techniques, which supports users to identify in a semi-automatic way the meaning of the data in the table and the relationships between them. Such techniques take a table and a Knowledge Graph (KG) as input, and deliver as output an RDF representation –a set of tuples <subject, predicate, object>–. The output contains the input table annotated with the KG concepts and properties. This thesis presents a new approach, rooted in the existing literature, to laid the foundations for the development of a new tool -called MantisTable- which automatically performs a complete semantic interpretation of a table. The conducted experiments have shown good results compared to similar techniques.

Sub-problem (ii) has been addressed by exploiting the results of semantic table interpretation techniques to enhance a popular description format and allow automatic retrieval and processing of table data. The delivery is a new kind of description format that combines the OpenAPI specification with JSON-LD.

Sub-problem (iii) has been addressed by defining a natural language generation technique that uses a neural network to translate Resource Description Format (RDF) data obtained from table interpretation into sentences. Thanks to these sentences, it is possible to create a textual representation of the content of the table, and possibly extend it with additional information from data sources that can be selected automatically using semantic annotations.



# Table of contents

<b>List of figures</b>	<b>ix</b>
<b>List of tables</b>	<b>xiii</b>
<b>List of acronyms</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The Semantic Interpretation of Tables and its Use . . . . .	2
1.2 Research Questions . . . . .	5
1.3 Contribution . . . . .	5
1.4 Thesis Structure . . . . .	8
<b>2 Preliminaries</b>	<b>11</b>
2.1 The Semantic Web and Resource Description Format . . . . .	12
2.1.1 Linked Data . . . . .	13
2.1.2 Resource Description Format . . . . .	15
2.1.3 Resource Description Format Schema . . . . .	19
2.1.4 Web Ontology Language . . . . .	20
2.1.5 Knowledge Graph . . . . .	21
2.2 Tables . . . . .	23
2.3 Application Programming Interface . . . . .	25
2.4 Neural Networks . . . . .	27
<b>I Semantic Table Interpretation</b>	<b>31</b>
<b>3 State-of-the-art</b>	<b>33</b>
3.1 Application Scenarios in Semantic Table Interpretation . . . . .	37
3.2 Approaches Supporting Semantic Table Interpretation . . . . .	38

3.3	Comparison of Tools . . . . .	42
3.3.1	DataGraft . . . . .	42
3.3.2	Karma . . . . .	44
3.3.3	Odalic . . . . .	46
3.3.4	OpenRefine . . . . .	49
3.3.5	STAN . . . . .	51
3.3.6	TableMiner+ . . . . .	53
3.3.7	Final comparison . . . . .	56
<b>4</b>	<b>New Semantic Table Interpretation Approach</b>	<b>59</b>
4.1	MantisTable approach . . . . .	59
4.1.1	Data Preparation . . . . .	60
4.1.2	Column Analysis . . . . .	61
4.1.3	Concept and Datatype Annotation . . . . .	63
4.1.4	Predicate Annotation . . . . .	67
4.1.5	Entity Linking . . . . .	74
4.2	Formalisation . . . . .	75
<b>5</b>	<b>MantisTable Tool</b>	<b>79</b>
5.1	Architecture and Interface . . . . .	79
5.2	Validation . . . . .	83
5.2.1	Datasets . . . . .	83
5.2.2	Evaluation Measures . . . . .	84
5.2.3	Experimental Settings . . . . .	85
5.2.4	Evaluation results and Discussion . . . . .	87
5.2.5	Challenge Evaluation Results . . . . .	89
5.2.6	STILT tool . . . . .	90
<b>II</b>	<b>API Composition using Semantic Annotations</b>	<b>91</b>
<b>6</b>	<b>State-of-the-art</b>	<b>93</b>
6.1	Approaches Supporting Service Descriptions and Composition . . . . .	96
6.1.1	Service Descriptions . . . . .	96
6.1.2	Service Composition . . . . .	98

---

<b>7</b>	<b>New API Description Model</b>	<b>101</b>
7.1	AutomAPIC Approach . . . . .	101
7.1.1	Semantic Description Creation . . . . .	101
7.1.2	Composition Types and Rules Analysis . . . . .	103
<b>8</b>	<b>AutomAPIC Tool</b>	<b>109</b>
8.1	Architecture and Process . . . . .	109
8.1.1	Getting OpenAPI Descriptions . . . . .	110
8.1.2	Adding Semantic Annotation . . . . .	110
8.1.3	Performing Automatic Composition . . . . .	113
8.2	Validation . . . . .	114
<b>III</b>	<b>Natural Language Generation with Neural Networks</b>	<b>117</b>
<b>9</b>	<b>State-of-the-art</b>	<b>119</b>
9.1	Sub-tasks of Natural Language Generation . . . . .	121
9.2	Application Scenarios in Natural Language Generation . . . . .	126
9.3	Training Dataset . . . . .	127
<b>10</b>	<b>New Pipeline for triple-text alignment</b>	<b>133</b>
10.1	SeAlion approach . . . . .	133
10.1.1	Data Preparation . . . . .	134
10.1.2	Entity Linking . . . . .	135
10.1.3	Literal Extraction . . . . .	135
10.1.4	Relation Extraction . . . . .	136
<b>11</b>	<b>SeAlion and GazelLex Tools</b>	<b>143</b>
11.1	SeAlion Architecture and Interface . . . . .	143
11.2	GazzelLex Tool . . . . .	144
11.3	Current Limitations and Lessons Learned . . . . .	149
<b>IV</b>	<b>Natural Language Generation from Tabular Data</b>	<b>151</b>
<b>12</b>	<b>Semantic Table Interpretation - Human Side</b>	<b>153</b>
12.1	Datasets . . . . .	154
12.2	An Exploratory Research through Eye Tracking . . . . .	160
12.2.1	Option I: Experts (E) – Non Experts (NE) Comparison . . . . .	161

12.2.2 Option II: Task Oriented (TO) – Not Task Oriented (NTO) Comparison	163
12.3 Corpus Analysis . . . . .	165
<b>13 MantisTablex Tool</b>	<b>167</b>
13.1 Process . . . . .	167
13.2 Architecture and Interface . . . . .	170
<b>V Conclusion and Future Directions</b>	<b>173</b>
<b>14 Conclusions</b>	<b>175</b>
14.1 Future Directions . . . . .	176
<b>References</b>	<b>179</b>



# List of figures

1.1	An example of a table. . . . .	3
1.2	An example of an annotated table. . . . .	3
1.3	An example of a table with no contextual information. . . . .	4
2.1	Linked Open Data Cloud showing links between datasets. lod-cloud.net. . .	14
2.2	Example of RDF Graph. . . . .	16
2.3	Example of RDF/XML syntax. . . . .	17
2.4	Example of N-Triples syntax. . . . .	17
2.5	Example of Turtle syntax. . . . .	18
2.6	Example of JSON-LD syntax. . . . .	18
2.7	Example of Knowledge Graph. . . . .	22
2.8	Structure of a table. . . . .	23
2.9	Example of a horizontal table. . . . .	23
2.10	Example of a vertical table. . . . .	23
2.11	Example of a two-dimensional table. . . . .	23
2.12	Example of a complex table. . . . .	24
2.13	Example of relational table. . . . .	24
2.14	A simple example of NNs architecture. . . . .	27
3.1	Example of a well-formed relational table, with labels that are used in this Section. . . . .	34
3.2	A sample of Knowledge Graph (KG). . . . .	34
3.3	Example of a table's semantic annotation. . . . .	34
3.4	DataGraft: interface devoted to operations on tables. . . . .	43
3.5	DataGraft: columns' annotation. . . . .	43
3.6	Karma: example of an empty model. . . . .	44
3.7	Karma: class annotation. . . . .	45
3.8	Karma: example of relation. . . . .	45

3.9	Odalic: classification e disambiguation's results summary. . . . .	47
3.10	Odalic: cell disambiguation. . . . .	48
3.11	Odalic: graph that represents the relations between columns. . . . .	48
3.12	OpenRefine: reconciliation. . . . .	49
3.13	OpenRefine: additional settings to improve reconciliation. . . . .	50
3.14	OpenRefine: the result of reconciliation. . . . .	51
3.15	STAN: the upload of a new table. . . . .	52
3.16	STAN: the selection of the subject column's annotation. . . . .	52
3.17	STAN: column annotation. . . . .	52
3.18	TableMiner+: adding an URL and the selection of a specific table. . . . .	53
3.19	TableMiner+: interactive annotated table. . . . .	54
3.20	TableMiner+: the edit of a cell's associated entity. . . . .	54
3.21	TableMiner+: graph of concepts and relations. . . . .	55
4.1	Concept annotation workflow. . . . .	67
4.2	The example table with Predicate Annotations. . . . .	73
4.3	ABSTAT profiling tool. . . . .	74
5.1	Architecture of MantisTable tool. . . . .	80
5.2	MantisTable interface overview: Visualization Mode (1. process, 2. console), Info Mode (3. right side bar), Edit Mode (4. edit form). . . . .	80
5.3	List of loaded tables on MantisTable main page. . . . .	81
5.4	Final step of MantisTable STI process. . . . .	82
5.5	Precision of the approach when the $k$ value changes. . . . .	86
5.6	Precision and Recall of the approach when the $\bar{\gamma}$ value changes. . . . .	86
5.7	Precision of the approach when the $\bar{\delta}$ and $\bar{\beta}$ value change. . . . .	87
7.1	Schema of sequence composition. . . . .	103
7.2	Sequence composition: examples of the four compatibility cases. . . . .	105
7.3	Example of a process of composition of the use case. . . . .	106
7.4	Schema of merge composition. . . . .	107
8.1	Architecture of AutomAPIC tool. . . . .	109
8.2	List of the possible compositions. . . . .	114
9.1	Examples of queries in TAC KBP. . . . .	129
9.2	An example of relation generation in FB15-237. . . . .	130
9.3	The pipeline of T-Rex [33]. . . . .	131

---

10.1	Results of Entity Linking in SeAlion. . . . .	136
10.2	Results of Literal Extraction in SeAlion. . . . .	137
10.3	An output from SeAlion’s current predicate aligner. . . . .	138
10.4	An example of how dependency parsing could help relation extraction. . . . .	139
10.5	The Frame Elements and Their Syntactic Realisations for known.a in FrameNet. 140	
10.6	Dependency parsing for the sentence “Julia, daughter of Julius Caesar, was born on 76 B.C.”. . . . .	141
11.1	Architecture of SeAlion tool. . . . .	144
11.2	The workflow of our model. . . . .	145
11.3	Page for creating and editing templates related to Text Structuring. . . . .	146
11.4	Page for the revision of the lexicalized triples. . . . .	147
11.5	Lexicalization of triples from the Juventus-Chivevo football match. . . . .	148
12.1	“Crypto-currencies” tables with some related information. . . . .	155
12.2	“Fruit” table that show fruit’s nutritional values. . . . .	156
12.3	Excerpt of the "Food that contains water" table, concerning data about the amount of water present in some groceries. . . . .	157
12.4	“America’s Top 50 Women In Tech” table, containing a ranking about women that work in technology fields. . . . .	158
12.5	Table that contains a list of apps available on Google Play Store. . . . .	159
12.6	“Mountains” table that contains a list of mountains. . . . .	160
12.7	Eyetracking screen comparing NE(left)/E(right) about Table 12.4. . . . .	162
12.8	Eyetracking screen comparing NE(left) / E(right) about Table 12.3. . . . .	162
12.9	Eyetracking screen comparing NE(left) / E(right) about Table 12.6. . . . .	162
12.10	The comparison of Eyetracking Frame between NTO (left) and TO (right) on the Table 12.1. . . . .	164
12.11	The comparison of Eyetracking Frame between NTO (left) and TO (right) on the Table 12.2. . . . .	164
12.12	The comparison of Eyetracking Frame between NTO (left) and TO (right) on the 12.5. . . . .	164
13.1	MantisTablex input example. . . . .	167
13.2	Architecture of MantisTablex tool. . . . .	171
13.3	MantisTablex interface: triples selection and summary preview for single-row selection mode. . . . .	172
13.4	MantisTablex interface: triples selection and summary preview for multi-row selection row, low familiarity. . . . .	172



# List of tables

3.1	Approaches supporting Semantic Table Interpretation (STI). . . . .	38
3.2	Tool comparison. . . . .	57
4.1	The table reports a list of the highest peaks in the world from T2D Gold Standard, extended with other columns to consider a variety of situations (grey columns). . . . .	60
4.2	Table 4.1 after the Data Preparation phase. . . . .	61
4.3	Features for Subject Column Detection. . . . .	62
4.4	Values of the features of the S-column detection for the mountain table. . .	62
4.5	The example table after Column Analysis phase. . . . .	63
4.6	Class frequencies for each extracted PEAK entity. . . . .	66
4.7	Global frequency values for the PEAK column. . . . .	66
4.8	Datatype and Regextype mapping. . . . .	68
4.9	Table 4.1 with column annotations. . . . .	68
4.10	Types of context for a table inside Predicate Annotation. . . . .	68
5.1	Characteristics of the Gold Standards. . . . .	84
5.2	Results of the Subject Column Detection step. . . . .	88
5.3	Problem on Subject Column Detection step. . . . .	88
5.4	Results of the Concept Annotation. . . . .	88
5.5	Results of the Predicate Annotation for NE-column. . . . .	89
5.6	Results of the Predicate Annotation for L-column. . . . .	89
5.7	Results of the Challenge Tasks. . . . .	89
6.1	Comparison of API description standards. . . . .	97
6.2	Comparison of API description models. . . . .	99
8.1	Validation dataset. . . . .	114
8.2	Confusion matrix. . . . .	115

9.1	Comparisons between different training datasets for Natural Language Generation (NLG). . . . .	128
9.2	An example of instance of Wikireadings. . . . .	130

# List of acronyms

**API** Application Programming Interface

**bow** bag-of-words

**BKG** Background Knowledge Graph

**EL** Entity Linking

**ER** Entity Recognition

**HTML** Hypertext Markup Language

**IoT** Internet of Things

**LOD** Linked Open Data

**LSTM** Long Short-Term Memory

**LM** Language Model

**LMs** Language Models

**KG** Knowledge Graph

**KGs** Knowledge Graphs

**ML** Machine Learning

**NEL** Named Entity Linking

**NLG** Natural Language Generation

**NLP** Natural Language Processing

**NMT** Neural Machine Translation

**NN** Neural Network

**NNs** Neural Networks

**OWL** Web Ontology Language

**RCP** Remote Procedure Calls

**RDF** Resource Description Format

**RDFS** Resource Description Format Schema

**REG** Referring Expression Generation

**REST** REpresentational State Transfer

**RNN** Recurrent Neural Network

**RNNs** Recurrent Neural Networks

**SOAP** Simple Object Access Protocol

**SRNs** Simple Recurrent Networks

**STI** Semantic Table Interpretation

**URI** Uniform Resource Identifier

**XML** eXtensible Markup Language



# Chapter 1

## Introduction

In today's society, the value of information is considerably higher than any other sort of contribution. With the term "information" we mean the formula that binds a certain amount of data to a specific context. The latter is crucial to allow the data itself to describe many different scenarios, according to the social context, moment and place when they are read. Digitisation has caused an explosion of data in every sector of our society that generates a huge amount of information, basically useless on its own because it is disorganised and complex.

This huge amount of data, commonly defined as Big Data, can be categorised as structured (tables) or non-structured (text) and can originate not only from human beings, but also from technology (think about sensors and other tools of data acquisition).

One of the enabling technologies that allowed an ever-increasing production of data was created in the 90', when a British physicist that worked at CERN, in Geneva, Tim Berners-Lee, had an intuition: make knowledge universal by creating a system that could host a huge amount of information, mutually connected to grant a flexible and open exchange. This led to the origin of the *World Wide Web*, a project that brought Internet into our homes.

The Web as it was conceived would contain resources mainly designed and produced for human consumption, rather than machines. As a matter of facts, reading and understanding a web page is very simple for a user, but it is not so easy for computers, which are able to distinguish different *markup* elements that define a page but cannot comprehend its meaning or how those elements are interrelated. The attempt to correct this weakness led to the creation of *Semantic Web* [8]. The idea behind this is precisely to transform web *documents* into *data* to which we assign a unique and well-defined meaning, adding information on how they can be linked together.

A huge source of data are the tables embedded into Web documents. When we happen to consult a table, some cognitive processes that get triggered inside our brains allow us

to give table values –expressed in form of free text– a meaning (or a semantic), that can be more or less accurate depending on several factors, like our familiarity with the domain of the exposed data, the background knowledge, or the presence of contextual information. The free-text content of tables cannot be understood by machine if it does not refer to a Knowledge Graph (KG) that assigns well-defined meaning to terms and support the creation of links between them –the real added value of the Semantic Web. Therefore, the issue is to find the right association between values in tables and concepts/properties of a KG that describes real-world entities and their interrelations, organised in a graph [74]. The attempt to solve this problem led to the birth of Semantic Table Interpretation (STI) techniques, which are processes through which, given a table and a KG as input, it is possible to interpret the structure and the semantic of the table by associating its content to semantic concepts taken from the KG.

The semantic interpretation of tables has different application fields of great importance, such as: (i) Data search, which is the option of making the relational data contained within tables accessible through human searches; (ii) Data enrichment, which is the option of completing and extend the table’s content with other data and therefore with additional information that come from other sources; and (iii) KG construction/KG population, meaning the option of building or enrich a KG thanks to the semantic information found by the annotation process.

Moreover, there is an emerging application of STI, which is the translation of tables into natural-language sentences to help users understand the table content even if they are not familiar with the KG terminology, or use devices that cannot fully display tables (e.g. smartphones with audio interaction capability via chatbots).

In this thesis we address the Semantic Table Interpretation, the API composition using Semantic Annotations and Natural Language Generation of tabular data.

## 1.1 The Semantic Interpretation of Tables and its Use

To provide an overview of STI and how it is exploited to describe resources and generate natural-language sentences, let’s consider an example.

In Figure 1.1 is shown a table that contain the description of some mountains:

A human reader can reach the conclusion that the table’s domain is mountains. Other users will also understand that for each listed mountain in the first column, the position, the height and the chain to which it belongs are in the next columns.

On the contrary, a computer will consider the information inside the table as a collection of strings, not really understanding which real entities they represent, which relations exist

Name	Coordinates	Height	Range
Mont Blanc	45°49'57"N 06°51'52"E	4808	Mont Blanc massif
Lyskamm	45°55'20"N 07°50'08"E	4527	Pennine Alps
Monte Cervino	45°58'35"N 07°39'31"E	4478	Pennine Alps

Figure 1.1 An example of a table.

between them, etc. So it is necessary to identify a method to annotate strings, so that machines can “understand” and then manage, integrate, and process them. STI techniques can add semantic annotations that associates additional information to say, for example, that Mont Blanc is of type Mountain, and that it is in relation with “4808”, which is of type integer, via the property elevation. (Figure 1.2).

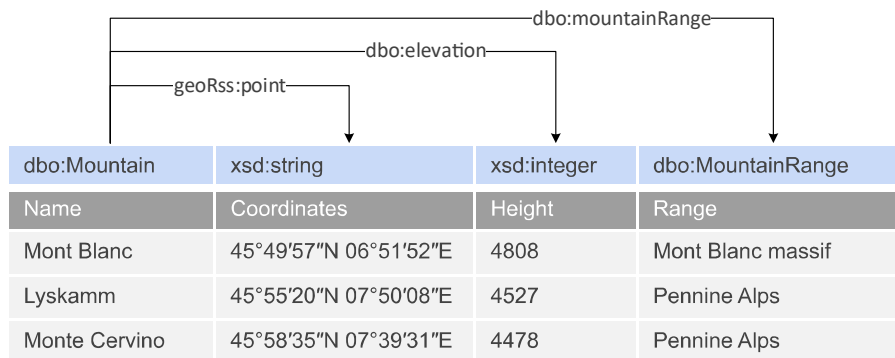


Figure 1.2 An example of an annotated table.

Moreover, semantic annotations allow machines to “understand” that the string Mont Blanc is not only a mountain, but also that it is located between the regions of Aosta Valley in Italy, and Savoie and Haute-Savoie in France. This kind of contextualisation and expansion –named extension– can be achieved by integrating information from different sources, in this case Wikipedia<sup>1</sup>.

```

1 <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4   xmlns:dbo="http://dbpedia.org/ontology/" >
5
6   <rdf:Description rdf:about="http://dbpedia.org/resource/Mont_Blanc">
7     <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
8     <dbo:firstAscendYear rdf:datatype="http://www.georss.org/georss/point">
9       XXX
10    </dbo:firstAscendYear>
11    <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
12      4804
13    </dbo:elevation>
14    <dbo:mountainRange rdf:resource="http://dbpedia.org/./Mont_Blanc_massif" />

```

<sup>1</sup>[https://en.wikipedia.org/wiki/Mont\\_Blanc](https://en.wikipedia.org/wiki/Mont_Blanc)

```

15     <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
16         Mont Blanc
17     </rdfs:label>
18     <dbo:locatedInArea rdf:resource="http://dbpedia.org/resource/Aosta_Valley" />
19     <dbo:locatedInArea rdf:resource="http://dbpedia.org/resource/Italy" />
20     <dbo:locatedInArea rdf:resource="http://dbpedia.org/resource/Haute-Savoie" />
21     <dbo:locatedInArea rdf:resource="http://dbpedia.org/resource/France" />
22 </rdf:Description>
23
24 </rdf:RDF>

```

Listing 1.1 An example of a STI output.

An example of the RDF output of an STI process can be read in Listing 1.1. Such descriptions are not so easy to read for humans, in particular if they are not familiar with semantic web technologies. Therefore, it may be convenient to apply NLG techniques to convert RDF into text – a process named lexicalization – as shown in Listing 1.2.

```

1 Mont Blanc is a mountain which rises 4,808 m above sea level.
2 It belongs to the Mont Blanc massif mountain range and it is located between
3 Aosta Valley and Haute-Savoie ...

```

Listing 1.2 The example of Listing 1.1 rendered in text form.

The example about mountains dealt with a table with enough contextual information to let a human reader understand the content of the table. Let's now consider another example with much less contextual information like the one in Figure 1.3.

Blacephalon	White	1.8 m	13	-
Illumise	Purple	0.6 m	17.7	-
Reuniclus	Green	1.0 m	20.1	Duosion
Gligor	Purple	2.0 m	42.5	Gligar

Figure 1.3 An example of a table with no contextual information.

The lack of a header row, and the use of fancy names in the first column make it very difficult to define what the table is about.

After applying the STI and NLG techniques discussed in this thesis, the content of the first row in the table can be translated into the sentences shown in Listing 1.3. Thanks to the semantic annotations, we can contextualise the data within the table, and so we can reach the conclusion that even if at first glance it might seem a list of medicines or plants, in reality this is a list of Pokémons<sup>2</sup>.

```

1 Blacephalon is a Fire Ghost-type Pokemon introduced in Generation VII.
2 It is mainly white in color and has a height of 1.8 m.
3 It weighs is 13 kg.

```

Listing 1.3 The example of Figure 1.3 rendered in text form.

<sup>2</sup>[www.wikidata.org/wiki/Q41791949](http://www.wikidata.org/wiki/Q41791949)

## 1.2 Research Questions

In this thesis, we mainly focus on the implementation of a semantic table interpretation approach, with the ultimate goal of making the data within the table more accessible, both to machines and humans. This leads to the following research questions:

- **Question 1:** What does it mean to annotate semantically a table? What are the table's elements that must be taken into consideration?
- **Question 2:** How can elements within a table be made unambiguous? How can the context to support disambiguation be created?
- **Question 3:** What enables the presence of semantic annotations with respect to the processing of data by a machine?
- **Question 4:** What make data more accessible for a user? Is it possible to create a representation of the tabular data in natural language? What is the most relevant information inside the table for the user?
- **Question 5:** What are the techniques for converting a RDF triple into natural language?

## 1.3 Contribution

The contributions of this thesis can be grouped in three main chapters:

1. A fully automated approach to a complete STI (Part I);
2. A practical approach to services composition by exploiting light semantic annotations (Part II);
3. A pipeline for RFD lexicalization (Part III).

Most of the the state-of-the-art approaches focus on individual aspects of the STI like, for example, the analysis of columns that contains entities or literal. Only recently few works attempting a comprehensive approach have been published. In particular, [123] proposed a very interesting and promising breakdown of the STI problem. We took inspiration from it to revise our work and deliver the novel, comprehensive approach that is described in the next chapter. In summary, our approach creates contexts using elements of the input table, i.e. headers of columns and cells in the same row in order to disambiguate the text elements within a cell [21, 22, 20]. More specifically, the approach calculates the similarity between

the representation (union of contexts) of the element in the table and the representation of the candidate entities in the reference KG. This task is repeated several times until the correct annotation is identified (annotation with highest score).

Once a table has been semantically annotated, it can be used to populate the semantic description of informative services that consume the data of the table. In the second part of this thesis, the STI is used to (semi) automatically create semantic descriptions that correlate API's properties at a semantic level [62, 18, 19]. To enhance interoperability we propose to enrich OpenAPI descriptions, a standard that is getting popular in the domain of services. By exploiting semantic descriptions, machines can automatically invoke services and process the results. Moreover, semantic descriptions are enabling the definition of automatic procedures for discovery and composition of services.

The RDF output of a STI process can be difficult to exploit by human readers, hence a more human friendly representation is desirable. The third part of this thesis proposes an approach aimed at converting groups of triples into natural-language textual descriptions [17]. The conversion takes place by applying a Neural Machine Translation (NMT) supervised technique: a neural network model receives a set of RDF triples in input, and produces a textual description containing the information present in the triples. The use of a supervised approach, however, requires the construction of a training dataset composed of a set of triple-text pairs to train the model. For this reason, we propose a pipeline for the creation of a new training dataset. This pipeline uses a combination of state-of-the-art tools to make alignments between triples and text.

An important contribution of the thesis are the tools that implement the proposed theoretical approaches:

1. STI approach - MantisTable: [bitbucket.org/disco\\_unimib/mantistablex-tool.py/](https://bitbucket.org/disco_unimib/mantistablex-tool.py/)
2. STI validation tool - STILTool: [bitbucket.org/disco\\_unimib/stiltool/](https://bitbucket.org/disco_unimib/stiltool/)
3. Semantic Description for API - AutomAPIC: [bitbucket.org/disco\\_unimib/automapic-tool/](https://bitbucket.org/disco_unimib/automapic-tool/)
4. Pipeline for RDF-text alignment - SeaLion: [bitbucket.org/disco\\_unimib/sealion/](https://bitbucket.org/disco_unimib/sealion/)
5. Tool that creates soccer articles automatically for RDF - GazelLex<sup>3</sup>



<sup>3</sup>The project was funded by a Digital News Innovation Fund call ([newsinitiative.withgoogle.com/dnifund/](https://newsinitiative.withgoogle.com/dnifund/)) and commissioned by an italian newspaper publisher; currently it is not possible to release the code, but the description of the project is in [17].

## Published Works

The thesis is based on the following papers:

1. Lucky, M. N., Cremaschi, M., Lodigiani, B., Menolascina, A., and De Paoli, F. (2016). Enriching api descriptions by adding api profiles through semantic annotation. In Sheng, Q. Z., Stroulia, E., Tata, S., and Bhiri, S., editors, *Service-Oriented Computing*, pages 780–794, Cham. Springer International Publishing
2. Cremaschi, M. and De Paoli, F. (2017). Toward automatic semantic api descriptions to support services composition. In De Paoli, F., Schulte, S., and Broch Johnsen, E., editors, *Service-Oriented and Cloud Computing*, pages 159–167, Cham. Springer International Publishing
3. Bianchi, F., Palmonari, M., Cremaschi, M., and Fersini, E. (2017). Actively learning to rank semantic associations for personalized contextual exploration of knowledge graphs. In Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., and Hartig, O., editors, *The Semantic Web*, pages 120–135, Cham. Springer International Publishing
4. Cremaschi, M. and De Paoli, F. (2018). A practical approach to services composition through light semantic descriptions. In Kritikos, K., Plebani, P., and de Paoli, F., editors, *Service-Oriented and Cloud Computing*, pages 130–145, Cham. Springer International Publishing
5. Cremaschi, M., Rula, A., Siano, A., and De Paoli, F. (2019c). Mantistable: A tool for creating semantic annotations on tabular data. In Hitzler, P., Kirrane, S., Hartig, O., de Boer, V., Vidal, M.-E., Maleshkova, M., Schlobach, S., Hammar, K., Lasierra, N., Stadtmüller, S., Hose, K., and Verborgh, R., editors, *The Semantic Web: ESWC 2019 Satellite Events*, pages 18–23, Cham. Springer International Publishing
6. Cremaschi, M., Rula, A., Siano, A., and De Paoli, F. (2019d). Semantic table interpretation using mantistable. In *The Fourteenth International Workshop on Ontology Matching 2019 (to appear)*
7. Cremaschi, M., Bianchi, F., Maurino, A., and Pierotti, A. P. (2019a). Supporting journalism by combining neural language generation and knowledge graphs. In *Sixth Italian Conference on Computational Linguistics 2019 (to appear)*

8. Cremaschi, M., De Paoli, F., Rula, A., and Spahiu, B. (2019b). A fully automated approach to a complete semantic table interpretation. *Manuscript submitted for publication*

## 1.4 Thesis Structure

The rest of this thesis are structured as follows:

- Chapter 2 introduces some core concepts and notation related to Semantic Web standards, Linked Data publishing, tables, API and Neural Network (NN).
- Part I: Semantic Table Interpretation
  - Chapter 3 positions this thesis with respect to related works. It gives an overview of the state-of-the-art techniques in the field of STI.
  - Chapter 4 describes our fully automated approach to a complete STI.
  - Chapter 5 describes the MantisTable tool, a tool that integrates the STI set out in this thesis. In the final part of the chapter an evaluation of the approach is proposed through the use of different Gold Standards. The last section shows a formalisation of the approach.
- Part II: API Composition using Semantic Annotations
  - Chapter 6 shows an analysis of the state-of-the-art relating to methods for the description and composition of APIs.
  - Chapter 7 the section proposes a new approach for the creation of semantic descriptions for the APIs and the use of these in the composition of services.
  - Chapter 8 describes the AutomAPIc tool. This tool allows the analysis of the semantic annotations in the descriptions in order to identify the possible compositions between different services. In the final part of the chapter, the evaluation part of the approach is presented.
- Part III: Natural Language Generation with Neural Networks
  - Chapter 9 shows an analysis of the pipeline of NLG. In the second part a comparative analysis of the datasets for the generation of natural language from triple RDF is proposed.



- 
- Chapter 10 proposes the description of a alignment pipeline (from RDF to text) for the creation of datasets for the training of neural networks.
  - Chapter 11 describes the SeAlion tool and GazeLex tool. The first implements the alignment pipeline, while the second shows the use of neural networks for the automatic generation of newspaper articles.
  - Part IV: Natural Language Generation from Tabular Data
    - Chapter 12 deals with the study of the interpretation of tables by the user, with the ultimate aim of identifying a series of requirements to be implemented in a tabular data lexicalization tool.
    - Chapter 13 describes the MantisTablex tool which allows to generate text starting from the data contained in a table, selecting the information in relation to the user’s information needs.
  - Chapter 14 finally summarises the results and compares them to the motivation presented in this chapter. We collect and comment on research questions that remain open, and outline the expected future work and impact of the research topic.



# **Chapter 2**

## **Preliminaries**

This chapter covers the fundamental knowledge of topics that are essential to understand this dissertation. It starts by describing the Semantic Web, resource description formats, and discusses publicly open KG. Afterwards, a definition for a table is provided along with a classification according to the different type of structure. The last two Sections are dedicated to the definition of the concept of Application Programming Interface (API), and the use of NN as a translation instrument that goes from data to natural language.

## 2.1 The Semantic Web and Resource Description Format

Most of the contents that are present online are designed to be read and interpreted by humans, therefore they are difficult to understand for machines, which fail to process them effectively. Computers can scan a Web page and identify critical elements such as titles, paragraphs and links to other pages, but, in general, they are not able to get the semantics, or to understand the meaning of these elements and the relationships among them. The continuous and rapid evolution of the Web is, however, increasingly starting to switch from a document-based Web to the so-called “Semantic Web”, which is not to be considered as a type of separate Web, but rather as an extension of the current one. In the Semantic Web, the information is expanded by a unique and well-defined meaning to allow better cooperation between people and computers [8]. The basic unit of information exchange is no longer the document as a whole, but the data included in it: while the first can be read, the second can be processed in different ways to create new information. With this approach, the Web becomes a *Web of Data*, a space meant for global data sharing. The Semantic Web can also be considered as a third evolutionary phase of it (Web 3.0), whose goal is to create an environment that is both human- and machine-understandable.

The Semantic Web provides a way to publish data in structured formats using standards such as RDF and Web Ontology Language (OWL). The machine-readable aspect of web documents can be achieved by adding supplementary tags to the current Hypertext Markup Language (HTML) based documents. For instance, compare the HTML statement given in line 1 of Listing 2.1 with the one in line 2. The latter has been augmented with semantic tags that machines could use to interpret the HTML content: the tag `rdf:about` qualifies the text “Semantic Web” by setting a relation with the “about” concept defined in the rdf KG.

```
1 <item>Semantic Web</item>  
2 <item rdf:about="https://www.w3.org/standards/semanticweb/">SemanticWeb</item>
```

Listing 2.1 Example of HTML statement with semantic web tags.

### 2.1.1 Linked Data

The technology that allows the aggregation is called “Linked Data” [10], and enables everyone to contribute to the Semantic Web by (i) publishing their data in a shared format, such as the RDF format, and (ii) making each of these resources unambiguously identified by a specific Uniform Resource Identifier (URI).

A resource could be anything: a person, a country, an abstract idea or a simple document on the Web. Linked Open Data (LOD) refers to the open-access RDF datasets that provide links among them. Specifically, the term LOD refers to a series of best practices for the publication of structured data on the Web. The principles of LOD are that: (i) URIs have to be used as names for things [8]; (ii) provide useful information about what a name identifies when it is looked up, using open standards such as RDF, SPARQL, etc; (iii) use HTTP URIs so that people can look up those names; and (iv) refer to other things using their HTTP URI-based names when publishing data on the Web. The LOD Cloud is the largest collections of interlinked LOD datasets on the web; is an ongoing project to which everybody can contribute. Thanks to projects such as DBpedia [3] and Wikidata [114], the LOD Cloud has already reached a significant dimension. The current LOD Cloud [1] is showed in Figure 2.1 where datasets are grouped by domains. Cross-domain datasets such as DBpedia and Freebase [11] are visualised as main hubs with many links coming from other datasets.

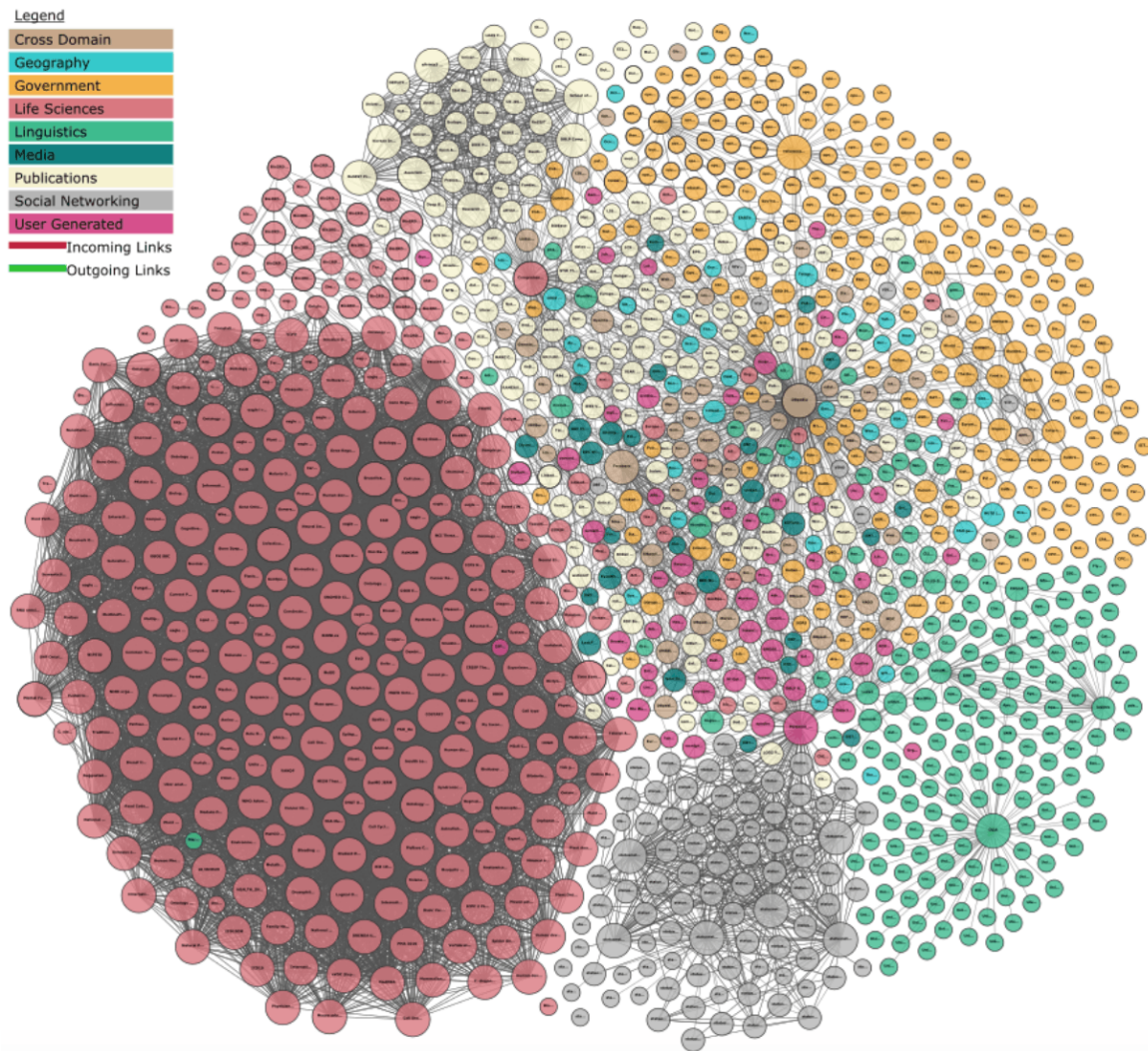


Figure 2.1 Linked Open Data Cloud showing links between datasets. lod-cloud.net.

## 2.1.2 Resource Description Format

RDF is a standard model proposed by W3C<sup>1</sup>, designed for those situations in which Web information needs to be processed by applications as well as shown to the user. The information thus represented can be exchanged between various applications without losing meaning. As we said, this data model allows data to become LOD. It is based on the idea that URIs can be used to name not only entities but also relationships between things.

RDF is composed of three parts<sup>2</sup>:

- resource: what RDF describes. It can be anything (documents, people, physical objects and abstract concepts) and resources are always identified by a URI;
- property: a property, attribute or relationship used to describe a resource;
- statement: it describes the resource and it is represented in the form of triples,  $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$  where:
  - subject can be a URI or a BLANK NODE;
  - object can be a URI, or BLANK NODE or a LITERAL;
  - predicate is the relationship between subject and object, and is a URI.

The use of the word “predicate” to indicate relationships comes from the fact that the relation, from a logic point of view, can be seen as a two-arguments predicate  $p(x, y)$ .

An example of a triple with a resource as object is shown in Listing 2.2. A triple with a literal as object is shown in Listing 2.3.

```
1 (http://dbpedia.org/page/Mount_Everest ,  
2 http://dbpedia.org/ontology/mountainRange ,  
3 http://dbpedia.org/page/Himalayas)
```

Listing 2.2 Example of triple with a resource as object.

```
1 (http://dbpedia.org/page/Mount_Everest ,  
2 http://dbpedia.org/ontology/elevation ,  
3 "8848")
```

Listing 2.3 Example of triple with a literal as object.

The same resource can be mentioned several times in different triples, both as a subject and as an object: this allows us to identify connections between triples that can be displayed in a graph called RDF Graph (Figure 2.2), where the nodes represent resources and arcs represent predicates.

<sup>1</sup>[www.w3.org/RDF/](http://www.w3.org/RDF/)

<sup>2</sup>[www.w3.org/TR/rdf11-concepts/](http://www.w3.org/TR/rdf11-concepts/)

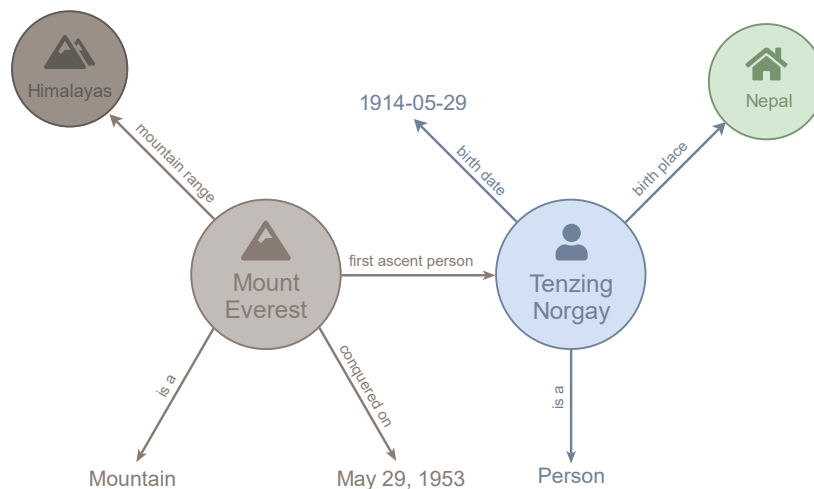


Figure 2.2 Example of RDF Graph.

The representation of RDF triples can be achieved using different kinds of syntax, among which the most common ones are: RDF/XML (.rdf), N-Triples (.nt), Turtle (.ttl) e JSON-LD (.json).

**RDF/XML**<sup>3</sup>. This type of syntax represents a RDF graph as a XML document. It is the first kind of syntax that has been introduced. However, to the present day, there are more simple and brief representations that are generally preferred. A feature in common with other syntax is the option of defining prefixes in the initial part of the XML document, to identify part of URIs so that we can express them in a shortened form. An example of RDF/XML representation is shown in Figure 2.3.

**N-Triples**<sup>4</sup>. The triples expressed as N-Triples are sorted in a document where each line represents a single triple (<subject> <predicate> <object>); the RDF/XML example in Figure 2.3 is written in N-Triples notation in Figure 2.4.

**Turtle**<sup>5</sup>. This kind of syntax allows us to represent a RDF Graph in a compact and readable form, thanks to the option of defining prefixes at the start of the document, so as to make possible to shorten each triple; triples clustering is also planned if they share the same subject (in this case, the triples are separated by “;” and the subject is written only once), or if they share the same subject and predicate (subject and predicate are listed only once and triples are separated by “;”)

The Figure 2.5 shows the RDF/XML example in Figure 2.3 written in Turtle notation.

<sup>3</sup>[www.w3.org/TR/rdf-syntax-grammar/](http://www.w3.org/TR/rdf-syntax-grammar/)

<sup>4</sup>[www.w3.org/TR/n-triples](http://www.w3.org/TR/n-triples)

<sup>5</sup>[www.w3.org/TR/turtle/](http://www.w3.org/TR/turtle/)



**JSON-LD**<sup>6</sup>. This type of serialisation is mainly intended for Web-based environments; it is based on JSON format and therefore it proves easy to read and write. The RDF/XML example in Figure 2.3 is listed in JSON-LD notation in Figure 2.6.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ns0="http://dbpedia.org/ontology/"
  xmlns:schema="http://schema.org/"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">

  <foaf:Person rdf:about="http://dbpedia.org/resource/Bob_Marley">
    <rdfs:label xml:lang="en">Bob Marley</rdfs:label>
    <rdfs:label xml:lang="fr">Bob Marley</rdfs:label>
    <rdfs:seeAlso rdf:resource="http://dbpedia.org/resource/Rastafari"/>
    <ns0:birthPlace>
      <schema:Country rdf:about="http://dbpedia.org/resource/Jamaica">
        <rdfs:label xml:lang="en">Jamaica</rdfs:label>
        <rdfs:label xml:lang="it">Giamaica</rdfs:label>
        <geo:lat rdf:datatype="http://www.w3.org/2001/XMLSchema#float">17.9833</geo:lat>
        <geo:long rdf:datatype="http://www.w3.org/2001/XMLSchema#float">-76.8</geo:long>
        <foaf:homepage rdf:resource="http://jis.gov.jm"/>
      </schema:Country>
    </ns0:birthPlace>

  </foaf:Person>

</rdf:RDF>
```

Figure 2.3 Example of RDF/XML syntax.

```
<http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/2000/01/rdf-schema#label> "Bob Marley"@en .
<http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/2000/01/rdf-schema#label> "Bob Marley"@fr .
<http://dbpedia.org/resource/Bob_Marley> <http://www.w3.org/2000/01/rdf-schema#seeAlso> <http://dbpedia.org/resource/Rastafari> .
<http://dbpedia.org/resource/Bob_Marley> <http://dbpedia.org/ontology/birthPlace> <http://dbpedia.org/resource/Jamaica> .
<http://dbpedia.org/resource/Jamaica> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Country> .
<http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2000/01/rdf-schema#label> "Jamaica"@en .
<http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2000/01/rdf-schema#label> "Giamaica"@it .
<http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2003/01/geo/wgs84_pos#lat> "17.9833"^^<http://www.w3.org/2001/XMLSchema#float> .
<http://dbpedia.org/resource/Jamaica> <http://www.w3.org/2003/01/geo/wgs84_pos#long> "-76.8"^^<http://www.w3.org/2001/XMLSchema#float> .
<http://dbpedia.org/resource/Jamaica> <http://xmlns.com/foaf/0.1/homepage> <http://jis.gov.jm/> .
```

Figure 2.4 Example of N-Triples syntax.

<sup>6</sup>[www.w3.org/TR/json-ld/](http://www.w3.org/TR/json-ld/)

```

@prefix dbr: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema: <http://schema.org/> .

dbr:Bob_Marley
  a foaf:Person ;
  rdfs:label "Bob Marley"@en ;
  rdfs:label "Bob Marley"@fr ;
  rdfs:seeAlso dbr:Rastafari ;
  dbo:birthPlace dbr:Jamaica .

dbr:Jamaica
  a schema:Country ;
  rdfs:label "Jamaica"@en ;
  rdfs:label "Giamaica"@it ;
  geo:lat "17.9833"^^xsd:float ;
  geo:long "-76.8"^^xsd:float ;
  foaf:homepage <http://jis.gov.jm/> .

```

Figure 2.5 Example of Turtle syntax.

```

[
  {
    "@id": "http://dbpedia.org/resource/Bob_Marley",
    "@type": ["http://xmlns.com/foaf/0.1/Person"],
    "http://www.w3.org/2000/01/rdf-schema#label": [
      {"@value": "Bob Marley", "@language": "en"},
      {"@value": "Bob Marley", "@language": "fr"}
    ],
    "http://www.w3.org/2000/01/rdf-schema#seeAlso": [{"@id": "http://dbpedia.org/resource/Rastafari"}],
    "http://dbpedia.org/ontology/birthPlace": [{"@id": "http://dbpedia.org/resource/Jamaica"}]
  },
  {
    "@id": "http://dbpedia.org/resource/Jamaica",
    "@type": ["http://schema.org/Country"],
    "http://www.w3.org/2000/01/rdf-schema#label": [
      {"@value": "Jamaica", "@language": "en"},
      {"@value": "Giamaica", "@language": "it"}
    ],
    "http://www.w3.org/2003/01/geo/wgs84_pos#lat": [
      {"@value": "17.9833", "@type": "http://www.w3.org/2001/XMLSchema#float"}
    ],
    "http://www.w3.org/2003/01/geo/wgs84_pos#long": [
      {"@value": "-76.8", "@type": "http://www.w3.org/2001/XMLSchema#float"}
    ],
    "http://xmlns.com/foaf/0.1/homepage": [{"@id": "http://jis.gov.jm/"}]
  },
  {"@id": "http://dbpedia.org/resource/Rastafari"},
  {"@id": "http://jis.gov.jm/"},
  {"@id": "http://schema.org/Country"},
  {"@id": "http://xmlns.com/foaf/0.1/Person"}
]

```

Figure 2.6 Example of JSON-LD syntax.

### 2.1.3 Resource Description Format Schema

RDF alone is not expressive enough to encapsulate all the knowledge that an approach with more semantics can convey. However, it has been extended with RDF schema, or Resource Description Format Schema (RDFS), that is a simple ontology language that allows us to define properties and classes (Listing 2.4).

```
1 example:Animal rdf:type rdfs:Class .
```

Listing 2.4 Example of triple with RDF schema.

`rdf:type` is used to indicate the type of a resource, and `rdfs:Class` can be used to define a set of resources that represents entities in the real world. In the triple above we stated an example of this: `Animal` is an RDFS Class. We can now populate the Class with new elements (Listing 2.5).

```
1 resource:Otter rdf:type example:Animal .  
2 resource:Hamster rdf:type example:Animal .
```

Listing 2.5 Example of triple with classes.

It is possible to define a hierarchical relation between Classes using the term `rdfs:subClassOf` (Listing 2.6).

```
1 example:Organisms rdf:type rdfs:Class .  
2 example:Animal rdfs:subClassOf example:Organisms .
```

Listing 2.6 Example of triple with subclasses.

From these statements, thanks to the relation introduced by `rdfs:subClassOf`, it is possible to make an inference. For example, `resource:Hamster` and `resource:Otter` belong to the class `example:Animal`. Given that `example:Animal` is a subclass of `Organisms`, we can infer that the two resources are also part of `example:Organisms`. In this way, we can begin to form an ontology.

## 2.1.4 Web Ontology Language

RDFS is considered a “lightweight” ontology language, however, other and more powerful languages do exist. Probably, the most well known is OWL, which is much more expressive than RDFS, and allows detailed and efficient reasoning. OWL gives us the possibility to compose new classes from others, and also to give them an internal structure. With OWL, it is possible to build a class as the union or the intersection of other classes, or as a complement of one of them, or to enumerate its members. OWL has some predefined terms, like `owl:Thing` (that acts as super-class of every OWL class) or `owl:ObjectProperty`. Going back to the previous example, we can describe the class `Organisms` (and its subclasses) in a more effective way (Listing 2.7).

```
1 SubClassOf( :Organisms owl:Thing )
2 SubClassOf( :Animal :Organisms)
3 SubClassOf( :Plant :Organisms)
4 SubClassOf( :Bacteria :Organisms)
5 ClassAssertion( :Animal {Otter, Hamster, Fish} )
6 ClassAssertion( :Plant {Pine, Oak} )
7 ClassAssertion( :Bacteria {Cyanophyta, Nitrospirae} )
```

Listing 2.7 Example of OWL.

In the example, we defined a child class of the built-in class `owl:Thing`, and three subclasses (`SubClassOf`) with two or three members each (using `ClassAssertion`). Propositional logic operators can be used to build new classes using the existing ones. For example, we can make use of these operators to create an extemporaneous class by selecting only two subclasses of `Organisms` (imagine that we want to retrieve only eukaryotes `Organisms`: with these operators we would be able to search for `Plant` or `Animal`, excluding `Bacteria`). Even though an individual can be an instance of several classes, in some cases we need to state that belonging in one class excludes membership in another. OWL enables us to do that, thanks to what is called class disjointness (Listing 2.8).

```
1 DisjointClasses( :Plant :Animal )
```

Listing 2.8 Example of `DisjointClasses` OWL.

These examples, however, are just the beginning. OWL allows us to compose new properties that describe relations between individuals, to make statements about classes, property, or individuals.

### 2.1.5 Knowledge Graph

The term Knowledge Graph was coined by Google in 2012, referring to their use of semantic knowledge in Web Search, and is also used to refer to Semantic Web knowledge bases such as DBpedia. From a broader perspective, any graph-based representation of some knowledge could be considered a knowledge graph (this would include any kind of RDF dataset, as well as description logic ontologies) [74]. An example is shown in Figure 2.7. It's a form of knowledge representation based on a graph in which nodes represent entities and the arcs that connect them represent the relations between them. For example, in the sentence "Shakespeare wrote the Hamlet", "Shakespeare" and "Hamlet" represent two nodes, "wrote", on the other hand, represents the arc that links them. Entities can also have some types characterised by a "is a" relation (e.g. Shakespeare was a writer). Generally, the set of possible types and relation is organised in a schema or in an ontology that defines their interrelations.

A KG follows this properties [74]:

- the real-world entities and their interrelations are represented in a graph layout;
- the possible entities' classes and relations are defined in a schema;
- it is possible to define arbitrary relations between entities;
- a great portion of domains that exist in the real world are processed.

This excludes, for example, all the knowledge bases that refer to tiny domains, or ontologies that don't include real instances but only abstract concepts.

So, in a KG, we can find:

- classes or concepts: they represent the abstract references of what we want to describe, e.g. "writer";
- instances or entities: they represent the factual materialisation of classes; e.g. "Shakespeare" is an instance of the class "writer";
- datatypes: the types, such as String or Date, of literal data, namely the data that cannot be associated to any concept;
- properties: they represent the existing connections between concepts, datatypes, instances and properties themselves.

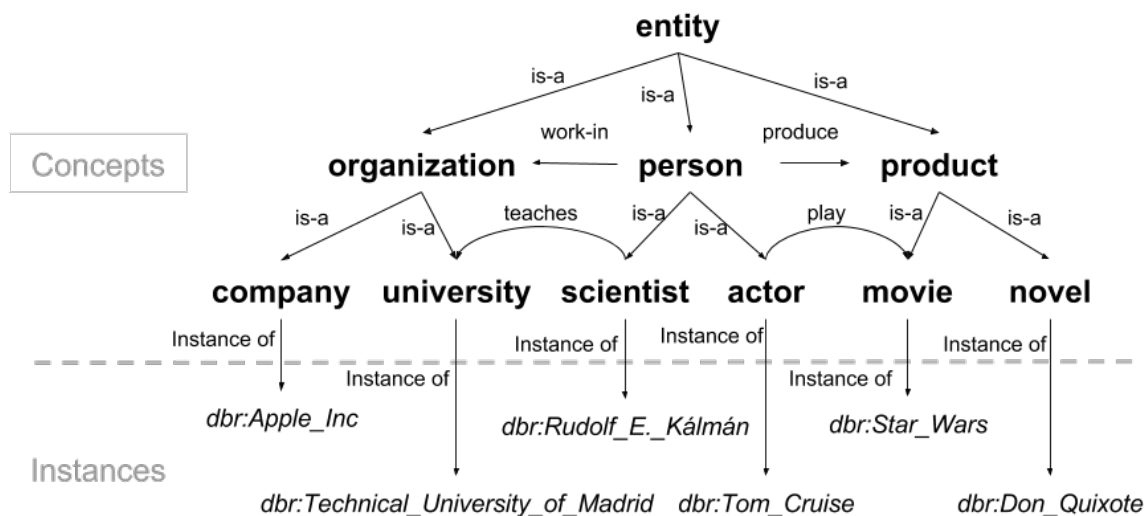


Figure 2.7 Example of Knowledge Graph.

Given the popularity of publishing RDF data as LOD, many domain specific and cross-domain knowledge bases have emerged. For instance, DBpedia [3], YAGO [99], Wikidata [114] and Freebase [11] are open-access knowledge bases that have factual information about entities and their relationships. The BabelNet [72] is another example of open-access knowledge base that contains multilingual linguistic data by combining data from other knowledge bases.

## 2.2 Tables

Tables refer to a matrix-type data structure that organises information in rows and columns; a cell is the intersection of a column with a row. Data is contained within a cell (Figure 2.8) [121].

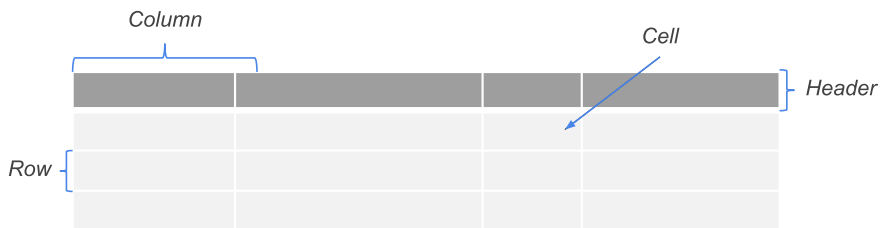


Figure 2.8 Structure of a table.

### Types of Tables

A table can also have one or more *headers* containing an identifier of the column (or row) under review; it is situated in the first row when we talk about horizontal tables (Figure 2.9), in the first column when we talk about vertical tables (Figure 2.10) or in both places when we deal with bi-dimensional tables (Figure 2.11).

First Name	Last Name	Email
Grayson	Bell	g.bell@domain.com
Ayaana	Jensen	a.jensen@domain.com

Figure 2.9 Example of a horizontal table.

First Name	Grayson	Ayaana
Last Name	Bell	Jensen
Email	g.bell@domain.com	a.jensen@domain.com

Figure 2.10 Example of a vertical table.

	Monday	Tuesday	Wednesday	Thursday	Friday
9-13	Work	Rest	Work	Work	Rest
14-18	Work	Rest	Rest	Work	Work

Figure 2.11 Example of a two-dimensional table.

A further classification is provided based on the structure of the table; in particular we have:

- Complex tables, when they contain multi-line cells, nested headers, multi-dimensional data or other irregular shapes (Figure 2.12); these features can coexist;

Day	Seminar		
	Schedule		Topic
	Begin	End	
Monday	8:00 a.m.	5:00 p.m.	Introduction to XML
			Validity: DTD and Relax NG
Tuesday	8.00 a.m.	11:00 a.m.	XPath
	2:00 p.m.	2:00 p.m.	XSL Transformations
Wednesday	8:00 a.m.	12:00 p.m.	XSL Formatting Objects

Figure 2.12 Example of a complex table.

- Simple tables, when they have a horizontal or vertical header, a regular form and they do not include any of the features of the complex tables (Figure 2.9 and Figure 2.10). This type of table is the most common one on the web.

A table is called relational when the columns have relations between them; an example is depicted in Figure 2.13.

Title	Artist	Company	Price	Year
Empire Burlesque	Bob Dylan	Columbia	10.90	1985
Hide your heart	Bonnie Tyler	CBS records	9.90	1988

Figure 2.13 Example of relational table: the table' subject is defined by the column "Title" and it identifies some music albums. The other rows are related to it because they represent, respectively, the artist, the record label, the price and the release year.

Finally, we define a table as *well formed* if:

- the values within a column belong to the same domain;
- the number of columns is the same for each row;
- the number of rows is the same for each column.



## 2.3 Application Programming Interface

With the term API we indicate an interface that facilitates the communication between the hardware and the programmer, in order to avoid duplicating each time the code necessary to implement recurring operations and to hide the real complexities of the machine. The concept of API has been evolving and, in a World Wide Web perspective, there is the need to create new interfaces that abstract the implementation from the protocol. Consequently, web applications developed web API, that can be used by the application itself or by some third party software. Web APIs allows websites, mobile apps and any other device to interact and share data online. The development of Web APIs breaks down into two paths: Simple Object Access Protocol (SOAP), mainly used in enterprise environments, and REpresentational State Transfer (REST), the last technology created and already the most used in mobile environments and Internet of Things (IoT) applications.

SOAP is a lightweight protocol meant for information exchange in a decentralised and distributed environment. It is based on eXtensible Markup Language (XML), which consists of three parts: an envelope that defines the structure used to describe what's inside a message and how to process it, a group of encoding rules and lastly a schema to define a Remote Procedure Calls (RCP) and how to respond. SOAP can send messages using various transport protocols, but for the most part it uses HTTP [76].

REST can't be described as a protocol like we did for SOAP, but rather as an architectural style. REST is a combination of principles and design methods that outline how the resources, basic functioning elements, are defined and addressed. A resource is any entity that can be addressable through the Web. Speaking of Web, the best way to locate a resource is provided by the concept of URI, since it can be easily tracked online. The principles that a REST architecture must respect are:

- client-server: to follow this principle, the activities that we have to perform on client and on server must be kept separated. In particular, the client can't deal with data processing and persistence, while the server can't manage the user's status and the graphic interface. By maintaining this separation, it is possible to develop these two components in a distinctive way, keeping the interfaces unchanged and making them scalable;
- stateless: the server must not keep track of the communication it shares with the client. Each request made by the client towards the server must be considered independent from the others, so each request should contain the necessary information to be processed by the server;

- cacheable: the information shared by client and server can be saved by caching. This allows us to evaluate when information is not valid or correct anymore, besides limiting the number of interactions between client and server, thus making possible to improve the system's scalability and performance;
- uniform interface: a coherent communication interface between components allows us to simplify the whole system, since we can develop client and server independently; furthermore, it improves the interactions' visibility. The only downside of this principle is the lowered system's efficiency, since the information gets transferred in standardised form rather than in a specific one suited for the application. The REST interfaces are designed to be efficient when it comes to large-scale transferring of hypermedia data;
- code-on-demand: REST grants the client features that can be extended by downloading and running code in the form of applet or script.

Another main principle of REST's architecture based applications is HATEOAS, abbreviation of Hypermedia as the Engine of Application State. This concept refers to a client that interacts with a web application exclusively through hypermedia that are dynamically provided by the server of the application itself. This way, a REST client doesn't need additional knowledge besides normal comprehension of hypermedia to interact with a specific app or server.

REST fully exploits the semantics and the HTTP protocol's methods, so its goal is to create servers capable of accepting HTTP methods defined by the protocol as input, to redefine them and use them to its advantage. The seven methods that get redefined in REST apps are:

- POST: it sends data to the server so that they can be processed by the given URI;
- GET: it requests the resource associated with the given URI;
- PUT: it updates or modifies the state of the resource associated with the given URI;
- DELETE: it deletes the resource associated with the given URI;
- PATCH: it allows the partial modification of a resource;
- HEAD: it is used to retrieve only the header information of the server's response. It can also be used to find out if a specific resource does exist or to learn information about the resource itself;
- OPTIONS: it allows us to check which are the methods supported by each resource. This method isn't usually specified during the definition phase.

## 2.4 Neural Networks

NN are mathematical models composed of artificial neurons. They are inspired by the biological functioning of the human brain and require advanced hardware chip to work. These models, can learn how to process the input: at the beginning, the network has no knowledge, but it can acquire concepts of a high level of abstraction, thanks to the so-called backpropagation. The most used Neural Networks (NNs) are the ones called Recurrent Neural Networks (RNNs). They have the advantage to be able to process long sequences of input.

### Long Short-Term Memory Networks

A neural network is composed of a network of small computing units. Each unit takes a vector of input values and, after some computations, generates a single output value. Modern networks have different layers of these units, and are often deep (this is why their use is called “deep learning”). Between input layer and output layer there is the hidden layer, formed by the so-called hidden units. We can see a simple example of NNs architecture in Figure 2.14. The functioning core of neural networks is that values in input vector of each neural unit are weighted by different factors. Moreover, a bias term is added to the vector.

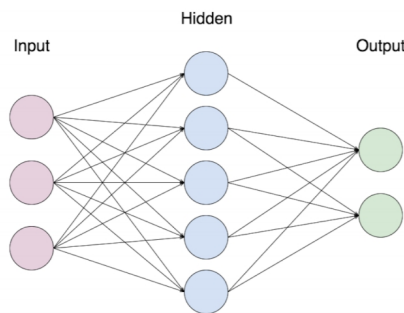


Figure 2.14 A simple example of NNs architecture.

The most straightforward kind of neural network is called “feed-forward network”. Often applied to classification, in this kind of network the computation proceeds from one layer to the next, and no outputs are passed back to lower layers. In this network, the correct output for the computation is known. So, the training procedure aims to learn the weights and the bias parameters for each layer, to obtain from the network a result as close as possible to the correct one [46]. With language models, feed-forward networks can be applied to make predictions about the next word in a sequence, having a limited context of preceding words. What’s different about RNNs is that they are explicitly designed to process sequences [46].

This NN makes possible to handle inputs of variable lengths without the use of windows with arbitrary size. A Recurrent Neural Network (RNN) contains a cycle within its connections: the value of a unit also depends on its own output (from a previous time-step) as an input. These architectures proved to be very useful when we deal with language issues. The most simple ones are Simple Recurrent Networks (SRNs). The main difference from a feed-forward network is that the input sent to the hidden layer receives the contribution from a previous one. This “recurrence” is a form of memory, or context, that influences the decisions that have to be made later. There is no fixed-length limit to this context, but the main problem of this simple RNNs is that it is difficult to train a system that makes use of information when it is particularly distant from the current point of processing. Hidden states tend to consider local information, but in many language application, long-distance information can be critical. So, more complex network architectures as Long Short-Term Memory (LSTM) networks have been designed. These networks can handle the task of maintaining contextual information over time. In these approaches, context is treated as a kind of memory unit that has to be managed. Information that is no longer needed must be forgotten, and information that will be needed for later tasks must be remembered.

LSTM satisfies these needs by using specialised neural units that employ “gates” to control the flow of information entering and exiting the units of network layers.

LSTM networks are used for tasks such as lexicalisation [44], because of the many advantages they provide. As we said, one of them is that they do not have limitations in input and output length. Furthermore, input and output are not independent, and that is a vital advantage in language generation. To predict a word in a sentence, it is useful to know and consider the previous one, and the hidden states of the network keep in memory what happened in previous time-steps. This way, LSTM can combine the previous state, the memory collected and the input, allowing dependencies to be maintained in the long run.

As described, RNN memorises the sequential context using cycles. The sequential nature of these models, however, makes a computation expensive [67, 120]. This point makes it difficult to scale to large corpora. The Transformer architecture [108] replaces RNN cells with self-attention and point-wise fully connected layers, which are highly parallelisable and thus cheaper to compute. Together with positional encoding, Transformers can capture long-range dependencies with vague relative token positions. This results in a coarse-grained sequence representation at the sentence level. Recent works such as GPT (or GPT2) [81, 82] and BERT [27] show that the representations learned on large-scale language modelling datasets are useful for fine-tuning both sentence-level tasks, such as GLUE benchmark [116] and token-level tasks. There are two existing strategies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The feature-based

---

approach, such as ELMo [77], uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) [81], introduces minimal task-specific parameters and is trained on the downstream tasks by simply fine-tuning all pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations. Therefore, these Language Models (LMs) can be used, after appropriate modifications, also in NMT activities.



# **Part I**

## **Semantic Table Interpretation**





# Chapter 3

## State-of-the-art

Tables represent a universal language used to express relational data and they're actually very common on the Web. To size the phenomenon, an analysis reports 25M relational tables within 500M Web pages [59], and another estimates 150M HTML tables represented in the English language [12]. A more recent work highlighted the popularity of tables by collecting 233M of them through the analysis of the Common Crawl<sup>1</sup> repository [55]. This large increase can be linked to the uptake of the Open Data movement, whose purpose is to make a large number of tabular data sources freely available, addressing a wide range of domains, such as finance, mobility, tourism, sports, or cultural heritage [73]. Tables express references to entities, attributes and relations, but since such data are shown in form of free text and without using a controlled vocabulary, Web searches struggle to benefit from this huge source of information. Tables are essential to perform queries, but the implicit or visual structures employed in tables are not easily machine-readable. In order to allow computers to interpret, combine and reuse such data for several artificial-intelligence tasks (such as classification, clustering, filtering, and retrieval [71]), the semantic of data should become explicit. The retrieval of semantic information from tables, therefore, plays a crucial role in the realisation of Semantic Web; the research aimed at solving this particular problem takes the name of STI. The input of STI is (1) a *well-formed and normalised* relational table (i.e. a table with headers and simple values, thus excluding nested and figure-like tables), as the one in Figure 3.1, and (2) a *KG* which describes real world entities in the domain of interest (i.e. a set of concepts, datatypes, predicates, instances, and the relations among them), as the example in Figure 3.2. The output returned is a semantically annotated table, as shown in Figure 3.3.

---

<sup>1</sup>commoncrawl.org

Column $C_j$		Cell $(i,j)$		Header $\mathcal{H}$
Name	Coordinates	Height	Range	
Mont Blanc	45°49'57"N 06°51'52"E	4808	Mont Blanc massif	Rows $\mathcal{R}$
Lyskamm	45°55'20"N 07°50'08"E	4527	Pennine Alps	
Monte Cervino	45°58'35"N 07°39'31"E	4478	Pennine Alps	
Columns $\mathcal{C}$				

Figure 3.1 Example of a well-formed relational table, with labels that are used in this Section.

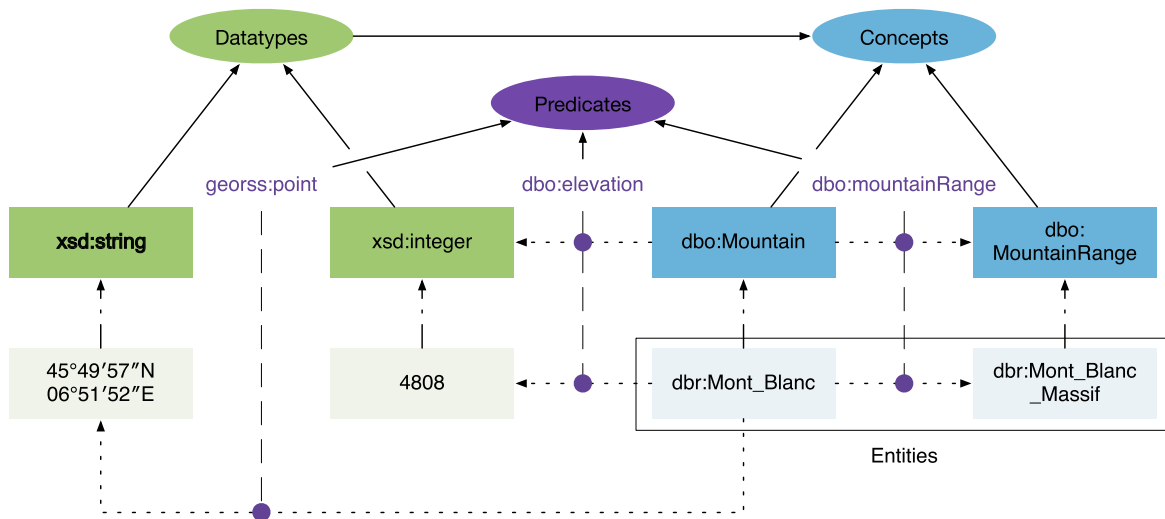


Figure 3.2 A sample of KG.

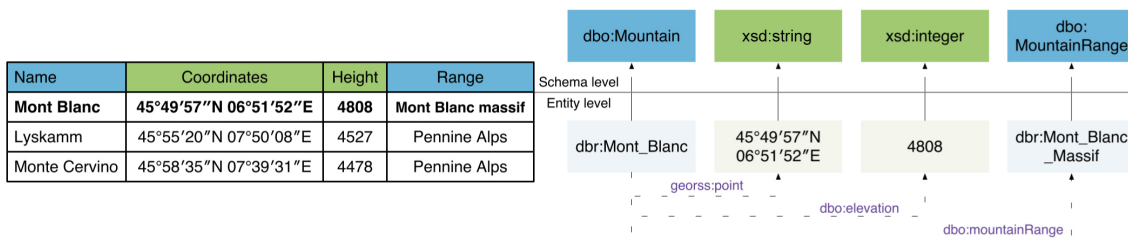


Figure 3.3 Example of a table's semantic annotation.

So, the purpose of STI is, as previously mentioned, to read tables' structures and semantics, associating their content to semantic concepts existing in a KG. To obtain this result, we need to establish some fundamental points on which to base this kind of activity, such as [123]:

- Named Entity Disambiguation: the content of cells that belong to the entity columns (*NE-columns*) gets associated to a regular entity of the knowledge base;
- Column Classification: the *NE-columns* are annotated thanks to the use of a semantic concept, while the columns that contain literal data like, for example, numbers, strings or dates (*L-columns*), are annotated by using properties that belong to the concept of the subject column;
- Relation Extraction: some binary relations between columns are detected.

Given that, generally, a table addresses a specific topic found in the subject column, we can also add to the previously mentioned activities the detection of the *S-column* among the various *NE-columns*; this implies that binary relations will always occur between the subject column and each one of the others.

To illustrate this process, let's take into consideration the table shown in Figure 3.3, that is about mountains. Through the STI it is possible to pinpoint the columns that contain entities, aka "Name" and "Range", and the ones that contain literal data, aka "Coordinates" and "Height".

Among the *NE-columns*, "Name" is the subject column, i.e. the column whose content, in this case a list of mountain's names, represents the main topic of the table itself. The content of type NE columns can be associated with the entities present in a KG: for example, "Mont Blanc" refers to <http://dbpedia.org/page/MontBlanc> while "Mont Blanc Massif" refers to [http://dbpedia.org/page/Mont\\_Blanc\\_massif](http://dbpedia.org/page/Mont_Blanc_massif). Each one of the *NE-column* gets associated with a knowledge base's class (e.g. "Name" is represented by <http://dbpedia.org/ontology/Mountain>) while each *L-column* is identified by a datatype (e.g. "Height" is a `xsd:integer`). Also the relations that occur between the subject column and each of the other ones are identified within the knowledge base: for example the relation that exists between the column "Name" and the column "Range", meaning between mountains and mountain ranges is defined by the property <http://dbpedia.org/ontology/mountainRange> and the one between "Name" and "Height" by <http://dbpedia.org/ontology/elevation>.

Consequently, the first row of the table can be represented using RDF triples in the following way:

```
<dbr:Mont_Blanc,georss:point,45.8336.865(xsd:string)>
<dbr:Mont_Blanc,dbo:elevation,4808(xsd:double)>
<dbr:Mont_Blanc,dbo:mountainRange,dbr:Mont_Blanc_massif>.
```

The annotation steps of the STI involve several key challenges [31]: (i) *disambiguation*, the entities' concepts described in a table are not known in advance, and the entities described may correspond to more than one concept in the KG. For example, the entity `Mont_Blanc`

in Figure 3.3 can refer to different entities associated with different concepts. As we might know, *Mont\_Blanc* is the name of a mountain, but, also, the name of a tunnel, a poem, and a dessert<sup>2</sup>; (ii) *homonym*, which is related to the presence of different entities with the same name and the same concept. In addition to the famous mountain located on the French-Italian border<sup>3</sup>, there is also a mountain with the same name on the Moon<sup>4</sup>; (iii) *matching*: the name of the entities in the table may be syntactically different from the name in a KG. *Johannisberg mountain* refers to the *Johannisberg (High Tauern)* entity in DBpedia<sup>5</sup>; (iv) *missing context*: it is often easier to extract the context from textual documents than from tables, due to the amount of content that has to be processed. Disambiguating possible meanings of literals is therefore more difficult.

In the last years, there has been a ton of works on STI which can be mainly classified as supervised (they exploit already annotated tables for training) [78, 101, 83, 50] or unsupervised (they do not require training data) [123, 88, 26, 80]; and as automatic [78, 83, 69] and semi-automatic [50]. Moreover, some approaches [59, 123, 88, 26, 80, 117, 109] largely focus on content analysis of Web tables, such as Web page title, table caption, or surrounding text, while others [78, 101, 83, 69, 100] address independent tables which can only rely on their own data.

We identify some limits of the state-of-the-art approaches as follows: i) they adopt lexical comparisons for matching, which ignore the contextual semantics; ii) they rely on metadata like column names and sometimes even external information like table description, both of which are often unavailable in real world applications; iii) they use personalised KG; iv) they perform only a few steps of STI.

To overcome such limitations, we propose a comprehensive approach and a tool, named *MantisTable*<sup>6</sup>, which provides an unsupervised method to annotate independent tables, possibly without a header row or other external information. Our experiments with T2Dv2 Gold Standard<sup>7</sup> from the general Web and Limaye Gold Standard [59] from Wikipedia pages have shown that our method is effective and can outperform the state-of-the-art approaches.

---

<sup>2</sup>[en.wikipedia.org/wiki/Mont\\_Blanc\\_\(disambiguation\)](https://en.wikipedia.org/wiki/Mont_Blanc_(disambiguation))

<sup>3</sup>[dbpedia.org/page/Mont\\_Blanc](https://dbpedia.org/page/Mont_Blanc)

<sup>4</sup>[dbpedia.org/page/Mont\\_Blanc\\_\(Moon\)](https://dbpedia.org/page/Mont_Blanc_(Moon))

<sup>5</sup>[dbpedia.org/page/Johannisberg\\_\(High\\_Tauern\)](https://dbpedia.org/page/Johannisberg_(High_Tauern))

<sup>6</sup>[mantistable.disco.unimib.it](https://mantistable.disco.unimib.it)

<sup>7</sup>[webdatacommons.org/webtables/goldstandardV2.html](https://webdatacommons.org/webtables/goldstandardV2.html)

## 3.1 Application Scenarios in Semantic Table Interpretation

The main application of STI can be discerned into the following activities:

- data search: the lack of semantic information in Web tables makes difficult to exploit the high quality data offered by such sources during a Web research. Thanks to the use of specific techniques for tables' semantic interpretation, it is possible to simplify the annotation process to make these data sources more accessible during users' searches;
- data enrichment: thanks to the identification of the semantic component of the elements in a table, it is possible to complete, extend and enrich the content in question with additional information or data from other sources;
- KG construction/KG population: the data contained in Web tables, when rightly associated to semantics, can be used to build knowledge bases or to expand already existing ones.

## 3.2 Approaches Supporting Semantic Table Interpretation

This section gives an overview of state-of-the art approaches on the STI. More than 50 papers collected from different sources have been studied in order to get an overview of the existing approaches for the semantic table interpretation.

For the analysis of the state-of-the-art, 18 approaches have been selected as representative because they are complete with respect to the workflow of STI and are thus considered the front line for comparison. To begin with, we give a short overview of the approaches and analyse them, considering 4 criteria as shown in Table 3.1. The first criterion, according to which we organised this section, is the learning technique: unsupervised and supervised. While the former category does not rely on labelled input data, the latter is only applicable for labelled data. In our analysis, the second criterion refers to the completeness with respect to the STI workflow. The third and the fourth criterion refers to the publication year and to the KG that these approaches use to annotate tables.

Table 3.1 Approaches supporting STI.

	Year	STI workflow									Learning Technique	KG
		Data Prep.	Column Type Analysis			Concept and Datatype Annotation			Predicate Ann.	Entity Link.		
			NE	L	S	NE	L	S				
Kruit et al. [52]	2019	-	✓	✓	✓	-	-	-	✓	✓	Unsup	DBpedia
Chen et al. [14]	2019	-	✓	-	-	✓	-	-	-	✓	Sup	DBpedia
Takeoka et al. [102]	2019	-	✓	✓	-	✓	✓	-	-	-	Sup	Wordnet
Zhang et al. [122]	2018	-	-	-	✓	-	-	✓	-	-	Sup	Wikipedia
Zhang et al. [123]	2017	-	✓	✓	✓	✓	✓	✓	✓	✓	Unsup	DBpedia
Efthymiou et al. [31]	2017	✓	✓	✓	✓	✓	✓	✓	✓	✓	Unsup	DBpedia
Pham et al. [78]	2016	-	✓	✓	-	✓	✓	-	✓	-	Sup	Domain independent
Taheryian et al. [101]	2016	-	✓	✓	-	✓	✓	-	✓	-	Sup	Domain independent
Ritze et al. [88]	2015	✓	✓	✓	-	✓	✓	-	✓	✓	Unsup	DBpedia
Ramnandan et al. [83]	2015	-	-	✓	-	-	✓	-	-	-	Sup	Domain independent
Deng et al. [26]	2013	-	✓	✓	-	✓	✓	-	-	-	Unsup	Freebase, Yago
Quercini et al. [80]	2013	✓	✓	✓	-	✓	✓	-	-	✓	Unsup	DBpedia
Mulwad et al. [69]	2013	✓	✓	✓	-	✓	✓	-	✓	✓	Unsup	DBpedia, Yago, Wikitology
Wang et al. [117]	2012	✓	✓	-	-	✓	-	-	✓	✓	Unsup	Enriched Probase
Knoblock et al. [50]	2012	-	✓	✓	-	✓	✓	-	✓	-	Sup	Domain independent
Venetis et al. [109]	2011	-	✓	✓	✓	✓	✓	✓	✓	-	Unsup	isA, relation database
Syed et al. [100]	2010	-	✓	✓	-	✓	✓	-	✓	✓	Unsup	Wikitology
Limaye et al. [59]	2010	-	✓	-	-	✓	-	-	✓	✓	Unsup	Yago
<b>MantisTable</b>	2019	✓*	✓*	✓*	✓**	✓**	✓*	✓**	✓**	✓	Unsup	DBpedia

\* refers to the improvement of the technique for the STI step

\*\* refers to the propose of novel techniques for the STI step

Approaches such as [78, 101, 83, 50, 14, 122, 102] use supervised machine learning techniques to label new sources of structured data with the support of training data which have been previously labelled manually. Such approaches use machine learning techniques in which a function of inference is created [101, 50, 14, 122, 102] or a classifier is trained with features corresponding to similarity metrics [78, 83]. Supervised approaches are sometimes more effective than the others, since a set of training data is enough to label new tables covering specific domains. However, in this regard, they can be limited for three reasons: (i) they require a set of training data that should be provided, (ii) such training set may not be available, thus forcing the user to create it by means of a manual labelling, and (iii) the

results are comparable to the ones achieved by unsupervised approaches with respect to the quality of the annotations. In order to avoid these limitations, MantisTable uses unsupervised learning techniques, so the approach can be applied to general purpose domains.

The remaining approaches apply unsupervised techniques. Approaches such as [69, 117, 109, 100] use custom or personalised KGs, while the rest use Open Source KGs available on the Web. In [117] the authors focus on Web tables and synthetically define the process of semantic interpretation of a table as finding its correct positioning within a taxonomy (hierarchical organisation of knowledge). Therefore, the approach tries to associate a specific table with one or more concepts contained in a KG. In particular, after establishing such associations, each row of the table describes the attributes of an entity, which has as 'type' a concept in a custom version of Probase (a general purpose taxonomy). For each concept, Probase returns a list of entities associated with a set of scores (plausibility and ambiguity). In order to select eligible entities of a class, authors score and merge candidate attributes and choose the top-ranked.

[100] makes custom queries to Wikitology (a hybrid knowledge base of structured and unstructured information extracted from Wikipedia, augmented by RDF data from DBpedia and other Linked Data resources) using the information inside the rows of a table. This approach infers automatically a (partial) semantic model for the tables using the information in the headings and the information stored in the table cells. Moreover it offers the possibility to export the data in the table as linked data.

[109] uses a majority-rule mechanism, where a potential concept that should be annotated to a specific column is selected because it is the one that occurs more often within the cells of the same column. A characteristic of this approach is that the search for the relevant semantic labels is performed directly on the Web rather than from a predefined KG. The subject column must not necessarily be a key to the table and may contain duplicate values. Moreover, it is possible that the subject column is represented by several columns of the same table. This principle can be ineffective in many aspects, for example the presence of general concepts (e.g. music) can be often combined with different textual content. Besides that, the consideration of a KG that does not have a hierarchical classification.

For the STI process, [69] uses queries and ranking metrics to generate an initial set of concepts, predicates and entities to be assigned as the headers of the columns, the content of the cells and the relationships between the columns. This approach uses the data in DBpedia, Yago and Wikitology. As for the columns, two sets of candidate concepts are generated (one for DBpedia concepts and the other for Yago concepts), which are formed by the union of the concepts retrieved from the queries performed with the values of the cells belonging to each column. The set of relationships to be considered when annotating the semantic

connections between the columns is obtained by the union of the set of candidate relations for each pair of cells within each row of the table. Unlike the above approaches, MantisTable uses Open Source KG, freely available on the Web. Moreover, querying KG is onerous and our approach reduces the number of queries, as it selects only a limited number of rows in a table. These rows are considered more significant as a result of ranking metrics. Finally, MantisTable generates context utilising the elements inside the table to discriminate the entities and to create high quality annotations.

Approaches such as [52, 123, 31, 88, 26, 80, 69, 117, 100] and [59] are unsupervised approaches that use Open Source KGs for the annotation.

[59] uses a comprehensive strategy for semantic annotation which examines the entire content of the table (for example, the classification of a column depends on all the cells in that column) through the application of a probabilistic graphic model. Such model has been identified by [69] as too expensive in terms of computational effort, thus the authors suggest an alternative Semantic Message Passing Algorithm that applies the same type of joint inference to a similar light-weight graphic model. Also, their approach takes into consideration semantic relations among columns as [59] does, but in contrast it takes into account both the headers of the columns and the entities within the rows. [123] uses a similar Semantic Message Passing Algorithm as [69]. Moreover, [123] includes steps such as the identification of the S-column, the L-column annotation, the analysis of a sample data extracted from the entire data source to reduce computational effort, etc. [31] proposes different unsupervised methods for matching entities of a table to entities of a KG on the Web. The similarity between entities is computed as the cosine distance between their vector representations. Such approaches work well with table contents alone, without relying on any metadata, but cells often lack entity correspondences, thus resulting in a decrease of their performance. The annotation process proposed by [88] uses similarity metrics for the creation of candidate concepts for the semantic annotation. Such candidate concepts are then sorted accordingly to the principle of weighted majority voting, where the weights are based on the matching scores, calculated between the values in the table and the entities in the KG associated with them. [26] offers a scalable and efficient solution for determining concepts associated to each NE-column within a table, using a MapReduce algorithm with two supporting techniques: knowledge concept aggregation and knowledge entity partition. These techniques allow the identification of top  $k$  candidate concepts for the NE-column of the table. In contrast to other approaches, fuzzy matching algorithms calculated between the values in the table and the entities of the KG, do not compromise the efficiency of the algorithm itself. The fuzzy matching and the ordering of the obtained results are based on generic similarity functions. Thus, such algorithm obtains better results, in particular with



respect to [109]. [80] proposes a mechanism for the annotation of cell value with entities that are not present in a KG, through Web searching.

Finally, the goal of [52] is to complete the KG with the information in the Web tables. The approach uses a Probabilistic Graphical Model which considers first the label similarity and then updates the likelihood score to maximise the coherence of entity assignments across the rows using Loopy Belief Propagation (LBP). Unlike other approaches, for entity matching the authors compute coherence as a combination of properties that are shared by the entities in the table and do not use class membership. If the label matching is not sufficient, the approach make use of embeddings of KG entities. This feature helps the approach to identify novel facts for KG completion. Similarly, MantisTable also uses Open Source KG. Anyhow, the performance of the MantisTable approach is compared to just some of the approaches in the state-of-the-art because (i) the code is not always available, and (ii) when available, it is difficult to be executed. Moreover, the embeddings cannot be always utilised as tables have often missing values.

Considering the annotation steps as in Table 3.1, [31] is the only approach that applies all the predefined annotation steps for the STI. Most of the approaches do not perform the Data Preparation step but only [31, 88, 80, 69] and [117] do. Most of the approaches do column type analysis and annotation for both NE and L columns. Even though the approach in [101] does not identify the S-column, it supports the annotation of the relations between columns. Differently from all the approaches, MantisTable performs the steps of STI proposing novel techniques in order to improve and provide high quality annotations.

Approaches such as [26, 117, 80, 109] mainly focus on the analysis of Web tables, thus limiting their range of actions to the content of Web pages. Such approaches use the information offered by Web tables (Web page title, table caption, or surrounding text, etc.) in the semantic annotation process, while approaches such as [123, 59, 100, 101, 83, 78, 69, 88] can rely on the data in the table. Moreover, considering only Web tables excludes the possibility of analysing data sources that contain a large number of tuples, thus ignoring the problem of performance and execution time. MantisTable does not consider only Web Tables but also other kinds of tables.

If in this section we have concentrated on comparing the approaches, in the next one we will propose the comparison of the tools.

## 3.3 Comparison of Tools

The purpose of the analysis presented in this section is to gather information on the current tools available for the STI process of a table or, more in general, of a structured data source.

### 3.3.1 DataGraft

DataGraft<sup>8</sup> [90] is a cloud-based service, which provides an integrated web environment for data hosting (linked data and file storage, dataset sharing, data querying) and table data transformations (interactively building, modifying, and sharing of repeatable and reusable data transformations). About semantic annotations, DataGraft integrates two tools: Grafterizer<sup>9</sup> and ASIA<sup>10</sup>; the first one is responsible for data cleaning and transformation, while the second one deals with the annotation itself. The interface's part that is devoted to operations on tables is made of three sections: "Tabular Transformation" in which the pipeline for data transformation gets defined (Figure 3.4a); "Tabular Annotation" where it is possible to manually associate the tables' columns with concepts and datatypes referring to a knowledge base and to indicate the relations that occur between them (Figure 3.4b); and "RDF Mapping", that allows us to manually create or automatically generate RDF triples based on the table notations (Figure 3.4c).

Moving to the tab "Tabular Annotation", it is possible to start the table's semantic annotation process; this operation requires manual intervention, but the tool provides support in the filling of required fields thanks to the integration of ABSTAT<sup>11</sup> [94], which provides suggestions about concepts and datatypes. For each column a dialog box allows the user to define it as L-column, and consequently associate a datatype, or as NE-column for which the user needs to identify a type. A section is also devoted to the definition of the relation that occurs between a selected column and another one that could be selected from a dropdown menu. The subject column is automatically established based on the inserted relations. An example of annotation is shown in Figure 3.5, where the column "CONQUERED\_ON" has been identified as a literal of type "date", and the relation with the column "MOUNTAIN" has been referred as <http://dbpedia.org/ontology/firstAscentYear> (Figure 3.5a), while the column "RANGE" has been identified as a URI of type <http://dbpedia.org/ontology/MountainRange>. The relation with the column "MOUNTAIN" has been referred as <http://dbpedia.org/ontology/mountainRange> (Figure 3.5b).

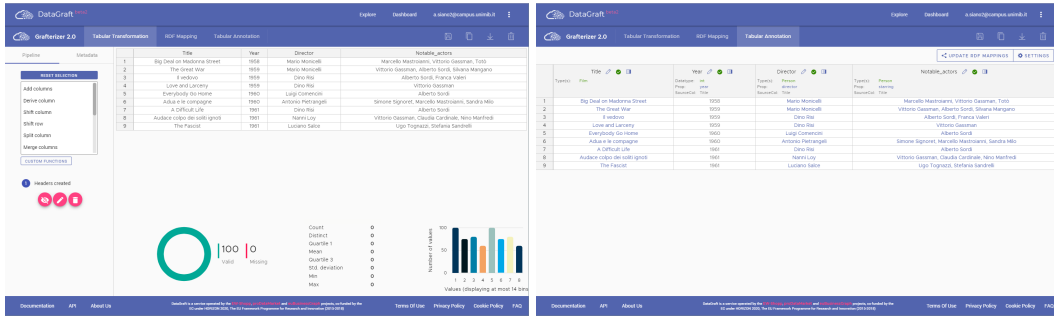
---

<sup>8</sup>[datagraft.io](http://datagraft.io)

<sup>9</sup>[eubusinessgraph.eu/grafterizer-2-0/](http://eubusinessgraph.eu/grafterizer-2-0/)

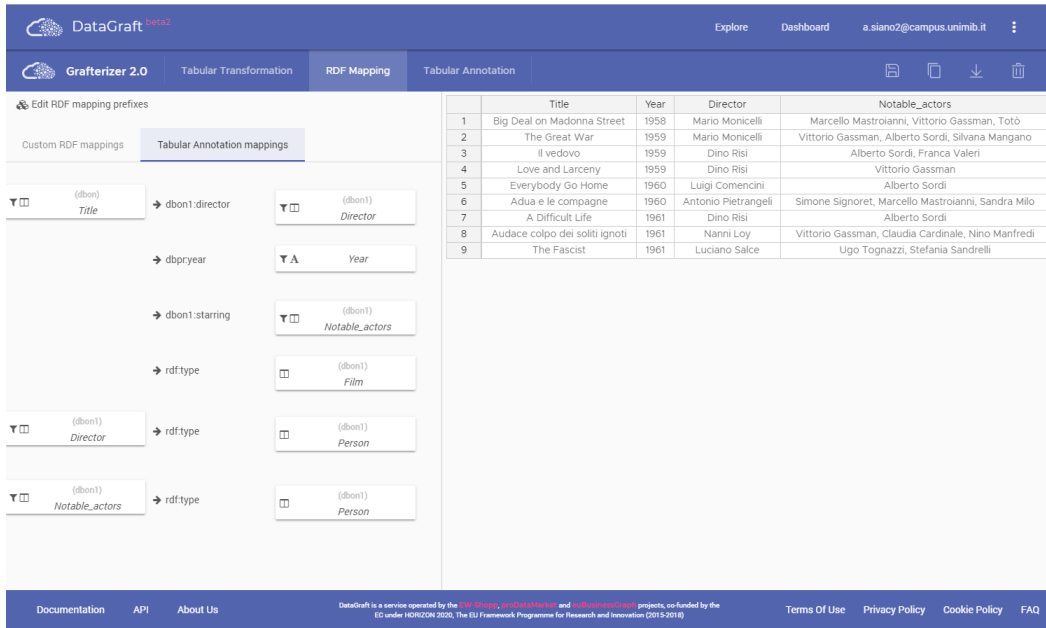
<sup>10</sup>[inside.disco.unimib.it/index.php/asia/](http://inside.disco.unimib.it/index.php/asia/) - [eubusinessgraph.eu/asia-2/](http://eubusinessgraph.eu/asia-2/)

<sup>11</sup>[abstat.disco.unimib.it](http://abstat.disco.unimib.it)



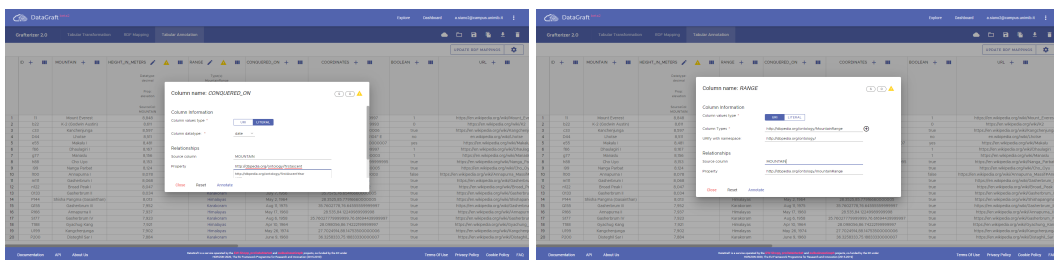
(a) Section “Tabular Transformation”.

(b) Section “Tabular Annotation”.



(c) Section “RDF mapping”.

Figure 3.4 DataGraft: interface devoted to operations on tables.



(a) L-column Annotation

(b) NE-column Annotation

Figure 3.5 DataGraft: columns’ annotation.

The annotation’s results can be exported in different formats such as RDF (N-Triples), CSV and JSON.

### 3.3.2 Karma

Karma<sup>12</sup> [42] is an Open Source information integration tool that allows users to integrate data coming from different sources such as databases, spreadsheets, delimited text files, XML, JSON, KML and Web API. The tool is available through an executable<sup>13</sup>, and it requires some configuration activities to run.

The graphical interface provides an easy way to transform data in order to normalise, restructure and express them in different forms. It is built to help users to integrate information by modelling it according to one or more ontologies. Karma learns to recognise the mapping of data to ontology classes and then uses the ontology to propose an automatically generated model that ties together these classes; this model can be refined by the user. Once the model is complete, the integrated data can be published as RDF or stored in a database.

accessionNumber	artist	birthDeath	creditLine	dimensions	title	nationality	materials	
26.1	Frishmuth, Harriet Whitney	1880-1980	Gift of the Friends of American Art	H: 61 in.	Joy of the Waters	American	bronze	http://w.waters.harriet
1991.345	Archipenko,	1887-1964	Gift of Frank C.	H: 12 3/4 in.	Concave or	American	bronze with	http://v

Figure 3.6 Karma: example of an empty model.

When the initial table is loaded, if a model for that specific source has never been defined, Karma will show an empty model represented by small red circles on top of each column (Figure 3.6). The mapping process is made of two phases: specifying the semantic types and specifying the relations between classes. The system shows some available suggestions regarding a column; the user can select one of those suggestions (Figure 3.7a) or click on the empty node to define a new class and subsequently on the label to define its property. For example, if a column contains names of artists, the user may want to set foaf:Person as a class and foaf:name as property. A new annotation can be identified by using a

<sup>12</sup>[usc-isi-i2.github.io/karma/](http://usc-isi-i2.github.io/karma/)

<sup>13</sup>[github.com/usc-isi-i2/Web-Karma/wiki/Installation%3A-One-Click-Install](https://github.com/usc-isi-i2/Web-Karma/wiki/Installation%3A-One-Click-Install)

specific window that will suggest all the available classes or properties categorised as “recommended”, “compatible” and “all” (Figure 3.7b).

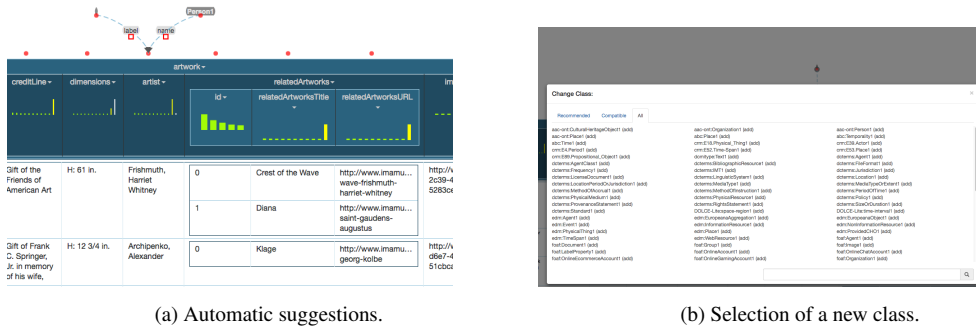


Figure 3.7 Karma: class annotation.

Once the semantic type of a certain column gets chosen by the user, Karma will learn the annotation and use it as a future suggestion in case of similar columns. Once the types of each column are defined, the user can determine the relations between them by following the same process previously described; it is also possible to drag an empty node on an existing class to activate a connection and build hierarchies; Figure 3.8 shows an example of a relation’s display.

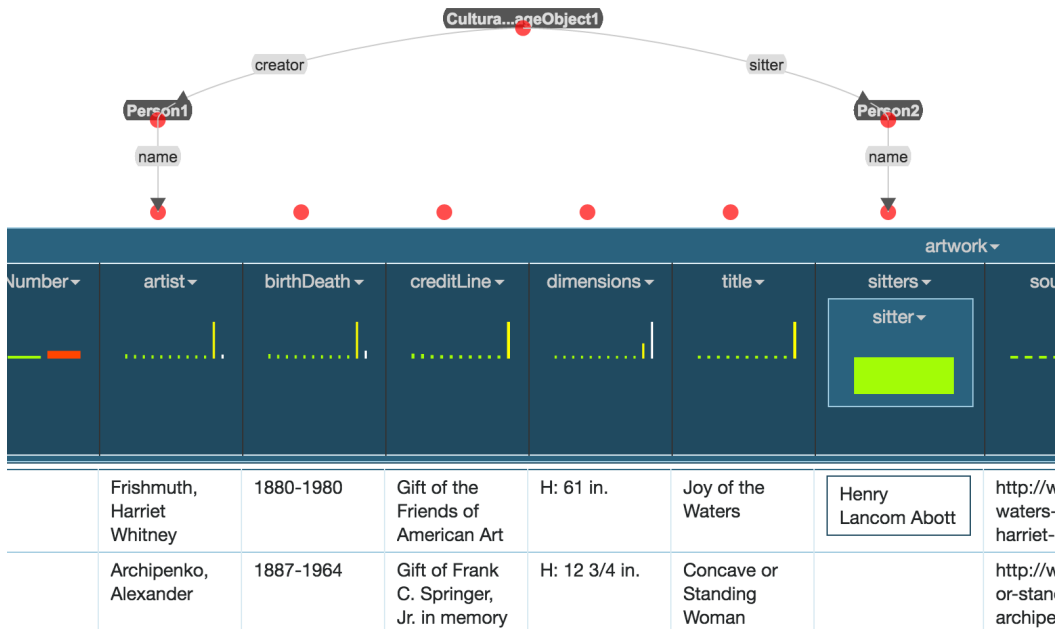


Figure 3.8 Karma: example of relation.

### 3.3.3 Odalic

Odalic<sup>14</sup> [49] is a tool for interpreting tabular data and publishing them as Linked Data. Odalic annotates columns using a KG, and links cell values to the entities of a KG. It takes as input CSV files and one or more KG. Odalic Server wraps the TableMiner+ algorithm [123]. Along the process users can provide feedback anytime since they can delete the suggested class/property for the annotation and suggest a new one, which is then added to the KG. Odalic allows users to manually specify multiple S-columns; supports any knowledge base accessible via SPARQL query language; and supports PoolParty KG<sup>15</sup>. The result of the STI process can be exported in several formats, including an extended CSV version and a RDF dataset.

The interface is made of different screens whose purpose is to allow the user to import new files or new Knowledge Bases, define new tasks, and view a summary of them in a dedicated page. Once a task has been defined and started, the semantic annotation process happens automatically; at the end, the user can view a page that shows the results of the classification of the columns, the disambiguation of the cells, the identification of the subject column and, finally, the identification of the relations between the columns. The user can modify the results to improve the final annotation.

The first tab (Figure 3.9) shows the results of column annotation and cells' reunion. The identified resources have different colours depending on their Knowledge Base (that can be viewed through tooltip); the links that we find next to each resource lead to their specific browser pages.

The header's tooltip of each column shows how they've been considered by the algorithm during the process:

- **Named Entity:** the column contains entities that are univocally identifiable through URIs from the Knowledge Base;
- **Non-named entity:** the column contains literals and therefore it had been ignored during the classification and disambiguation's process;
- **Ignored:** the column had been intentionally ignored because of a prior explicit request from the user;
- **Compulsory:** the column wasn't recognized as Named-Entity, but the user requested to include it in the process anyway.

---

<sup>14</sup>[github.com/odalic/](https://github.com/odalic/)

<sup>15</sup>[www.poolparty.biz](http://www.poolparty.biz)

Ort	PLZ	Bezirk	Bundesland
<ul style="list-style-type: none"> <li>Ort (dbpedia-owl:Settlement)</li> <li>Settlement (dbpedia-owl:Settlement)</li> <li>Spatial Thing (geo:SpatialThing)</li> <li>Concept (skos:Concept)</li> </ul>	<ul style="list-style-type: none"> <li>population total (dbpedia-owl:populationTotal)</li> <li>PLZ (dbpedia:PLZ)</li> <li>PLZ (http://de.dbpedia.org/property/plz)</li> </ul>	<ul style="list-style-type: none"> <li>Q 486972 (http://www.wikidata.org/entity/Q486972)</li> <li>Item (http://www.wikidata.org/ontology#Item)</li> <li>Place (schema.org:Place)</li> <li>Concept (skos:Concept)</li> </ul>	<ul style="list-style-type: none"> <li>Q 486972 (http://www.wikidata.org/entity/Q486972)</li> <li>Region (yago:Region10833985)</li> <li>Place (schema.org:Place)</li> <li>NUTS Region (http://ec.europa.eu/eurostat/ramon)</li> </ul>
<ul style="list-style-type: none"> <li>Linz (dbpedia:Linz)</li> <li>Linz (dbpedia:Linz)</li> <li>Linz (http://de.dbpedia.org/resource/Linz)</li> <li>J.J. Linz (http://linked.opendata.cz/ontology/domain/value/hv/kicove/Slovo/%20%20Linz)</li> </ul>	4040	<ul style="list-style-type: none"> <li>Linz-Stadt</li> <li>Stadt Linz@de (http://www.wikidata.org/entity/Q15848788)</li> <li>Kinderheim der Stadt Linz@de (http://de.dbpedia.org/resource/Liste_von_Kinderheimen_in_Osterreich)</li> </ul>	<ul style="list-style-type: none"> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (http://de.dbpedia.org/AT31 - Oberösterreich (http://ec.europa.eu/eurostat/ramon)</li> </ul>
<ul style="list-style-type: none"> <li>Linz (dbpedia:Linz)</li> <li>Linz (dbpedia:Linz)</li> <li>Linz (http://de.dbpedia.org/resource/Linz)</li> <li>J.J. Linz (http://linked.opendata.cz/ontology/domain/value/hv/kicove/Slovo/%20%20Linz)</li> </ul>	4020	<ul style="list-style-type: none"> <li>Linz-Stadt</li> <li>Stadt Linz@de (http://www.wikidata.org/entity/Q15848788)</li> <li>Kinderheim der Stadt Linz@de (http://de.dbpedia.org/resource/Liste_von_Kinderheimen_in_Osterreich)</li> </ul>	<ul style="list-style-type: none"> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (http://de.dbpedia.org/AT31 - Oberösterreich (http://ec.europa.eu/eurostat/ramon)</li> </ul>
<ul style="list-style-type: none"> <li>Leonding (dbpedia:Leonding)</li> <li>Leonding (dbpedia:Leonding)</li> <li>Leonding (http://de.dbpedia.org/resource/Leonding)</li> </ul>	4060	<ul style="list-style-type: none"> <li>Linz-Land</li> <li>Linz-Land District@en (dbpedia:Linz-Land-District)</li> <li>Bezirk Linz-Land@de (http://www.wikidata.org/entity/Q265632)</li> <li>Bezirk Linz-Land@de (http://de.dbpedia.org/resource/Bezirk_Linz-Land)</li> </ul>	<ul style="list-style-type: none"> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (http://de.dbpedia.org/AT31 - Oberösterreich (http://ec.europa.eu/eurostat/ramon)</li> </ul>
<ul style="list-style-type: none"> <li>Gallneukirchen (dbpedia:Gallneukirchen)</li> <li>Leonding (dbpedia:Leonding)</li> <li>Gallneukirchen (http://de.dbpedia.org/resource/Gallneukirchen)</li> </ul>	4210	<ul style="list-style-type: none"> <li>Urfahr-Umgebung</li> <li>Urfahr-Umgebung District@en (dbpedia:Urfahr-Umgebung-District)</li> <li>Bezirk Urfahr-Umgebung@de (http://www.wikidata.org/entity/Q253585)</li> <li>Bezirk Urfahr-Umgebung@de (http://de.dbpedia.org/resource/Bezirk_Urfahr-Umgebung)</li> </ul>	<ul style="list-style-type: none"> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (http://de.dbpedia.org/AT31 - Oberösterreich (http://ec.europa.eu/eurostat/ramon)</li> </ul>
<ul style="list-style-type: none"> <li>Wels (dbpedia:Wels)</li> <li>Wels (dbpedia:Wels)</li> <li>Wels (http://de.dbpedia.org/resource/Wels_(Stadt))</li> <li>wels (http://linked.opendata.cz/ontology/domain/value/hv/kicove/Slovo/wels)</li> </ul>	4600	<ul style="list-style-type: none"> <li>Wels-Stadt</li> <li>Kategorie:Wels (Stadt)@de (http://www.wikidata.org/entity/Q1722208)</li> <li>Wels (Stadt)@de (http://de.dbpedia.org/resource/Wels_(Stadt))</li> </ul>	<ul style="list-style-type: none"> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (dbpedia:Upper_Austria)</li> <li>Oberösterreich (http://de.dbpedia.org/AT31 - Oberösterreich (http://ec.europa.eu/eurostat/ramon)</li> </ul>

Figure 3.9 Odlalic: classification e disambiguation's results summary.

The user can modify the results by deleting the annotations that he considers incorrect or non-significant; additionally, he can suggest a new disambiguation of a cell or a new column's classification that will be added to the knowledge base, or research a different annotation between the ones already listed in a dialog box that shows more information compared to the usual table view (Figure 3.10).

**Subject columns.** The second tab allows us to edit the subject column in case of wrong detection, since selecting the correct column is vital to pinpoint the right subject of the relations we found.

**Relations.** The third tab enable us to revise the relations found between the subject column and the other ones. Such relations are shown as arcs, in a graph whose nodes represent the table's columns (Figure 3.11). The graph can be explored in two modes: "Node Dragging", that allows to drag the vertices on the screen to change their position, and "Link Creation" that allows to add new arches and define new relations that can be selected through a dedicated modal window.

When the review process is done, the algorithm must be rerun; the adjustments made will be used as constraints. Then, the process' results can be exported anytime by clicking

Figure 3.10 Odalic: cell disambiguation.

a specific button, but only the output of the last performance will be considered (if some manual changes were made but the algorithm wasn't rerun, those changes will be ignored). The annotation can be exported as RDF Turtle, JSON-LD or as extended CSV, in which the initial columns and the additional ones that contain the annotations are reported.

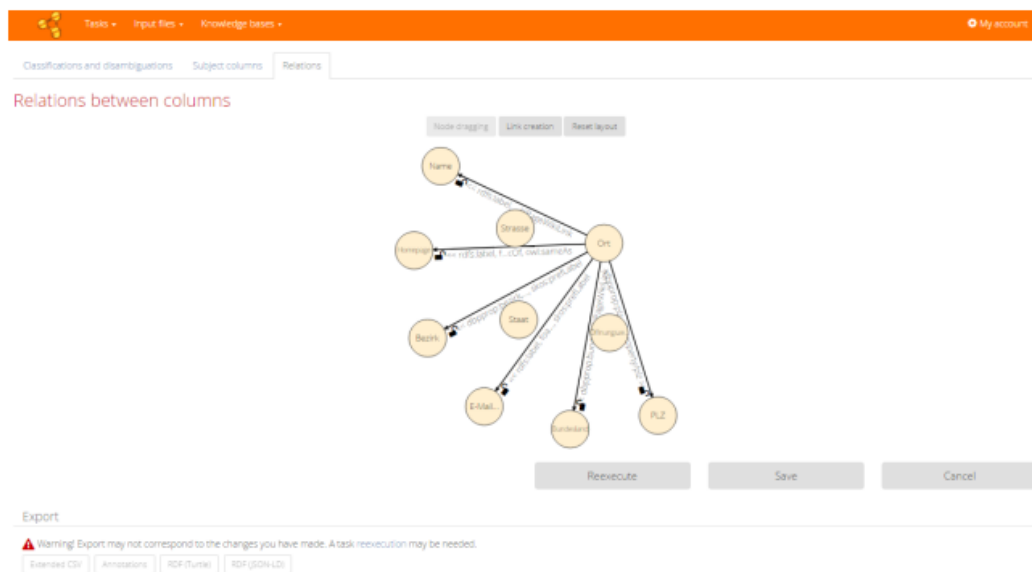


Figure 3.11 Odalic: graph that represents the relations between columns.



### 3.3.4 OpenRefine

OpenRefine<sup>16</sup> is a tool for cleaning, transforming and extending messy data. It can perform a semi-automatic reconciliation process against any database that exposes a web service using Reconciliation Service API<sup>17</sup> specification or a SPARQL endpoint. If a cell has multiple entity candidates the user needs to pick manually the correct one. To improve the quality of the matches, a class for the rows can be selected in order to restrict the matches, only to items which are instances of any subclass of the given class. In addition, the reconciliation interface can be configured to take into account other columns in the dataset in the matching scores. OpenRefine functionalities can be extended by installing extensions; moreover there are other distributions of the tool that have been customized for a specific usage or integration with other technologies (e.g. LOD refine).

OpenRefine supports a huge range of input formats including CSV, Excel (.xls, .xlsx), Google Spreadsheets, XML, RDF/XML, RDF N3 e JSON.

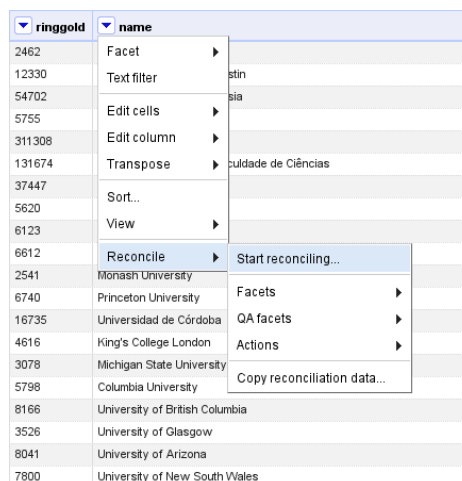


Figure 3.12 OpenRefine: reconciliation.

To execute the cells' reconciliation, we need to select the dropdown menu located in the header of the targeted column and click "Reconcile > Start reconciling" (Figure 3.12); this operation will open a dialog box where we can select the service of reconciliation we want (Wikidata is the default, to use others we're required to add them first). These settings define the minimal configuration to launch the process, but the tool provides more of them to make the results as accurate as possible (Figure 3.13).

The first additional configuration that can be chosen is the type of data that we need to analyze: for example, if the column in question is about university, restricting the search to the

<sup>16</sup>[openrefine.org](http://openrefine.org)

<sup>17</sup>[github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API](https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API)

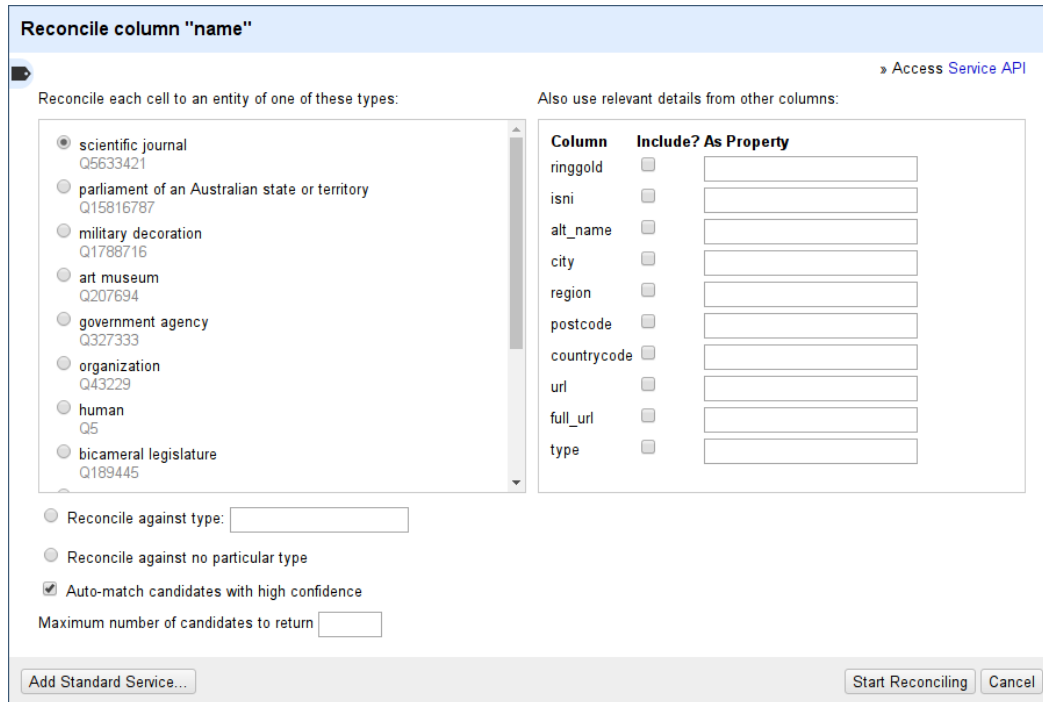


Figure 3.13 OpenRefine: additional settings to improve reconciliation.

type “university” will avoid the incorrect selection of <https://www.wikidata.org/wiki/Q2966> (which represents the city of Heidelberg) as a possible candidate for “University of Heidelberg”. The tool provides some possible suggestions identified on the basis of the dataset’s first elements: if the correct type is listed among these suggestions it can be selected right away, otherwise it’s possible to manually add another one.

A second parameter that can be set is about the addition of other columns of the dataset to solve ambiguity issues: for example, if we want to run the reconciliation of people of the sporting world and one of the table’s columns says which sport they practice, we can configure OpenRefine so that it will take into consideration this extra column.

After configuration, we can launch the process: once it finishes it will be possible to view the reconciliation results directly in the cells involved (Figure 3.14) and two situations can occur:

- inside the cell, a single result will be shown as a dark blue link; in this case the reconciliation was successful and the match is correct; there is no need for further interventions from the user;
- inside the cell there are multiple candidates, represented by lighter blue links; in this case we need to select one manually; the choice can be only applied to the single cell, or to all the ones still not reconciled for good that share the same content.

city	country
Oxford	GB
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Oxford (100)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Oxford (71)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Oxford (71)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Create new topic	
Search for match	
Paris	FR
Choose new match	
Geneva	CH
Choose new match	
Cambridge	GB
Choose new match	
Cambridge	US
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Cambridge (100)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Cambridge (100)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Cambridge (100)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Cambridge (71)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Cambridge (71)	
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Create new topic	
Search for match	

Figure 3.14 OpenRefine: the result of reconciliation.

### 3.3.5 STAN

STAN is an online tool that semantically annotates tables from popular KGs. Like DataGraft, it uses a self-completion API offered by ABSTAT [94], for the properties and concepts contained in a KG. It considers CSV files as input. STAN proposes an introductory page where the user can launch a new annotation or, if present, retrieve the last file he was working on; this latter operation is possible even without a registration process thanks to the use of specific cookies. When starting a new process, the first fundamental step is to import the table that we want to annotate; STAN supports the CSV format for which it is necessary to provide the separator that identifies the columns, if the table has a header or not and, eventually, a delimiter character from the text if present (Figure 3.15).

Once we complete the upload, the table will be shown. The annotation must be executed manually but, as DataGraft, the user gets offered support to correctly fill in the required fields thanks to an API of auto-completion offered by ABSTAT. By clicking on the header's label of each column a dialog box will pop up, which allows the annotation of the column in question:

- if it's the subject column, it must be written in the appropriate field and we will have to specify the semantic type that needs to be associated with it; if this type refers to DBpedia, an auto-completion system helps the user to fill in the form (Figure 3.16);
- If we're dealing with a column of any other kind, it will be necessary to provide the annotation as a property that will be automatically associated with the *subject column*

previously defined; even in this case, the user is helped by the auto-completion service that suggests the possible property that could be assigned to the column in question. Furthermore, it is also possible to define the type of object (datatype or ontological concept) that should be assigned to the column's values (Figure 3.17).

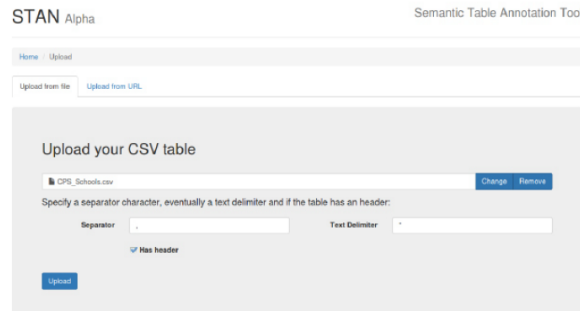


Figure 3.15 STAN: the upload of a new table.

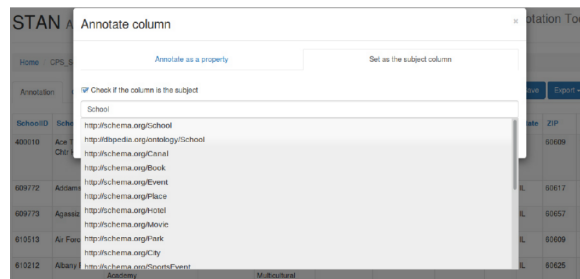


Figure 3.16 STAN: the selection of the subject column's annotation.

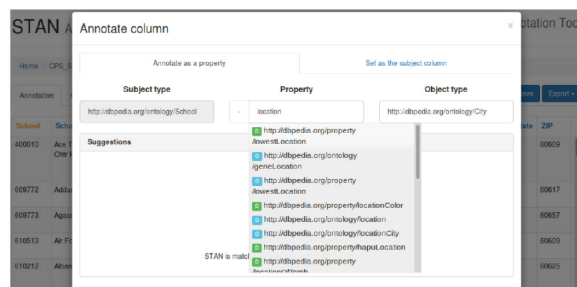


Figure 3.17 STAN: column annotation.

The uniqueness of the tool is the ability to define a personal ontology during the annotation process, without referring to a known KB. After the annotation, we can export the mapping in RML format and the table's data in RDF triples.

### 3.3.6 TableMiner+

The TableMiner+ [65] tool refers to the homonymous approach [123]. It only supports Web tables for which the user must provide an URL. The whole process is executed in batches and users can be notified with an email message when it finishes. The annotation is provided in JSON format which is interpreted and displayed using an annotated table and a graph visualisation module. Both visualisation components are interactive to allow users feedback and an output customisation. It was not possible to test TableMiner+ because, although the code for the tool is available, it was not possible to install it because of inter-dependencies in the code. The following analysis was conducted from the description provided in [65].

In input we must provide the URL of a web page in which there is at least one HTML table: then, a preview of the identified tables that contain relational data appears; the user can select which column he wants to annotate (Figure 3.18). Because of the length of the annotation process, we are given the option to enter an email address to be notified when the procedure ends.

Title	Year	Director	Notable actors
<i>Big Deal on Madonna Street</i>	1958	Mario Monicelli	Marcello Mastroianni, Vittorio Gassman, Toto
<i>The Great War</i>	1959	Mario Monicelli	Vittorio Gassman, Alberto Sordi, Silvana Mangano
<i>Il vedovo</i>	1959	Dino Risì	Alberto Sordi, Franca Valeri
<i>Love and Larceny</i>	1959	Dino Risì	Vittorio Gassman
<i>Everybody Go Home</i>	1960	Luigi Comencini	Alberto Sordi
<i>Adua e le compagne</i>	1960	Antonio Pietrangeli	Simone Signoret, Marcello Mastroianni, Sandra Milo
<i>A Difficult Life</i>	1961	Dino Risì	Alberto Sordi
<i>Audace colpo dei soliti ignoti</i>	1961	Nanni Loy	Vittorio Gassman, Claudia Cardinale, Nino Manfredi
<i>The Fascist</i>	1961	Luciano Salce	Ugo Tognazzi, Stefania Sandrelli
<i>Divorce, Italian Style</i>	1962	Pietro Germi	Marcello Mastroianni, Stefania Sandrelli
<i>Boccaccio '70</i>	1962	Mario Monicelli, Federico Fellini, Luchino Visconti, Vittorio De Sica	

Figure 3.18 TableMiner+: adding an URL and the selection of a specific table.

The outcome of annotation processing is a JSON file that gets interpreted and visualised in the two main components of the tool: an interactive annotated table and a graph visualisation module.

**The interactive annotated table.** The annotated table is the first point of interaction with the user; it shows the original annotated table with the identified entities, concepts and relations. The headers of each column present a set of concept candidates that describe the headers and the column's data in the most accurate way; each concept has a score that states

the system's confidence value; such concept candidates can be view through a dropdown menu and are additionally defined by a colour-code: those with a higher score are highlighted in green, while the ones with the lowest score are shown in red. By default the concept with the highest score gets selected as the winning one and put directly under the column's name (Figure 3.19-B). The user can, at any time, change the winning concept by selecting another among the listed ones.

Q	Title	Year	Director	Notable actors
	Italian Films 1.66		Film Director 3.7	Actor 2.88
Q	Big Deal on Madonna Street <a href="http://dbpedia.org/resource/Big_Deal_on_Madonna_Street">http://dbpedia.org/resource/Big_Deal_on_Madonna_Street</a> , 2.06	1958	Film Maker 2.99 Yago Legal Actor 2.81	Marcello Mastroianni <a href="http://dbpedia.org/resource/Marcello_Mastroianni">http://dbpedia.org/resource/Marcello_Mastroianni</a> , 2.39
Q	The Great War <a href="http://dbpedia.org/resource/The_Great_War">http://dbpedia.org/resource/The_Great_War</a> , 1.60	1959	Organism 2.38 Producer 2.38	Vittorio Gassman <a href="http://dbpedia.org/resource/Vittorio_Gassman">http://dbpedia.org/resource/Vittorio_Gassman</a> , 2.19
Q	Il vedovo <a href="http://dbpedia.org/resource/Il_vedovo">http://dbpedia.org/resource/Il_vedovo</a> , 2.56	1959	Physical Entity 2.38 Natural Person 2.38	Alberto Sordi <a href="http://dbpedia.org/resource/Alberto_Sordi">http://dbpedia.org/resource/Alberto_Sordi</a> , 2.25
Q	Love and Larceny	1959	Whole 2.38 Creator 2.38	Vittorio Gassman <a href="http://dbpedia.org/resource/Vittorio_Gassman">http://dbpedia.org/resource/Vittorio_Gassman</a> , 2.19
Q	Everybody Go Home <a href="http://dbpedia.org/resource/Everybody_Go_Home">http://dbpedia.org/resource/Everybody_Go_Home</a> , 2.27	1960	Italian Atheists 2.06	Alberto Sordi <a href="http://dbpedia.org/resource/Alberto_Sordi">http://dbpedia.org/resource/Alberto_Sordi</a> , 2.25
Q	Adua e le compagne	1960	Italian Screenwriters 1.58 Film Directors Who Committed Suicide 1.09 Screenwriter 1.08	Simone Signoret <a href="http://dbpedia.org/resource/Simone_Signoret">http://dbpedia.org/resource/Simone_Signoret</a> , 2.65
Q	A Difficult Life <a href="http://dbpedia.org/resource/A_Difficult_Life">http://dbpedia.org/resource/A_Difficult_Life</a> , 2.32	1961	People From Viareggio 0.66 People From Sal C 0.25	Alberto Sordi <a href="http://dbpedia.org/resource/Alberto_Sordi">http://dbpedia.org/resource/Alberto_Sordi</a> , 2.25
Q	Audace colpo dei soliti ignoti <a href="http://dbpedia.org/resource/Audace_colpo_dei_soliti_ignoti">http://dbpedia.org/resource/Audace_colpo_dei_soliti_ignoti</a> , 1.91	1961	<a href="http://dbpedia.org/resource/Nanni_Loy">http://dbpedia.org/resource/Nanni_Loy</a> , 2.29	Vittorio Gassman <a href="http://dbpedia.org/resource/Vittorio_Gassman">http://dbpedia.org/resource/Vittorio_Gassman</a> , 2.19
Q	The Fascist <a href="http://dbpedia.org/resource/The_Fascist">http://dbpedia.org/resource/The_Fascist</a> , 2.53	1961	Luciano Salce <a href="http://dbpedia.org/resource/Luciano_Salce">http://dbpedia.org/resource/Luciano_Salce</a> , 2.16	Ugo Tognazzi <a href="http://dbpedia.org/resource/Ugo_Tognazzi">http://dbpedia.org/resource/Ugo_Tognazzi</a> , 2.49
Q	Divorce, Italian Style <a href="http://dbpedia.org/resource/Divorce_Italian_Style">http://dbpedia.org/resource/Divorce_Italian_Style</a> , 1.86	1962	Pietro Germi <a href="http://dbpedia.org/resource/Pietro_Germi">http://dbpedia.org/resource/Pietro_Germi</a> , 2.19	Marcello Mastroianni <a href="http://dbpedia.org/resource/Marcello_Mastroianni">http://dbpedia.org/resource/Marcello_Mastroianni</a> , 2.39
Q	Boccaccio '70	1962	Mario Monicelli <a href="http://dbpedia.org/resource/Mario_Monicelli">http://dbpedia.org/resource/Mario_Monicelli</a> , 2.60	
Q	The Easy Life <a href="http://dbpedia.org/resource/The_Easy_Life">http://dbpedia.org/resource/The_Easy_Life</a> , 1.97	1962	Dino Risi <a href="http://dbpedia.org/resource/Dino_Risi">http://dbpedia.org/resource/Dino_Risi</a> , 2.73	Vittorio Gassman <a href="http://dbpedia.org/resource/Vittorio_Gassman">http://dbpedia.org/resource/Vittorio_Gassman</a> , 2.19
Q	The Last Judgement	1962	Vittorio De Sica <a href="http://dbpedia.org/resource/Vittorio_De_Sica">http://dbpedia.org/resource/Vittorio_De_Sica</a> , 2.43	Vittorio Gassman <a href="http://dbpedia.org/resource/Vittorio_Gassman">http://dbpedia.org/resource/Vittorio_Gassman</a> , 2.19
Q	Mafioso	1962	Alberto Lattuada <a href="http://dbpedia.org/resource/Alberto_Lattuada">http://dbpedia.org/resource/Alberto_Lattuada</a> , 2.46	Alberto Sordi <a href="http://dbpedia.org/resource/Alberto_Sordi">http://dbpedia.org/resource/Alberto_Sordi</a> , 2.25
Q	March on Rome	1962	Dino Risi <a href="http://dbpedia.org/resource/Dino_Risi">http://dbpedia.org/resource/Dino_Risi</a> , 2.73	Vittorio Gassman <a href="http://dbpedia.org/resource/Vittorio_Gassman">http://dbpedia.org/resource/Vittorio_Gassman</a> , 2.19
Q	The Conjugal Bed	1963	Marco Ferreri <a href="http://dbpedia.org/resource/Marco_Ferri">http://dbpedia.org/resource/Marco_Ferri</a> , 2.56	Ugo Tognazzi <a href="http://dbpedia.org/resource/Ugo_Tognazzi">http://dbpedia.org/resource/Ugo_Tognazzi</a> , 2.49
Q	I mostri <a href="http://dbpedia.org/resource/I_mostri">http://dbpedia.org/resource/I_mostri</a> , 2.54	1963	Dino Risi <a href="http://dbpedia.org/resource/Dino_Risi">http://dbpedia.org/resource/Dino_Risi</a> , 2.73	Vittorio Gassman <a href="http://dbpedia.org/resource/Vittorio_Gassman">http://dbpedia.org/resource/Vittorio_Gassman</a> , 2.19
Q	Alta infedeltà	1965	Mario Monicelli <a href="http://dbpedia.org/resource/Mario_Monicelli">http://dbpedia.org/resource/Mario_Monicelli</a> , 2.60	Ugo Tognazzi <a href="http://dbpedia.org/resource/Ugo_Tognazzi">http://dbpedia.org/resource/Ugo_Tognazzi</a> , 2.49
Q	Il diavolo <a href="http://dbpedia.org/resource/Il_diavolo">http://dbpedia.org/resource/Il_diavolo</a> , 2.54	1963	Gian Luigi Polidoro <a href="http://dbpedia.org/resource/Gian_Luigi_Polidoro">http://dbpedia.org/resource/Gian_Luigi_Polidoro</a> , 2.12	Alberto Sordi <a href="http://dbpedia.org/resource/Alberto_Sordi">http://dbpedia.org/resource/Alberto_Sordi</a> , 2.25
Q	Il Boom <a href="http://dbpedia.org/resource/Il_Boom">http://dbpedia.org/resource/Il_Boom</a> , 2.67	1963	Vittorio De Sica <a href="http://dbpedia.org/resource/Vittorio_De_Sica">http://dbpedia.org/resource/Vittorio_De_Sica</a> , 2.43	Alberto Sordi <a href="http://dbpedia.org/resource/Alberto_Sordi">http://dbpedia.org/resource/Alberto_Sordi</a> , 2.25
Q	Yesterday, Today and Tomorrow	1963	Vittorio De Sica <a href="http://dbpedia.org/resource/Vittorio_De_Sica">http://dbpedia.org/resource/Vittorio_De_Sica</a> , 2.43	Marcello Mastroianni <a href="http://dbpedia.org/resource/Marcello_Mastroianni">http://dbpedia.org/resource/Marcello_Mastroianni</a> , 2.39

Figure 3.19 TableMiner+: interactive annotated table.

About the cells, inside each one we can view, when found, the corresponding entity in the knowledge base (Figure 3.19-A); in case of wrong or missing annotation, the user can double click on the cell itself and provide an URI (Figure 3.20).

Q	Divorce, Italian Style <a href="http://dbpedia.org/resource/Divorce_Italian_Style">http://dbpedia.org/resource/Divorce_Italian_Style</a> , 1.86	Q	Divorce, Italian Style <a href="http://dbpedia.org/resource/Divorce_Italian_Style">http://dbpedia.org/resource/Divorce_Italian_Style</a> , 1.86
Q	Boccaccio '70	Q	Boccaccio '70 <a href="https://en.wikipedia.org/wiki/Boccaccio_%2770">https://en.wikipedia.org/wiki/Boccaccio_%2770</a>
Q	The Easy Life <a href="http://dbpedia.org/resource/The_Easy_Life">http://dbpedia.org/resource/The_Easy_Life</a> , 1.97	Q	The Easy Life <a href="http://dbpedia.org/resource/The_Easy_Life">http://dbpedia.org/resource/The_Easy_Life</a> , 1.97

Figure 3.20 TableMiner+: the edit of a cell's associated entity.

**Graph visualisation.** By clicking the button “inspect” (the magnifying glass) we have access to the visualisation through graph. This kind of visualisation, referring to the header line, shows the headers as numbered nodes and the candidate concepts as nodes linked to

them, where the most relevant class is shown as a solid and coloured line, while all the other classes are shown as dotted lines (Figure 3.21); right-clicking on one of the dotted lines allows us to choose that class as a winner for the corresponding node. In this mode, it is also possible to visually isolate a certain node and its connections by simply clicking on them: all the non-connected arcs and nodes will be made semi-transparent.

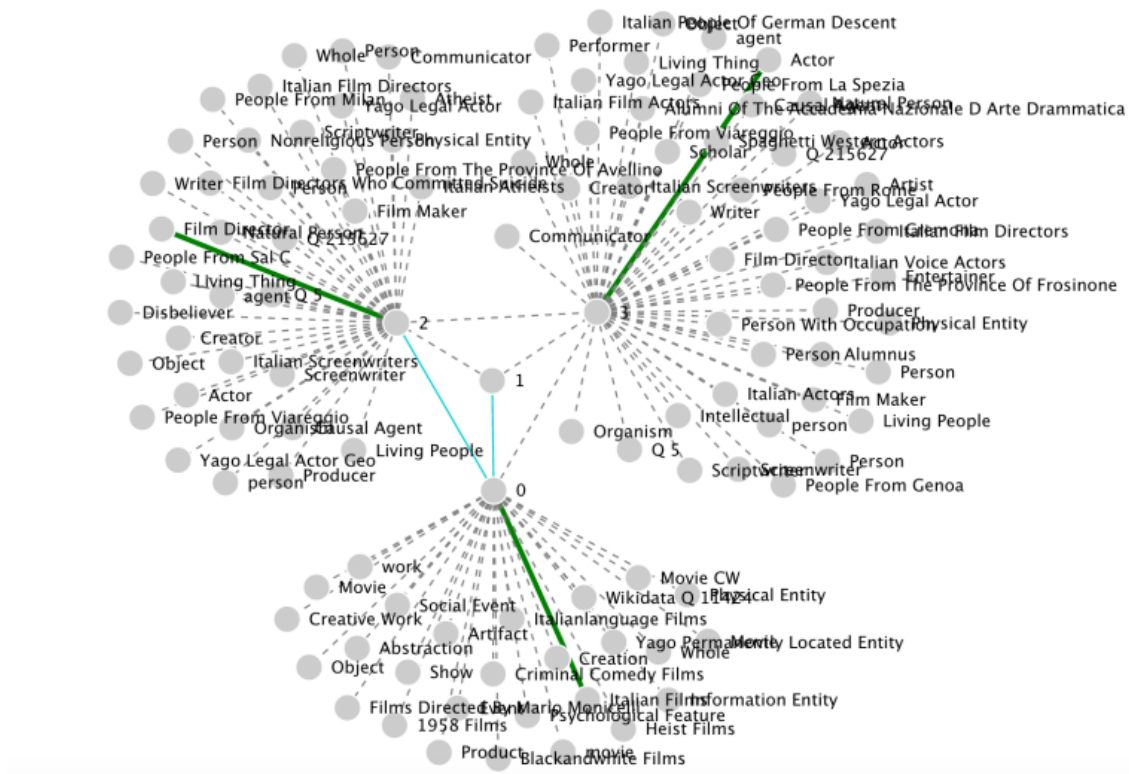


Figure 3.21 TableMiner+: graph of concepts and relations.

The numbered nodes, in addition to being connected with the knowledge base's classes, are also connected between themselves by lines that represent a relation: if they are indicative, the lines will be dashed, if, instead, TableMiner+ is able to establish a valid relation between the two columns, the line will be shown as solid and coloured.

The graph can be also visualised with single rows of the table and not only with the header.

### 3.3.7 Final comparison

Table 3.2 provides a comparison of the described tools on the basis of 7 criteria: the availability of the tool; the possibility to import the ontology; the annotation method; the availability of an auto-completion support for manual annotation; the STI tasks; and the possibility to export the results.

Regarding the possibility of defining an ontology to support the STI tasks, only Karma, Odalic and Datagraft allow users to import one or more and combine them, while the others only use predefined KG. In order to prepare data, Karma, OpenRefine and Datagraft give the users the ability to manipulate tables and refine them by allowing the modification of columns, such as renaming, deleting, or exchanging order. Moreover OpenRefine has peculiar features, such as the automatic creation of new columns, the exploration of the cells through facets that allows the comparison of values on the basis of a chosen constraint, and the use of clustering to consider groups of cells.

While for the STI tasks TableMiner+ and Odalic are based on an automatic process, Karma and Open Refine require user interaction. Datagraft and Stan, on the other hand, support only manual annotation although they provide an auto-complete service using ABSTAT. In STAN, if the class which is being examined refers to a class in DBpedia, uses the auto-completion feature both for the annotation of the S-column and properties of the other columns. All tools, except OpenRefine provide an annotation for the S-column, the NE columns, the L columns and the relationships between them. Karma, Odalic and TableMiner+ offer a list of suggestions for the correct semantic annotation that users can select in order to adjust the tool output. The linking of the cells with specific entities within the KG is performed by TableMiner+, Odalic, OperRefine and Datagraft. Furthermore, Tableminer+, Odalic and Karma offers a graphical representation of the semantic mapping. Apart from TableMiner+, all the other tools have the automatic saving feature.

Regarding the export of the mappings, Karma and Odalic allow the export in RDF format or in JSON-LD while STAN in RML format. In order to export the tabular data Karma uses the R2RM format, which allows to highlight the association between table and the ontology. STAN converts the tabular data into RDF triples while OpenRefine into JSON, YAML, RDF, and others.

Finally, an interesting feature is the use of APIs which allows the integration of external services. These services are present in Karma, OpenRefine and in particular in STAN, they allow two different annotation services, publicly accessible through HTTP GET and POST operations.

In the next Chapter the MantisTable approach will be described in detail, taking into account the attributes present in the Table 3.2.



Table 3.2 Tool comparison.

Tool	Working and Available	Ontology import	Annotation method	Auto-complete support (if manual)	Annotations				Entity Link.	Export
					Col.			Pred.		
					S	NE	L			
Datagraft	✓ (online)	✓	manual	✓ with ABSTAT	✓	✓	✓	✓	-	-
Karma	✓ (installer)	✓	semi-auto	-	✓	✓	✓	✓	-	-
Odalic	✓ (docker)	✓	auto	-	✓	✓	✓	✓	✓	✓
Open Refine	✓ (exe)	-	semi-auto	-	-	-	-	-	✓	✓
Stan	✓ (online)	-	manual	✓ with DBpedia	✓	✓	✓	✓	-	✓
TableMiner+	-	-	auto	-	✓	✓	✓	✓	✓	?
MantisTable	✓ (online)	-	auto	-	✓	✓	✓	✓	✓	-



# Chapter 4

## New Semantic Table Interpretation Approach

### 4.1 MantisTable approach

In this section we describe in detail the MantisTable approach which consists of the following phases:

0. **Data Preparation**, which aims to prepare the data inside the table;
1. **Column Analysis**, whose tasks are the semantic classification that assigns types to columns (NE-column or L-column), and the detection of the subject column (S-column);
2. **Concept and Datatype Annotation**, which deals with mappings between columns (or headers, if they are available) and semantic elements (concepts or datatypes) in a KG;
3. **Predicate Annotation**, whose task is to find relations, in the form of predicates, between the main column and the other columns to set the overall meaning of the table;
4. **Entity Linking**, which deals with mappings between cells and entities in a KG.

To describe each phase of the STI approach we consider Table 4.1, which lists the highest peaks in the world with additional information, such as heights, coordinates, etc. The table has been extracted from the T2D Gold Standard<sup>1</sup>, and extended by adding new columns (i.e.

---

<sup>1</sup>T2D table index: 14311244\_0\_7604843865524657408 - webdatacommons.org/webtables/goldstandardV2.html

COORDINATES, URL, DESCRIPTION, TEMPERATURE and a column with booleans) in order to demonstrate each phase of the approach<sup>2</sup>.

Table 4.1 The table reports a list of the highest peaks in the world from T2D Gold Standard, extended with other columns to consider a variety of situations (grey columns).

ID	PEAK	HEIGHT	RANGE	CONQUERED ON	COORD.	B	URL	DESCRIPTION	T
11	Mount Everest	8,848 km	Himalayas	May 29, 1953	27.98785, 86.92502	1	<a href="https://en.wikipedia.org/wiki/Mount_Everest">https://en.wikipedia.org/wiki/Mount_Everest</a>	Mount Everest, known in Nepali as Sagarmāthā and in Tibetan as Chomolungma, is Earth's highest mountain above sea level [...]	-35C
b22	K-2	8,611 m	Karakoram	July 31, 1954	35.87998, 76.51510	0	<a href="https://en.wikipedia.org/wiki/K2">https://en.wikipedia.org/wiki/K2</a>	K2, also known as Mount Godwin-Austen or Chhogori, is the second highest mountain in the world, after Mount Everest (8,848 metres), at 8,611 metres (28,251 ft).	-30C
c33	Kanchenjunga	8,597 km	Himalayas	Wednesday, 25 May 1955	27.70249, 88.14753	true	<a href="https://en.wikipedia.org/wiki/Kanchenjunga">https://en.wikipedia.org/wiki/Kanchenjunga</a>	Kangchenjunga, also spelled Kanchenjunga, is the third highest mountain in the world, and lies partly in Nepal and partly in Sikkim, India.	-29C
D44	Lhotse	8,511 m	Himalayas	-429926400	27°57'45.4"N 86°56'1.4"E	no	<a href="https://en.wikipedia.org/wiki/Lhotse">en.wikipedia.org/wiki/Lhotse</a>	Lhotse is the fourth highest mountain in the world at 8,516 metres (27,940 ft), after Mount Everest, K2, and Kangchenjunga.	-28C
...									

### 4.1.1 Data Preparation

Before starting the annotation process, it is advisable to apply standard transformation rules to the values and the structure of the table so to preserve only the relevant information. Examples of transformation are the following: deletion of HTML tags and of some characters (i.e. " '), transformation of text into lowercase, deletion of text in brackets, explanation of acronyms and abbreviations, and normalisation of units of measurement. To decrypt acronyms and abbreviations, the Oxford English Dictionary<sup>3</sup> is used. The normalisation of units of measurement is performed by applying regular expressions, as described in [88]. MantisTable extends the original set of regular expressions to cover a complete set of units, which includes area, currency, density, electric current, energy, flow rate, force, frequency, fuel efficiency, information unit, length, linear mass density, mass, numbers, population density, power, pressure, speed, temperature, time, torque, voltage and volume. For each type of unit of measurement, a series of conversion rules has been defined, as the ones presented in Listing 4.1, which shows the equivalent ways to express measures of area.

```

1 | area | squareMetre | squaremetre, m2, m2, μ2, μ2, τ.μ.
2 | area | squareMillimetre | squaremillimetre, mm2, μμ2, μμ2 | 1.0E-6
3 | area | squareCentimetre | squarecentimetre, cm2, cm2 | 0.0001

```

<sup>2</sup>[bitbucket.org/disco\\_unimib/mantistable-tool/src/master/app/private/tables/test/mountains/](https://bitbucket.org/disco_unimib/mantistable-tool/src/master/app/private/tables/test/mountains/)

<sup>3</sup>[public.oed.com/how-to-use-the-oed/abbreviations/](https://public.oed.com/how-to-use-the-oed/abbreviations/)

```

4 | area | hectare | hectare, ha | 10000.0
5 | area | squareMile | squaremile, sqmi, mi2, mi^2 | 2589988.110336

```

Listing 4.1 Conversion table for units of measurement of area.

Table 4.2 shows the transformation of Table 4.1 after the completion of the Data Preparation phase. It is worth noticing that in this simulation the data preparation phase failed in the normalisation of some rows (e.g. the CONQUERED ON column). Despite this the tool will be able to treat these values.

Table 4.2 Table 4.1 after the Data Preparation phase.

ID	PEAK	HEIGHT	RANGE	CONQUERED ON	COORD.	B	URL	DESCRIPTION	T
11	mount everest	8848 m	himalayas	may 29, 1953	27.98785, 86.92502	1	..	mount everest, known in nepali as sagarmāthā and in tibetan as chomolungma, is earth's highest mountain above sea level [...]	-35C
b22	k-2	8611 m	karakoram	july 31, 1954	35.87998, 76.51510	0	..	k2, also known as mount godwin-austen or chhogori is the second highest mountain in the world, after mount everest (8,848 metres), at 8,611 metres (28,251 ft).	-30C
c33	kanchenjunga	8597 m	himalayas	may 25, 1955	27.70249, 88.14753	true	..	kangchenjunga, also spelled kanchenjunga, is the third highest mountain in the world, and lies partly in nepal and partly in sikkim, india.	-29C
d44	lhotse	8511 m	himalayas	-429926400	27°57'45.4"N 86°56'1.4"E	no	..	lhotse is the fourth mountain in the world at 8,516 metres (27,940 ft), after mount everest, k2, and kangchenjunga.	-28C

## 4.1.2 Column Analysis

The column analysis starts with the semantic classification of columns which takes into account the values of a column to mark it as Literal column (L-column) if values are datatypes (e.g. strings, numbers, dates, such as 4808, 10/04/1983), or as Named-Entity column (NE-column) if values are concepts (e.g. Mountain, Mountain Range, such as Mont\_Blanc, Mont\_Blanc\_massif). Identifying good L-columns candidates is the first step of Column Analysis phase. To accomplish this task, we follow and extend some approaches in the state-of-the-art [123, 88, 80, 69, 56]; we consider 16 regular expressions identifying the following Regextypes: geo coordinate, address, hex color code, numeric, boolean, URL, image type, credit card number, email, IP address, ISBN code, date, ID, textual description, IATA code and currency. If the number of occurrences of the most frequent Regextype exceeds a certain threshold  $\bar{\gamma}$  (the value of  $\bar{\gamma}$  will be discussed in Section 5.2.3), the Regextype is assigned to the analysed column. Otherwise it will be annotated as NE-column.

For example, in the mountain table (Table 4.2), this step assigns the L-column tag to the COORD. column, and the NE-column tag to the PEAK column.

The second step deals with the subject column detection that takes into account the detected NE-columns. The subject column (S-column) can be defined as the main column in the table: this column contains entities for which the other columns provide additional

information. Our approach considers different features (characteristics of the table, such as empty cells, cells with unique content, etc.) to identify the column that most likely corresponds to a subject. Table 4.3 shows a list of features. Our approach does not consider costly features such as the Web Search (ws) which requires the use of a search engine, and Context Match Score (cm) which analyses the external contexts (e.g. texts present in a web page). The use of features related to the presence of acronyms does not bring improvements in the detection of the S-column. Instead, we added the feature Average Number of Words (aw), which considers the average number of words in each cell. Like [123], we adopt the features Fraction of Empty Cells (emc), Fraction of Cells with Unique content (uc) and Distance from the First NE-column (df).

Table 4.3 Features for Subject Column Detection.

Feature	Notation
Fraction of empty cells	emc
Fraction of cells with unique content	uc
Distance from the first NE-column	df
Average number of words in each cell	aw

The aw feature [109] calculates the average number of words within the cells of each column. The column with the lowest average number of words per cell is considered the best candidate.

Finally, features are combined to compute the  $subcol(c_j)$  score for each NE-column, as follows:

$$subcol(c_j) = \frac{2uc_{norm}(c_j) + aw_{norm}(c_j) - emc_{norm}(c_j)}{\sqrt{df(c_j) + 1}} \quad (4.1)$$

The column with the highest score will be selected as the S-column for the considered table. Formula 4.1 is an adaptation of the one presented in [123], more details on its validation are in Section 5.2.3.

The values of the features for the S-column detection related to the mountain table (Table 4.1) are shown in Table 4.4. In this case the PEAK column is the S-column of the table (Table 4.5).

Table 4.4 Values of the features of the S-column detection for the mountain table.

Feature	PEAK column	RANGE column
emc	0	0
uc	1	0,1
df	0	2
aw	1,8	1

Table 4.5 The example table after Column Analysis phase.

L	S	L	NE	L	L	L	L	L	L
ID	PEAK	HEIGHT	RANGE	CONQUERED ON	COORD.	B	URL	DESCRIPTION	T
11	mount everest	8848 m	himalayas	may 29, 1953	27.98785, 86.92502	1	en.wikipedia.org /wiki/Mount_Everest	mount everest, known in nepali [...]	-35C
b22	k-2	8611 m	karakoram	july 31, 1954	35.87998, 76.51510	0	en.wikipedia.org /wiki/K2	k2, also known as [...]	-30C
c33	kanchenjunga	8597 m	himalayas	may 25, 1955	27.70249, 88.14753	true	en.wikipedia.org /wiki/Kangchenjunga	kangchenjunga also spelled [...]	-29C
d44	lhotse	8511 m	himalayas	-429926400	27°57'45.4"N 86°56'1.4"E	no	en.wikipedia.org /wiki/Lhotse	lhotse is the fourth [...]	-28C

### 4.1.3 Concept and Datatype Annotation

The main purpose of the Concept and Datatype Annotation phase is to search and extract entities, and the related concepts, to annotate the NE-columns. The task is accomplished by implementing a *Entity Matcher* and a *Concept Matcher*. The phase starts by considering the S-column, since it should contain a high number of distinct values and unambiguous entities (see features in Section 4.1.2). These characteristics allow for an easy identification of a valid candidate concept for the annotation of the S-column.

*a. Concept Annotation:* in the first step of this phase, we perform the entity-linking by searching the KG with the content of a cell  $tx(i, j)$ . The approach selects at most  $k$  cells of the column (the value of  $k$  will be discussed in Section 5.2.3). We use such terms to search the KG and get a set of entities. The similarity between the content of the cell  $tx(i, j)$  and the candidate entities  $e_{i,j} \in E_{i,j} \in E$  is used to disambiguate the content of the cell. Given a candidate entity  $e_{i,j}$ , the similarity depends on two components: entity context EC (econtext) and entity name EN (ename). EC is a score representing the similarity between the representation of the entity in the KG with the row and the column elements to which the cell belongs to. EN is a score that represents the similarity between the name of the entity in the KG and the text in the cell. EC is calculated by computing a candidate entity  $e_{i,j}$  with the cell's context  $x_{i,j} \in X_{i,j}$  which considers header and row content:

1. row content: is the concatenation of all the words in the cells in the same row  $j$  from every columns  $i$ , without considering the content of cell  $(i, j)$ ;
2. header content: is the concatenation of all the words of header  $(0, j)$  plus the concatenation of all the synonyms (e.g. from Wordnet<sup>4</sup> or Oxford dictionary<sup>5</sup>).

We calculate the EC as follow:

$$econtext(e_{i,j}) = |bow(abstract(e_{i,j})) \cap bow(rcontent(i, j))| + |bow(abstract(e_{i,j})) \cap bow(hcontent(i, j))| \quad (4.2)$$

<sup>4</sup>wordnet.princeton.edu

<sup>5</sup>oed.com

EN is calculated by computing the edit distance (Levenshtein) between the labels (in different languages) of candidate entity  $e_{i,j} \in E_{i,j}$  and the content of the cell  $tx(i, j)$ :

$$ename(e_{i,j}) = editDistance(tx(i, j), e_{i,j}) \quad (4.3)$$

The final objective is to identify the entity with the highest confidence score ECF, which will then be used for annotating the cell. The confidence score ECF (econf) is computed as follows:

$$econf(e_{i,j}) = bonus(e_{i,j}) + econtext(e_{i,j}) - ename(e_{i,j}) * 2 \quad (4.4)$$

The score  $bonus(e_{i,j})$  in the Formula 4.4 is used to rank the entities most related to the content of the cell by considering the presence of the tokens of the text in the cell, within the entity labels and the entity abstract (Formula 4.5).

$$bonus(e_{i,j}) = |bow(tx(i, j)) \cap bow(e_{i,j})| + |bow(tx(i, j)) \cap bow(abstract(e_{i,j}))| \quad (4.5)$$

For each cell  $tx(i, j)$  a set of candidate entities  $E_{i,j}$  is extracted from the KG, through the query shown in Listing 4.2. The query searches the entities considering both the entire content of the cell and the individual words. In addition, we search the descriptions associated with the entities according to the synonyms of the header. Values in the header are assumed to be nouns, thus the respective synonyms are extracted from WordNet, which is a semantic-lexical database of the English language, and from the thesaurus of Oxford dictionary. For instance, considering the example in Table 4.1, the approach searches the KG with the synonyms of PEAK, which is the header of the S-column, which results in *summit, mountain, bluff, ridge*<sup>6</sup>. The maximum number of results per query is set at 10; this number has been defined empirically with several tests which gave evidence that the correct result is mostly within the first 5 results.

```

1 SELECT DISTINCT (str(?s) as ?s) (str(?abstract) as ?abstract)
2 WHERE {
3   {
4     ?s dbo:abstract ?abstract .
5     ?s a ?type .
6     ?s rdfs:label ?label .
7     ?label <bif:contains> 'mount AND everest' .
8     ?abstract <bif:contains> '("peak" OR "summit" OR "mountain" OR [synonyms])' .
9   }
10  FILTER NOT EXISTS { ?s dbo:wikiPageRedirects ?r2 } .
11  FILTER (!strstarts(str(?s), 'http://dbpedia.org/resource/Category:')) .
12  FILTER (!strstarts(str(?s), 'http://dbpedia.org/property/')) .
13  FILTER (!strstarts(str(?s), 'http://dbpedia.org/ontology/')) .
14  FILTER (strstarts(str(?type), 'http://dbpedia.org/ontology/')) .

```

<sup>6</sup>[www.lexico.com/en/synonym/peak](http://www.lexico.com/en/synonym/peak)



```

15  FILTER (lang(?abstract) = 'en') .
16  }
17  ORDER BY ASC(strlen(?label))
18  LIMIT 10

```

Listing 4.2 SPARQL query to retrieve a set of candidate entities for a text in a cell.

In this example the row and header content are

1. row content: 11 8848 himalayas may 29 1953 27.98785 86.92502 ...;
2. header content: the synonyms of the header “peak” are “summit, mountain, bluff, ridge, ben, beg, jebel, mount”.

Applying the Formulas 4.2,4.3,4.4, the EC, EN and ECF are calculated for candidate entities:

```

1  "dbr:Mount_Everest"
2  score EC 1
3  score EN 0
4  score BONUS 1
5  score ECF 2.5 # winning entity
6  "dbr:Mount_Everest_Nepal"
7  score EC 0.06
8  score EN 0.31
9  score BONUS 1
10 score ECF 0.79
11 "dbr:Mount_Everest_webcam"
12 score EC 0.11
13 score EN 0.35
14 score BONUS 1
15 score ECF 0.61
16 "dbr:Joint_Himalayan_Committee"
17 score EC 0.10
18 score EN 0.8
19 score BONUS 0.5
20 score ECF -0.78
21 [...]

```

Listing 4.3 List of entities with entity context score, entity name score and confidence score.

In this case the winning entity is `dbr:Mount_Everest`<sup>7</sup>.

In the second step, a set of concepts  $CO_{i,j} \in CO$  associated with the winning entities  $e_{i,j}$ , identified in the previous step, are obtained. This set will then be used to identify a concept to be associated with the column.

In particular, for each winning entity, all the `rdf:type` values are extracted. Then, for each extracted type, the frequency (considering different ontologies) and the number of cells in which the type appears are calculated. Table 4.6 reports the frequencies referred to the example in Table 4.1 and the column PEAK.

To avoid possible incorrect links, we decide to select a set of types whose frequency score is close to the maximum frequency score of all types of the candidate list, up to a certain threshold. In other terms, we select the set of type candidates which fall within the range defined by the maximum frequency score as upper bound and by a threshold  $\bar{\delta}$  as lower

<sup>7</sup>[dbpedia.org/resource/Mount\\_Everest](http://dbpedia.org/resource/Mount_Everest)

Table 4.6 Class frequencies for each extracted entity.

PEAK	entity	rdf:type	type	frequency
mount everest	http://dbpedia.org/page/Mount_Everest	dbo:Place, dbo:Location, dbo:Mountain, dbo:NaturalPlace, schema:Mountain, schema:Place, umbel:Mountain [...]	Place	2
			Mountain	3
			NaturalPlace	1
			Location	1
k-2	http://dbpedia.org/resource/K-2_(Kansas_highway)	dbo:Place, dbo:Location, dbo:ArchitecturalStructure, dbo:Infrastructure, dbo:Road, dbo:RouteOfTransportation, schema:Place [...]	Place	2
			Location	1
			ArchitecturalStructure	1
			Infrastructure	1
			Road	1
			RouteOfTransportation	1
kangchenjunga	http://dbpedia.org/page/Kangchenjunga	dbo:Place, dbo:Location, dbo:Mountain, dbo:NaturalPlace, schema:Mountain, schema:Place, umbel:Mountain [...]	Place	2
			Mountain	3
			NaturalPlace	1
			Location	1
lhotse	http://dbpedia.org/page/Lhotse	dbo:Place, dbo:Location, dbo:Mountain, dbo:NaturalPlace, schema:Mountain, schema:Place, umbel:Mountain [...]	Place	2
			Mountain	3
			NaturalPlace	1
			Location	1

bound and if the type belongs to a number of cells greater than a threshold  $\bar{\beta}$  (the value of  $\bar{\delta}$  and  $\bar{\beta}$  will be discussed in Section 5.2.3).

Therefore, the final result from the phase is the one shown in Table 4.7.

Table 4.7 Global frequency values for the PEAK column.

Type	Global frequency	# cells
Place	28	14
Mountain	36	13
NaturalPlace	13	13
ArchitecturalStructure	1	1
Infrastructure	1	1
Road	1	1
RouteOfTransportation	1	1

The hierarchy of the concept in the example above is *Place* > *NaturalPlace* > *Mountain*, *Place* > *Location*. The occurrences of *NaturalPlace* are added to the occurrences of *Mountain*. The minimal concept is used to annotate the column. Figure 4.1 shows the workflow of Concept Annotation.

*b. Datatype Annotation:* for the Datatype Annotation, the results of the Column Type Analysis (Section 4.1.2) are taken into consideration. In that phase, a column is associated with a specific Regextype. To identify the correct Datatype, a mapping between the Regextype and the Datatype was created (Table 4.8).

For some Regextype the correspondence with the Datatype is univocal; for example, *geo coordinates* is associated with *xsd:float*. The numeric Regextype instead corresponds to more Datatypes. In this case further analysis in the last step of the approach (Predicate Annotation) is necessary to identify the correct Datatype in relation of the occurrences. Table 4.9 shows the example in Table 4.1 with final columns annotations.

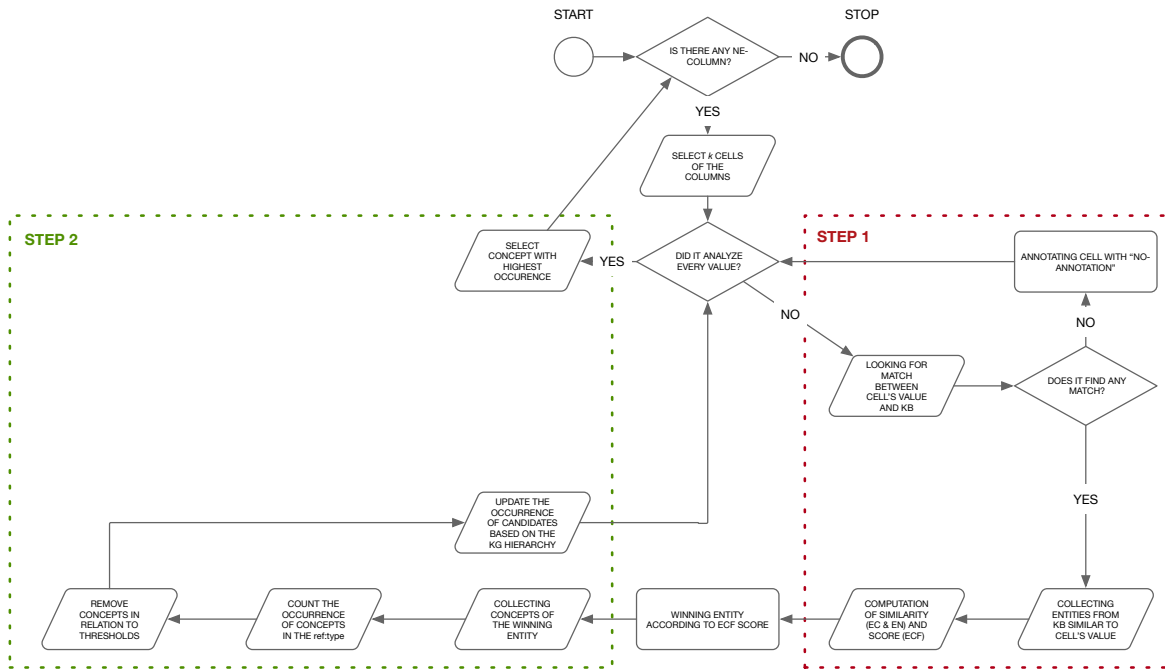


Figure 4.1 Concept annotation workflow.

#### 4.1.4 Predicate Annotation

For Predicate Annotation, the MantisTable approach considers the winning concept of the S-column as the subject of the relationship, and annotations of the other columns as objects. With this regard, KG is searched for the subject and the object. However, a clear distinction is made for the annotation of the properties between the S-column and NE-columns or S-column and L-columns. To perform predicate annotations we exploit two complementary techniques based on exploratory queries and summary profiles.

##### Predicate Annotation using exploratory queries

*a. Predicate Annotation for NE-column:* after annotating the NE-columns successfully, we proceed with the retrieval of all relations (predicates in the respective RDF triples) between the instances of the S-column concept and the instances of the NE-columns. Given the concept  $co_i$  of the S-column and the entities of the NE-column, we find the predicate whose subjects are of type  $co_i$  and the objects are instances of the NE-column. In addition, we execute a query where given the concept  $co_m$  of the NE-column and the instances of the S-column, we find the predicate whose subjects are all instances of the S-column and the objects are of type  $co_m$ . In order to identify the correct  $p_j \in P_j \in P$ , we compare the content of the column and the candidate predicates. Given a candidate predicate, the confidence score

Table 4.8 Datatype and Regextype mapping.

Regextype	Datatype	Description
geo coordinates	xsd:float	32-bit floating point
address	xsd:string	string
hex color	xsd:string	string
numeric	xsd:double	64-bit floating point
	xsd:float	32-bit floating point
	xsd:integer	integer value
	xsd:nonPositiveIntegerbyte	integer containing only non-positive values (...,-2,-1,0)
	xsd:negativeInteger	integer containing only negative values (...,-2,-1)
xsd:nonNegativeInteger	integer containing only non-negative values (0,1,2,...)	
xsd:positiveInteger	integer containing only positive values (1,2,...)	
boolean	xsd:boolean	boolean (true or false)
URL	xsd:anyURI	Uniform Resource Identifier
image	xsd:string	string
credit card	xsd:string	string
email	xsd:string	string
IP	xsd:string	string
ISBN	xsd:string	string

Table 4.9 Table 4.1 with column annotations.

-	dbo:Mountain	xsd:integer	dbo:MountainRange	xsd:date	xsd:string	-	xsd:string	-	-
ID	MOUNTAIN	HEIGHT	RANGE	CONQUERED ON	COORD.	B	URL	DESC.	T
11	mount everest	8848 m	himalayas	may 29, 1953	27.98785, 86.92502	1	en.wikipedia.org /wiki/Mount_Everest	[.]	-35C

depends on two components: the predicate context PC (pcontext) and predicate frequency PF (pfreq). PC is a score that represents the similarity between the representation of the predicate from the KG with the representation of the NE-column. PC is calculated by comparing a candidate predicate  $p_j$  with the column context  $x_j \in X_j$  which is further divided into in-table context and out-table context as in Table 4.10.

Table 4.10 Types of context for a table inside Predicate Annotation.

In-table context	Out-table context
column header	web search
column content	

We calculate the overlap between the representation of the candidate property  $p_j$  and the representation of each context  $x_j$  using the Dice similarity, computed as follows:

$$pcontext(p_j) = dice(p_j, x_j) = \frac{2 \cdot \sum_{w \in bowset(p_j) \cap bowset(x_j)} (freq(w, bow(p_j)) + freq(w, bow(x_j)))}{|bow(p_j)| + |bow(x_j)|} \quad (4.6)$$

PF refers to the ratio between the candidate predicate frequency and the sum of all candidate predicate frequencies.

$$pfreq(p_j) = \frac{|p_j|}{\sum_j |p_j|} \quad (4.7)$$

The selection of the predicate is computed using predicate confidence score (PCF):

$$pconf(p_j) = pcontext(p_j) + pfreq(p_j) \quad (4.8)$$

Listing 4.4 shows a query returning a set of candidate predicates  $P_j$  from the KG. The maximum number of the returned result is set at  $k = 50$  for performance reasons. Two distinct parts in the query can be identified. In the first part, we look for all the properties having entities of type `dbo:Mountain` as their subject and some values extracted from the NE-column (e.g. Himalayas, Karakoram) as object. In the second part, we look for all properties having some values extracted from the S-column as their subject (e.g. Mount Everest, Kangchenjunga) and some entities of type `dbo:Mountain_Range` as objects.

```

1 SELECT DISTINCT ?p (str(?plabel) as ?plabel) (count(?p) as ?count)
2 WHERE {
3   {
4     VALUES ?o {
5       <http://dbpedia.org/resource/Himalayas>
6       <http://dbpedia.org/resource/Karakoram>
7       [...]
8     } {
9       ?s ?p ?o .
10      ?p rdfs:label ?plabel .
11      ?s a <http://dbpedia.org/ontology/Mountain> .
12      FILTER (lang(?plabel)="en")
13    }
14  }
15  UNION {
16    VALUES ?s {
17      <http://dbpedia.org/resource/Mount_Everest>
18      <http://dbpedia.org/resource/Kangchenjunga>
19      [...]
20    } {
21      ?s ?p ?o .
22      ?p rdfs:label ?plabel .
23      ?o a <http://dbpedia.org/ontology/MountainRange> .
24      FILTER (lang(?plabel)="en")
25    }
26  }
27 }

```

Listing 4.4 SPARQL query to retrieve a set of candidate properties for a NE-column.

In this case the In-table context is

- column context: himalaya karakoram
- header context: range limit sierra line span

The Out-table context is

- web search: karakoram wikipedia the karakoram or karakorum is a large mountain range spanning the border of pakistan a significant part 28 50 of the karakoram range is glaciated compared to the himalaya 8 12 and alp 2 2 mountain glacier may serve a karakoram range mountain asia britannica com pakistan the himalayan and karakoram range the himalaya which has long been a physical and cultural divide between south and central asia ...

Applying the Formulas (4.6) (4.7) (4.8), PC, PF and PCF are calculated for predicate entites:

```

1 "dbo:locatedInArea"
2   frequency 13
3   score PC 0.0868
4   score PF 0.0050
5   score PCF 0.0918
6 "dbo:mountainRange"
7   frequency 1904
8   score PC 0.9898
9   score PF 0.9294
10  score PCF 1.9193 # winning predicate
11 "dbo:regionCode"
12   frequency 13
13   score PC 0.0050
14   score PF 0.0193
15   score PCF 0.0244

```

Listing 4.5 List of predicates with predicate context score, predicate frequency score and predicate confidence score.

In this case the winning predicate is `dbo:mountainRange`<sup>8</sup>.

*b. Predicate Annotation for L-column:* in a similar way to the Predicate Annotation for NE-column, we execute a query that given the concept  $co_j$  of the S-column and the values of the L-column, finds all predicates where the subject is  $co_j$  and the objects are values. In contrast to the previous query, we consider the synonyms of the headers (if present) in order to increase the number of predicate candidates. The query shown in Listing 4.6 returns a set of candidate predicates from the KG.

```

1 SELECT DISTINCT ?p (str(?plabel) as ?plabel) (count(?p) as ?count)
2 WHERE {
3   {
4     VALUES ?o {
5       8.848 8.611 8.597 8.511 8.481 8.167 8.156 8.153 8.124 8.078
6     } {
7       ?s ?p ?o .
8       ?p rdfs:label ?plabel .
9       ?s a <http://dbpedia.org/ontology/Mountain> .
10      FILTER (lang(?plabel)="en")
11    }
12  }
13  UNION {
14    ?s ?p ?o .
15    ?p rdfs:label ?plabel .
16    ?plabel bif:contains '("height" OR "meters" OR [synonyms])' .
17    FILTER (lang(?plabel)="en") .
18    FILTER isLiteral(?o)
19  }
20 }
21 LIMIT 50

```

Listing 4.6 SPARQL query to retrieve a set of candidate properties for a L-column.

The scores are computed with formula (4.6) (4.7) (4.8) as in the previous step.

```

1 "dbo:elevation"
2   frequency 238
3   score PC 0.9674
4   score PF 1.0635
5   score PCF 1.321 # winning predicate

```

<sup>8</sup>[dbpedia.org/ontology/mountainRange](http://dbpedia.org/ontology/mountainRange)

```

6  "dbo:heightMetric"
7  frequency 1
8  score PC 0.0040
9  score PF 0.8634
10 score PCF 0.2918
11 "dbo:heightDatum"
12 frequency 5
13 score PC 0.0203
14 score PF 0.8634
15 score PCF 0.3081

```

Listing 4.7 List of predicates with predicate context score, predicate frequency score and predicate confidence score.

*c. Predicate Annotation for numerical values:* the approach proposed for the Predicate Annotation for L-column often produces unsatisfactory results in case of columns containing numerical values (further discussion in Section 5.2.3). This is due to how numerical values are stored in the KG. To improve the quality of this step, we extended the idea proposed in [73], which applies a hierarchical clustering algorithm on a reference KG to build a Background Knowledge Graph (BKG). This BKG contains information about “numerical representative” of “contexts”, i.e. predicates and their shared domains (subject concept). For example, city temperatures, people age, longitude and latitude of cities. Starting from a set of numerical data, a k-nearest neighbours search is performed to associate a set of semantic labels with numerical values. Such labels enable the association of a value with a context. For example, if numerical values associated with label *height* are considered, such values can represent either the height of a building, of a person, or the elevation of a mountain.

More specifically, given two numerical sets, we analyse and compare the distribution of numerical values through the distance of Kolmogorov - Smirnov. This distance verifies whether two samples are taken from the same distribution by comparing the cumulative distribution functions  $F1$  and  $F2$ :

$$ksdist = \sup_x |F1(x) - F2(x)| \quad (4.9)$$

where  $\sup$  is the supremum of the distances. If two samples are equally distributed, i.e. the two bags hold the same numeric values, then the distance  $ksdist$  converges to 0.

The application of this technique returns top-k candidate predicates with related concepts. In order to identify among the top-k the most suitable predicate for the annotation of the numerical column, it is possible to exploit the annotations obtained in the previous phases. Starting from the annotation  $co_j$  of the S-column, we propose a technique supporting the rearrangement of the top-k results on the basis of their relevance with respect to  $co_j$  to obtain the most probable predicate candidates for the numerical columns. We propose three different methods of integration:

- **Exact Match.** An exact match is made between the  $co_j$  concept of the S-column and the top-k concepts obtained as a result of the search in the BKG. If the exact concept is present among the top-k results, the associated predicate is chosen as a semantic annotation.
- **Similarity match.** It is possible that the concept  $co_j$  of the S-column is not in the top-k results, but an equivalent concept is present. Starting from  $co_j$ , we then derive the list of similar concepts from DBpedia (using predicate `owl:equivalentClass`) which are compared one by one to the top-k candidate concepts. If at least one of the identified equivalent concepts is present among the top-k results, the associated predicate is chosen as the semantic annotation.
- **Filter for common ancestor.** A further level of filtering is performed by comparing the super-class of  $co_j$  (the classes that are hierarchically above the  $co_j$  class) with the super-class of each of the obtained top-k concepts, with the aim to exclude the candidates that do not belong to the same branch of the ontology. As we already described, numerical values referring to different concepts and predicates can share the same distribution, thus obtaining the discordant results that existing concepts and relative predicates belong to very different domains. For example, the results related to the capacity of stadiums are `Location` (with predicate `populationTotal`) and `Agent` (with predicate `numberOfStudents`). Therefore, starting from the set of top-k candidates, the SPARQL query in Listing 4.8 allows for counting the number of common super-class to  $co_j$ . In this case the only common ancestor is the `owl:Thing` class, thus the candidate and its predicate are deleted. This way, the ordering of the predicates is maintained, and, at the end of the comparisons, the predicate with the highest confidence score (PCF) is selected among the top-k results.

```

1 SELECT ?numentity (count(distinct ?entity) as ?count)
2 WHERE {
3     VALUES ?numentity { <top-k candates> }
4     <subjectConcept t> rdfs:subClassOf* ?entity .
5     ?numentity rdfs:subClassOf* ?entity2 .
6     FILTER (?entity = ?entity2).
7 }

```

Listing 4.8 SPARQL query to retrieve a set of super-classes of a concept.

In case the application of the three filters does not identify a single predicate (e.g. because more predicates are valid, or no predicates share common ancestry types), then the one with the highest predicate confidence score is selected.

For example, starting from the HEIGHT column of Table 4.1 containing numeric values that refer to mountain elevation, if the search within the BKG produces ambiguous results,



such as `populationTotal` and `elevation`, the concept associated with the subject column of the table becomes useful to disambiguate the correct annotation. In this specific case, if the subject column concept is `Mountain` the result of the numerical column annotation is `elevation`. Figure 4.2 shows the predicate annotation for Table 4.1.

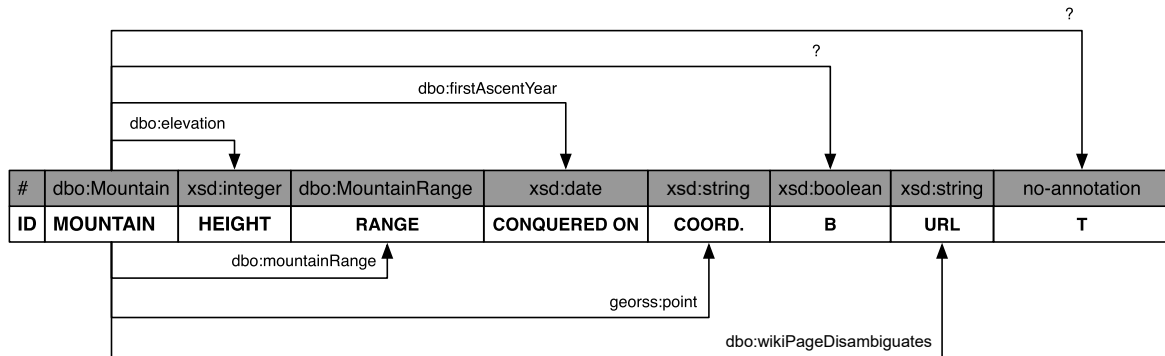


Figure 4.2 The example table with Predicate Annotations.

### Predicate Annotation using summaries profiles

The queries shown in Listings 4.4 and 4.6 may take too long to get an answer due to the number of synonyms to look for in the labels. Moreover, queries often cannot be executed because of timeout issues at the endpoint server. To increase the efficiency of the approach, we have integrated ABSTAT<sup>9</sup>, a distributed tool for calculating and exploring profiles for RDF data. ABSTAT takes a dataset and the related ontology (in OWL<sup>10</sup> format) as input, and produces a summary and statistics about the dataset. The summary is composed of patterns in the form of tuples  $\langle E, P, V \rangle$  which represent the presence of instances of  $E$  that are linked to instances of  $V$  via instances of  $P$ . Statistics includes frequencies for classes and predicates that are present in patterns. Figure 4.3 shows an example of patterns extracted by ABSTAT. The first line is a pattern  $\langle \text{dbo:Mountain} \text{ dbo:mountainRange} \text{ dbo:MountainRange} \rangle$  that states that there are instances of type `Mountain` linked to instances of type `MountainRange` via the predicate `mountainRange`. For each pattern several statistics are returned. Considering the highlighted pattern, the frequency of the pattern shows how many times does this pattern occur in the data set. The number of instances shows how many instances have this pattern including those for which the types `Mountain` and `MountainRange` and the predicate `mountainRange` can be inferred. Max (Min, Avg) `subjs-obj` cardinality is the maximal (minimal, average) number of distinct entities of type `Mountain` linked to a single entity of type `MountainRange` through

<sup>9</sup>[backend.abstat.disco.unimib.it](http://backend.abstat.disco.unimib.it)

<sup>10</sup>[www.w3.org/OWL/](http://www.w3.org/OWL/)

the predicate `mountainRange`. Max (Min, Avg) subj-objs is the maximal (minimal, average) number of distinct entities of type `MountainRange` linked to a single entity of type `Mountain` through the predicate `mountainRange`. Frequency is given also for types and predicates.

	subject type (occurrences)	predicate (occurrences)	object type (occurrences)	frequency	instances	Max subj-obj	Avg subj-obj	Min subj-obj	Max subj-objs	Avg subj-objs	Min subj-objs
filter	<input type="text" value="dbo:Mountain"/>	<input type="text" value="predicate"/>	<input type="text" value="dbo:MountainRange"/>								
	<a href="#">dbo:Mountain (16123)</a>	OP <a href="#">dbo:mountainRange (13104)</a>	<a href="#">dbo:MountainRange (2457)</a>	9972	9972	1653	11	1	3	1	1
	<a href="#">dbo:Mountain (16123)</a>	OP <a href="#">dbo:locatedInArea (46400)</a>	<a href="#">dbo:MountainRange (2457)</a>	50	50	9	2	1	1	1	1
	<a href="#">dbo:Mountain (16123)</a>	OP <a href="#">dbo:parentMountainPeak (2463)</a>	<a href="#">dbo:MountainRange (2457)</a>	17	17	15	6	1	1	1	1
	<a href="#">dbo:Mountain (16123)</a>	OP <a href="#">owl:differentFrom (50072)</a>	<a href="#">dbo:MountainRange (2457)</a>	7	7	2	1	1	1	1	1
	<a href="#">dbo:Mountain (16123)</a>	OP <a href="#">rdfs:seeAlso (156792)</a>	<a href="#">dbo:MountainRange (2457)</a>	3	3	1	1	1	1	1	1

Figure 4.3 ABSTAT profiling tool.

For the annotation of the predicates, MantisTable exploits the ABSTAT's APIs to extract the top  $n$  predicates (ranked with ABSTAT frequency statistics) that link two classes. For each predicate the PCF (Formula 4.8) must be calculated as previously specified in order to identify the predicate for the annotation.

### 4.1.5 Entity Linking

Entity Linking is the last phase of the approach. The annotations obtained in the previous steps are used to create a query for the disambiguation of the cell contents (Listing 4.9).

```

1 SELECT (str(?s) as ?s) ?type
2 WHERE {
3   ?s rdfs:label ?l .
4   ?l <bif:contains> '("mount everest" OR ("mount" AND "everest"))' .
5   ?s rdf:type ?type .
6   FILTER (lang(?l) = "en")
7 }

```

Listing 4.9 SPARQL query to retrieve entities.

However, the use of the winning class for filtering results was too stringent. This is because, from a series of experiments, it was noted that not all elements within a column have the same values for the `rdf:type` property due to some inconsistency of DBpedia. For this reason, to increase the number of results, we have chosen to consider only the entities for which, within the values of `rdf:type`, it is possible to find the label of the winning class. If more than one entity is returned, the one with a smaller edit distance (i.e. Wagner-Fischer distance) is taken.

## 4.2 Formalisation

In the following, we propose a formalisation of the STI approach. The formalisation takes into consideration the definition of STI proposed in the previous section. In order to understand the outputs of the approach, a formalisation of the inputs, a table and a KG, are proposed.

**Definition 1** *A rectangular array (matrix) of strings arranged in  $n$  rows and  $m$  columns is called a table. Every pair  $(i, j)$  with  $1 \leq i \leq n$  and  $1 \leq j \leq m$ , unambiguously identifies a cell of the table.*

**Definition 2** *Given an  $n \times m$  table, let  $r_i$  denote respectively the  $i$ -th row of the table, that is  $r_i = \{(i, j) | 1 \leq j \leq m\}$  and  $c_j$  denote the  $j$ -th column of the table, that is  $c_j = \{(i, j) | 1 \leq i \leq n\}$ . Let  $R = \{r_i | 1 \leq i \leq n\}$  and  $C = \{c_j | 1 \leq j \leq m\}$  be the set of all rows and columns of the table, respectively.*

**Definition 3** *A function header that associates each column  $c_j$  of the table with a word of a language  $LA(c_j \rightarrow LA)$  is called a Column Header Function.*

**Definition 4** *The pair  $T_h = (T, h)$  where  $T$  is a table and  $h$  is a column header function we define  $H = h(C)$  as the header of table  $T$ .*

See Fig 3.1 as an example of the elements just described.

The second input of STI is a KG. Inside a KG is possible to identify an ontology. Ontologies are structures for the organisation of knowledge in a particular domain. They are used to classify the terms, possible relationships, and define possible constraints on using those terms.

**Definition 5** *An ontology is a multigraph  $O = (S, P, A)$ , where:*

- $S = CO \cup DT$  is the set of semantic elements (e.g. *DBpedia Ontology, GeoNames Ontology*);
  - $CO$  is the set of concepts (e.g. *dbo:Mountain, dbo:MountainRange*);
  - $DT$  is the set of datatypes (e.g. *xsd:date, xsd:integer*);
- $P$  is the predicate label set (e.g. “*mountainRange*”, “*elevation*”);
- $A$  is a set of labeled directed edges  $A \subset S^2 \times P$ , where an edge can exist only between concepts or between a concept and a datatype.

According to the definition proposed in [92] and the definition of an ontology described above, the definition of a *KG* is given as follows:

**Definition 6** Given an ontology  $O = (S, P, A)$ , a *KG* is a directed multigraph defined by the tuple  $KG = (V, B, O, map, pmap)$  where:

- $V = E \cup L$  is the set of vertices;
  - $E$  is a set of entities (e.g.  $dbr:Mont\_Blanc$ ,  $dbr:Lyskamm$ );
  - $L$  is a set of literals (e.g. “4808”, “45.833 6.865”);
- $B$  is a set of directed edges connecting two vertices  $B \subset V^2$ , they represent links between entities, or between entities and literals;
- $map$  is the ontology mapping function  $map : V \rightarrow S$ , which links a vertex to a concept or datatype in the ontology (e.g.  $dbr:Mont\_Blanc$  maps to  $dbo:Mountain$  concept,  $dbr:Mont\_Blanc\_massif$  maps to  $dbo:MountainRange$  concept, “4808” maps to  $xsd:double$  datatype);
- $pmap$  is the predicate mapping function  $pmap : B \rightarrow P$ , which maps an edge to a predicate (e.g.  $dbr:Mont\_Blanc$   $dbo:mountainRange$   $dbr:Mont\_Blanc\_massif$ ,  $dbr:Mont\_Blanc$   $dbo:elevation$  “4808”).

A set of knowledge graph *KGs* is defined as follows:  $KGs = \{KG_1, KG_2, \dots, KG_x\}$ .

Table and Knowledge Graphs (*KGs*) allow us to define a threesome of *matcher*, in order to associate a concept or a datatype with a column and an entity (vertex) with a cell.

**Definition 7** Given a knowledge graph  $KG_x$ , the *Concept Matcher* is a function  $comatcher_x : T \rightarrow CO_x \cup \emptyset$ :

$$comatcher_x(i, j) = \begin{cases} co_x \in CO_x & \forall (i, j) \in T \\ \emptyset & \end{cases} \quad (4.10)$$

**Definition 8** Given a knowledge graph  $KG_x$ , the *Datatype Matcher* is a function  $dtmatcher_x : T \rightarrow DT_x \cup \emptyset$ :

$$dtmatcher_x(i, j) = \begin{cases} dt_x \in DT_x & \forall (i, j) \in T \\ \emptyset & \end{cases} \quad (4.11)$$

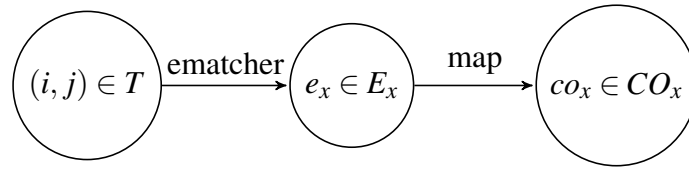
**Definition 9** Given a knowledge graph  $KG_x$ , an *Entity Matcher* is a function  $ematcher_x : T \rightarrow E_x \cup \emptyset$ :

$$ematcher_x(i, j) = \begin{cases} e_x \in E_x & \forall (i, j) \in T \\ \emptyset & \end{cases} \quad (4.12)$$

The application of the matchers can support the journey of the cell value and ontology In the scope of STI passing through the identification of an entity or a literal.

**Lemma 1** *Given a knowledge graph  $KG_x$ , a table  $T$  and an Entity Matcher  $ematcher_x$ , a particular Concept Matcher is defined as:*

$$comatcher_x(i, j) = \begin{cases} map_x(ematcher_x(i, j)), & \text{if } ematcher_x(i, j) \in E_x \\ \emptyset, & \text{otherwise} \end{cases} \quad \forall (i, j) \in T \quad (4.13)$$



The next definition supports the identification of L-column and NE-column from the output produced by the Datatype Matcher.

**Definition 10** *A Column Identifier is a function  $colid_x : C \rightarrow \{“NE-column”, “L-column”\}$ .*

**Lemma 2** *Given a knowledge graph  $KG_x$ , a table  $T$ , a threshold value  $\tilde{\gamma} \in \mathbb{R}$  and a function  $isdt_x : T \rightarrow \{1, 0\}$  defined as follows:*

$$isdt_x(i, j) = \begin{cases} 1, & \text{if } dtmatcher_x(i, j) \in DT_x \\ 0, & \text{otherwise} \end{cases} \quad \forall (i, j) \in T \quad (4.14)$$

*a particular Column Identifier can be defined as:*

$$colid_x(c_j) = \begin{cases} “L-column”, & \text{if } \sum_i isdt_x(i, j) \geq \tilde{\gamma}; \\ “NE-column”, & \text{otherwise.} \end{cases} \quad \forall c_j \in C \quad (4.15)$$

The next part provides a definition of semantic annotation for the elements of a table. In particular, the results of the Concept Matcher and the Datatype Matcher are considered in order to extrapolate the winning concepts and datatypes in relation to the number of occurrences.

**Definition 11** *Given a knowledge graph  $KG_x$  and table defined by a set of columns  $C$ , a Semantic Column Annotator is a function  $semannotator_x : C \rightarrow S_x$ .*

**Lemma 3** Given a knowledge graph  $KG_x$ , a table  $T$ , a Concept Matcher  $comatcher_x$  and Datatype Matcher  $dtmatcher_x$ ; sets  $D_{co_x}^j$  and  $D_{dt_x}^j$  defined as follows:

$$D_{co_x}^j = \{(i, j) | (i, j) \in T \wedge comatcher_x(i, j) = co_x \wedge i \in \{1, \dots, n\}\}, \quad \forall co_x \in CO_x \wedge \forall j \in \{1, \dots, m\}$$

$$D_{dt_x}^j = \{(i, j) | (i, j) \in T \wedge dtmatcher_x(i, j) = dt_x \wedge i \in \{1, \dots, n\}\}, \quad \forall dt_x \in DT_x \wedge \forall j \in \{1, \dots, m\}$$

a Semantic Column Annotator is the function:

$$semannotator_x(c_j) = \begin{cases} \arg \max_{dt_x \in DT_x} |D_{dt_x}^j|, & \text{if } colid(c_j) = \text{"L-column"}; \\ \arg \max_{co_x \in CO_x} |D_{co_x}^j|, & \text{otherwise.} \end{cases} \quad (4.16)$$

**Definition 12** Given a knowledge graph  $KG_x$ , the Predicate Matcher is a function  $pmatcher_x : C^2 \rightarrow A_x \cup \emptyset$ :

$$pmatcher_x((i, j), (i^*, j^*)) = \begin{cases} a_x \in A_x & \forall ((i, j), (i^*, j^*)) \in T \wedge colid(c_i) = \text{"N-column"}; \\ \emptyset & \end{cases} \quad (4.17)$$

**Lemma 4** Given a knowledge graph  $KG_x$ , a table  $T$  and Predicate Matcher  $pmatcher_x$

$$Q = \{(i, j), (i^*, j^*) | (i, j), (i^*, j^*) \in T \wedge pmatcher_x((i, j), (i^*, j^*)) = a_x \\ \wedge i, i^* \in \{1, \dots, n\} \\ \wedge j, j^* \in \{1, \dots, m\}\}, \forall a_x \in A_x$$

a Predicate Annotator is the function:

$$pannotator_x((i, j), (i^*, j^*)) = \arg \max_{a_x \in A_x} |Q| \quad (4.18)$$

# Chapter 5

## MantisTable Tool

### 5.1 Architecture and Interface

MantisTable is a web application developed with Python<sup>1</sup> and the Django framework<sup>2</sup>. A MongoDB<sup>3</sup> database acts as table and KG repository. The code is freely available through a Git repository<sup>4</sup>. In order to achieve the scalability of the application, and therefore improve efficiency, MantisTable has been installed in a Docker container to achieve parallelisation at the application level and to facilitate the deployment on servers. The management of resources is performed by using Task Queues (i.e. Celery Workers<sup>5</sup>). At database level, we exploited the capability of MongoDB to use the Sharding<sup>6</sup> method to distribute data across multiple machines. For the correct management of data collections we used a unique key (i.e. Shard Key<sup>7</sup>). The five phases of the STI have been modularly implemented, allowing an easy replacement or extension by other developers.

Figure 5.1 shows the modular architecture of MantisTable, which is organised in three layers: the *View Layer* that provides a graphic user interface to serve different types of tasks such as storing and loading tables, exploring the annotated tables to navigate every executed step and analyse the result, executing the STI steps, and editing to modify and enhance the results; the *Controller Layer* that creates the abstractions between the View layer and the Model layer, and implements all the STI steps; and the *Model Layer* that manages mainly

---

<sup>1</sup>[www.python.org](http://www.python.org)

<sup>2</sup>[www.djangoproject.com](http://www.djangoproject.com)

<sup>3</sup>[www.mongodb.com](http://www.mongodb.com)

<sup>4</sup>[bitbucket.org/disco\\_unimib/mantistable-tool.py](https://bitbucket.org/disco_unimib/mantistable-tool.py)

<sup>5</sup>[docs.celeryproject.org/en/latest/userguide/workers.html](https://docs.celeryproject.org/en/latest/userguide/workers.html)

<sup>6</sup>[docs.mongodb.com/manual/sharding/](https://docs.mongodb.com/manual/sharding/)

<sup>7</sup>[docs.mongodb.com/manual/core/sharding-shard-key](https://docs.mongodb.com/manual/core/sharding-shard-key)

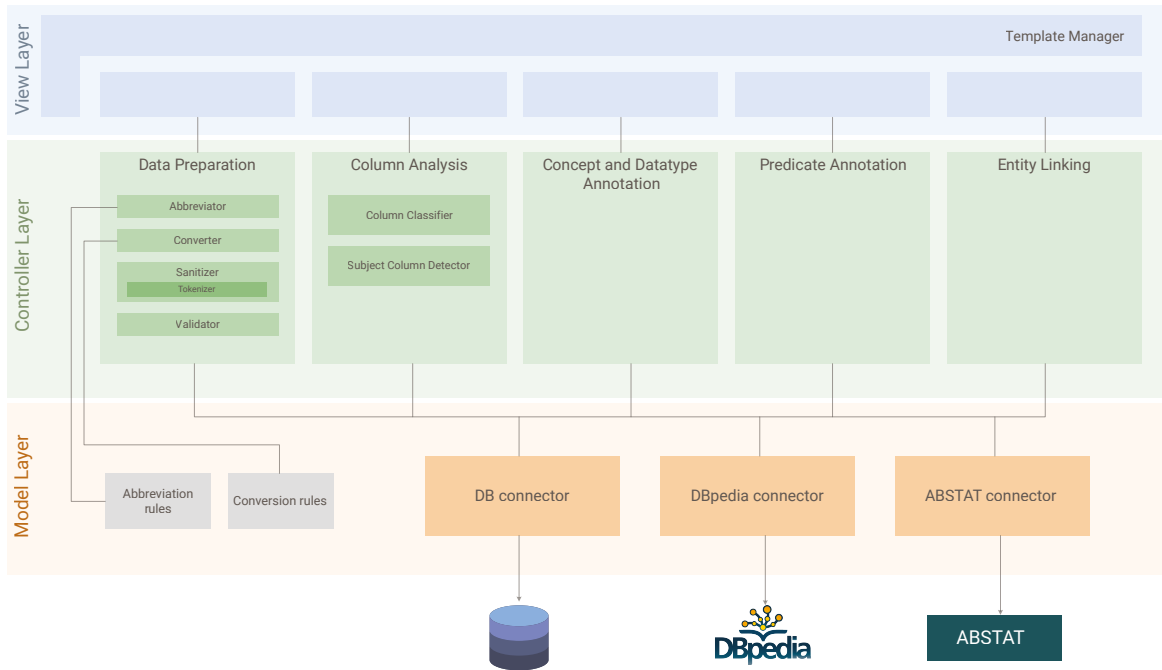


Figure 5.1 Architecture of MantisTable tool.

data access components to communicate with external data sources such as DB connector and DBpedia connector.

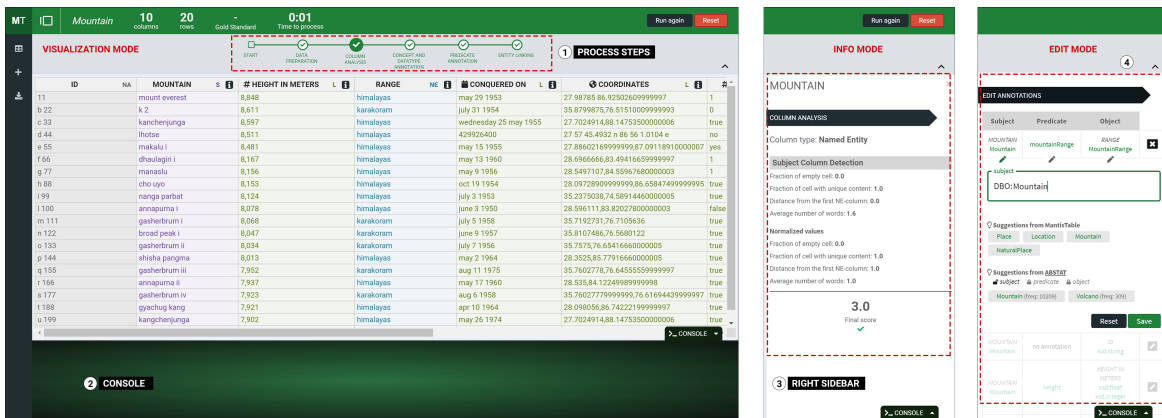


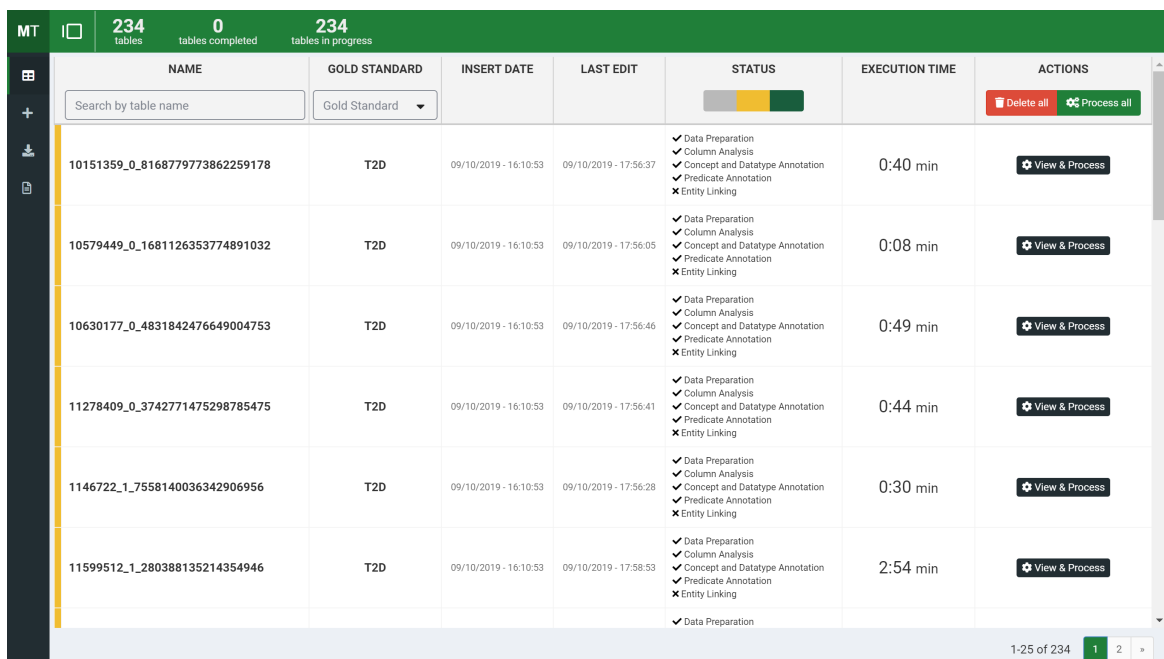
Figure 5.2 MantisTable interface overview: Visualization Mode (1. process, 2. console), Info Mode (3. right sidebar), Edit Mode (4. edit form).

This tool provides the following features: i) loading and storing ii) execution iii) exploration, and iv) editing.

**MantisTable Loading and Storing.** Tables are imported and stored in a MongoDB database. In MantisTable, a list of loaded tables is displayed on the main page (Figure



5.3). For each table, a series of metadata is provided, such as name, date of loading, date of last modification, which tasks have already been performed and which are being executed thus making the user aware of the status of the annotation process. Finally, users can download the annotated tables at the end of the annotation process. Through the interface it is possible to add and load new tables (in JSON format), delete all tables of the Gold Standard (T2Dv2<sup>8</sup> and Limaye200 [59]) and process all tables in batch. It is also possible to update or delete every single table.



MT	234 tables	0 tables completed	234 tables in progress					
NAME		GOLD STANDARD	INSERT DATE	LAST EDIT	STATUS	EXECUTION TIME	ACTIONS	
Search by table name		Gold Standard					Delete all	Process all
10151359_0_816879773862259178	T2D	09/10/2019 - 16:10:53	09/10/2019 - 17:56:37	<ul style="list-style-type: none"> <li>✓ Data Preparation</li> <li>✓ Column Analysis</li> <li>✓ Concept and Datatype Annotation</li> <li>✓ Predicate Annotation</li> <li>✗ Entity Linking</li> </ul>	0:40 min	View & Process		
10579449_0_1681126353774891032	T2D	09/10/2019 - 16:10:53	09/10/2019 - 17:56:05	<ul style="list-style-type: none"> <li>✓ Data Preparation</li> <li>✓ Column Analysis</li> <li>✓ Concept and Datatype Annotation</li> <li>✓ Predicate Annotation</li> <li>✗ Entity Linking</li> </ul>	0:08 min	View & Process		
10630177_0_4831842476649004753	T2D	09/10/2019 - 16:10:53	09/10/2019 - 17:56:46	<ul style="list-style-type: none"> <li>✓ Data Preparation</li> <li>✓ Column Analysis</li> <li>✓ Concept and Datatype Annotation</li> <li>✓ Predicate Annotation</li> <li>✗ Entity Linking</li> </ul>	0:49 min	View & Process		
11278409_0_3742771475298785475	T2D	09/10/2019 - 16:10:53	09/10/2019 - 17:56:41	<ul style="list-style-type: none"> <li>✓ Data Preparation</li> <li>✓ Column Analysis</li> <li>✓ Concept and Datatype Annotation</li> <li>✓ Predicate Annotation</li> <li>✗ Entity Linking</li> </ul>	0:44 min	View & Process		
1146722_1_7558140036342906956	T2D	09/10/2019 - 16:10:53	09/10/2019 - 17:56:28	<ul style="list-style-type: none"> <li>✓ Data Preparation</li> <li>✓ Column Analysis</li> <li>✓ Concept and Datatype Annotation</li> <li>✓ Predicate Annotation</li> <li>✗ Entity Linking</li> </ul>	0:30 min	View & Process		
11599512_1_280388135214354946	T2D	09/10/2019 - 16:10:53	09/10/2019 - 17:58:53	<ul style="list-style-type: none"> <li>✓ Data Preparation</li> <li>✓ Column Analysis</li> <li>✓ Concept and Datatype Annotation</li> <li>✓ Predicate Annotation</li> <li>✗ Entity Linking</li> </ul>	2:54 min	View & Process		
					✓ Data Preparation			

Figure 5.3 List of loaded tables on MantisTable main page.

**MantisTable Execution.** Having selected a table, it is possible to manage the execution of the five phases described in Section 4.1. The user can either run all steps together or run them step-by-step to supervise the execution.

**MantisTable Exploration.** It is possible to navigate all the executed steps by clicking on each phase and analyse the results in the visualization mode (Figure 5.2 - 1. process). For all phases, additional information about the execution is shown in the console located under the table (Figure 5.2 - 2. console). By clicking on a header or body cell, information about the current phase is reported in the info mode (Figure 5.2 - 3. right sidebar). Figure 5.4 shows the final annotated table at the end of MantisTable STI process.

<sup>8</sup>[webdatacommons.org/webtables/goldstandardV2.html](http://webdatacommons.org/webtables/goldstandardV2.html)

The screenshot displays the MantisTable interface during the final step of the STI process. The top bar shows 'Mountain' with 10 columns and 20 rows, a 'Gold Standard' status, and a processing time of 0:53. A progress bar indicates the current step is 'ENTITY LINKING'. The main table lists mountains with columns for ID, NA, MOUNTAIN, # HEIGHT IN METERS, RANGE, and CONQUERED. An 'ENTITY LINKING' panel on the right shows a table with Subject, Predicate, and Object columns, listing various relationships like 'elevation', 'mountainRange', and 'firstAscentYear'.

ID	NA	MOUNTAIN	# HEIGHT IN METERS	RANGE	CONQUERED
11		mount everest <a href="http://dbpedia.org/resource/Mount_Everest">http://dbpedia.org/resource/Mount_Everest</a>	8848	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1953-29-05
b 22		k 2 <a href="http://dbpedia.org/resource/K2">http://dbpedia.org/resource/K2</a>	8611	karakoram <a href="http://dbpedia.org/resource/Karakoram">http://dbpedia.org/resource/Karakoram</a>	1954-31-07
c 33		kanchenjunga <a href="http://dbpedia.org/resource/Kangchenjunga">http://dbpedia.org/resource/Kangchenjunga</a>	8597	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	wednesday 25
d 44		lhotse <a href="http://dbpedia.org/resource/Lhotse">http://dbpedia.org/resource/Lhotse</a>	8511	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	429926400
e 55		makalu i <a href="http://dbpedia.org/resource/Makalu">http://dbpedia.org/resource/Makalu</a>	8481	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1955-15-05
f 66		dhaulagiri i <a href="http://dbpedia.org/resource/Dhaulagiri">http://dbpedia.org/resource/Dhaulagiri</a>	8167	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1960-13-05
g 77		manaslu <a href="http://dbpedia.org/resource/Manaslu">http://dbpedia.org/resource/Manaslu</a>	8156	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1956-09-05
h 88		cho uyo <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	8153	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1954-19-10
i 99		nanga parbat <a href="http://dbpedia.org/resource/Nanga_Parbat">http://dbpedia.org/resource/Nanga_Parbat</a>	8124	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1953-03-07
l 100		annapurna i <a href="http://dbpedia.org/resource/Annapurna_I">http://dbpedia.org/resource/Annapurna_I</a>	8078	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1950-03-06
m 111		gasherbrum i <a href="http://dbpedia.org/resource/Gasherbrum_I">http://dbpedia.org/resource/Gasherbrum_I</a>	8068	karakoram <a href="http://dbpedia.org/resource/Karakoram">http://dbpedia.org/resource/Karakoram</a>	1958-05-07
n 122		broad peak i <a href="http://dbpedia.org/resource/Broad_Peak">http://dbpedia.org/resource/Broad_Peak</a>	8047	karakoram <a href="http://dbpedia.org/resource/Karakoram">http://dbpedia.org/resource/Karakoram</a>	1957-09-06
o 133		gasherbrum ii <a href="http://dbpedia.org/resource/Gasherbrum_II">http://dbpedia.org/resource/Gasherbrum_II</a>	8034	karakoram <a href="http://dbpedia.org/resource/Karakoram">http://dbpedia.org/resource/Karakoram</a>	1956-07-07
n 144		shisha pangma <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	8013	himalayas <a href="http://dbpedia.org/resource/Himalayas">http://dbpedia.org/resource/Himalayas</a>	1964-07-05

Subject	Predicate	Object
MOUNTAIN Mountain	elevation	HEIGHT IN METERS xsd:float xsd:integer xsd:double
MOUNTAIN Mountain	mountainRange	RANGE MountainRange
MOUNTAIN Mountain	firstAscentYear	CONQUERED ON xsd:date
MOUNTAIN Mountain	geoRes:point	COORDINATES xsd:string
MOUNTAIN Mountain	not found	BOOLEAN xsd:boolean
MOUNTAIN Mountain	wikiPageExtern...	URL xsd:anyURI
MOUNTAIN Mountain	not found	DESCRIPTION xsd:string

Figure 5.4 Final step of MantisTable STI process.

**MantisTable Editing.** Even if MantisTable implements a fully automated annotation process, it is important to allow users to understand what has been achieved and give them the opportunity to modify and enhance the results. The former has been achieved with the exploration features sketched above, the latter has been accomplished by providing a widget to edit the annotations (Figure - 4. edit mode). The annotation validation and editing require that the user has previous knowledge about the structure of the KG. Therefore, to support the user we integrate ABSTAT<sup>9</sup> [94].

<sup>9</sup>[backend.abstat.disco.unimib.it](http://backend.abstat.disco.unimib.it)

## 5.2 Validation

This section describes the evaluation of our approach. The aim of the experiments is to evaluate the correctness of the overall approach and, separately, of each single phase. We also compare our results with state of the art approaches. Finally, we describe STILT tool that is used to support the evaluation of the approach.

### 5.2.1 Datasets

For the first part of the experiments, we use two Gold Standards: Version 2 of the T2Dv2<sup>10</sup> and Limaye200 [59].

- The T2Dv2 consists of manually annotated row-to-instance, attribute-to-property and table-to-concept correspondences between 779 Web tables and the DBpedia Knowledge base Version 2014<sup>11</sup>. The tables originate from the English-language subset of the Web Data Commons Web Tables Corpus<sup>12</sup>. As described in [87], during the extraction, the tables have been classified in layout, entity, relational, matrix and other tables. Concerning our goal, we select the relational tables because they contain information about an entity. In particular, from the 779 tables in the Gold Standard, we select 234 tables which share at least one instance with DBpedia. The tables cover different topics including places, works, and people. Altogether, the Gold Standard contains 234 concepts, 25119 instances and 618 property correspondences. About half of the property correspondences refer to entity label attributes, while 381 correspondences refer to other attributes (object as well as data type attributes). The correspondences were created manually. Inside the T2Dv2, the tables are structured in JSON format; the JSON also contains additional text, which represents the external context of the respective table. The correspondences between the data in the table and the KG (the semantic annotations) are contained in 3 CSV files, where the different types row-to-instance, attribute-to-property and table-to-concept are respectively illustrated. The first type annotates each row of a table to an entity within the KG; the second provides an annotation for each attribute of the table to an ontological property; the last one associates the whole table with a unique ontological concept.
- The Limaye200 [123] is a dataset consisting of 200 Wikipedia tables extracted from LimayeAll [59]. For each table the types of columns (i.e. NE-column, L-column

---

<sup>10</sup>[webdatacommons.org/webtables/goldstandardV2.html](http://webdatacommons.org/webtables/goldstandardV2.html)

<sup>11</sup>[wiki.dbpedia.org/data-set-2014](http://wiki.dbpedia.org/data-set-2014)

<sup>12</sup>[commoncrawl.org](http://commoncrawl.org)

and S-column) are identified. For the NE-columns, annotation is provided using the concepts in Freebase. As previously specified, MantisTable uses DBpedia as KG. For this reason, as explained in Section 5.2.3, for each entity associated with a cell, the corresponding DBpedia concepts were extracted. The concepts are then used as a concept annotation for the column. In this case the tables are structured as XML.

Considering the two Gold Standards, the total number of annotated tables is 434. The characteristics are summarised in Table 5.1.

Table 5.1 Characteristics of the Gold Standards.

	Table	Columns				Rows				Structuredness	Columns			Concepts	Pred.
		total	min	max	avg	total	min	max	avg		S	NE	L		
T2Dv2	234	1157	1	13	4	27966	5	585	119	0.92	231	-	-	39	154
Limaye200	200	919	2	11	4	4036	3	102	20	0.97	200	504	216	84	-

Limaye200, compared to T2Dv2, is much smaller in terms of the total number of rows, while the number of total columns for both Gold Standards is the same. The ‘‘Structuredness’’ field indicates how many blank cells are present within tables. In particular, the Structuredness is a weighted sum of each table’s structuredness, where the weight of each table is based on its sum of columns and rows, normalised by the total sum of columns and rows [28].

In addition, we extend the experiments and run MantisTable on additional tables proposed by the challenge, ‘‘Tabular Data to Knowledge Graph Matching’’<sup>13</sup> which includes the following tasks organised into several evaluation rounds: i) assigning a type to a column (CTA task), ii) matching a cell to a KG entity (CEA task), iii) assigning a KG property to the relationship between two columns (CPA task). We considered 69 tables with 120 columns to annotate for the CTA task, 64 tables with 8418 cells to annotate for the CEA and 49 columns with 110 pairs of columns to annotate for the CPA.

## 5.2.2 Evaluation Measures

To measure the effectiveness of the annotation process, we adopt the metrics proposed in the challenge<sup>14</sup>. *Precision*  $P$  of the mapping between the table data and the KG is calculated using the following formula:

$$P = \frac{|PA|}{|SA|} \quad (5.1)$$

where a perfect annotation  $PA$  refers to the annotation returned by the STI algorithm which corresponds to the annotation of the Gold Standard. A submitted annotation  $SA$  refers to the annotation returned by the STI algorithm.

<sup>13</sup>[www.cs.ox.ac.uk/isg/challenges/sem-tab/](http://www.cs.ox.ac.uk/isg/challenges/sem-tab/)

<sup>14</sup>[www.aicrowd.com/challenges/iswc-2019-column-type-annotation-cta-challenge](http://www.aicrowd.com/challenges/iswc-2019-column-type-annotation-cta-challenge)

*Recall R* is calculated as follows:

$$R = \frac{|PA|}{|GA|} \quad (5.2)$$

where the number of ground truth annotations *GAs* correspond to the number of annotations in the Gold Standard. Finally, we combine the predefined measures through the *F-measure* (F1), which represents the harmonic mean between precision and recall.

### 5.2.3 Experimental Settings

**Approaches and implementation choices.** MantisTable is evaluated against two state of the art approaches and a baseline as follows:

1. the two state of the art approaches are [87] and [123]. The solution described in [87] applies the T2Dv2 described above, while [123] uses Limaye200. For a detailed description, please refer to the Section 3.2.
2. a further comparison is made with the results obtained from a Baseline semantic annotation algorithm, defined below.

The approach described in [123] could not be directly tested, as no runnable code was available at the time of conducting the evaluation, as reported also in [14]. The data shown in the comparison tables have been extracted from a new implementation<sup>15</sup>.

**Baseline Method.** The Baseline method executes the *Subject Column Detection* phase as defined for MantisTable, while it uses a different method for the *Concept and Datatype Annotation* and the *Predicate Annotation* steps. In this case, we use the contents of the cells to look for the corresponding entities within the KG, using a SPARQL query. The recovered entity is the one that has a label equivalent to the exact value of the cell. Subsequently, for the annotation of the NE-columns with a semantic concept, the candidate with the highest number of occurrences is selected. The *Predicate Annotation* follows a similar procedure: the candidate relationships are retrieved by looking for the predicate which exists between the entities identified in the S-column, and the value/entity in the same row of the other columns. The most frequent predicate is chosen for the final annotation.

<sup>15</sup>[bitbucket.org/disco\\_unimib/tableminer-imp](https://bitbucket.org/disco_unimib/tableminer-imp)

**Configuration of the hardware.** All experiments have been performed on a Linux machine with 2 cores (2.3 Ghz) and 8GB RAM. As KG the local replica of the online version of DBpedia was used<sup>16</sup>.

**Parameters settings.** In our approach, different parameters are included. During the evaluation we use different configurations to define the final values of these parameters; in particular for  $k$ , relative to the number of cells to be considered during the Concept and Datatype annotation phase. Figure 5.5 represents the variation of the precision when the  $k$  value changes. As from the graph, the maximum Precision is obtained for a value of  $k$  equal to 30. Regarding the thresholds  $\bar{\gamma}$  for the identification of types of columns (NE-column or L-column) in the Column Analysis (Section 4.1.2) phase, Figure 5.6 shows the different values of Precision and Recall. The analysis shows that at least 50% of the cells must belong to a specific Regextype in order to consider a column as an L-column. Figure 5.7 shows the precision for different values of  $\bar{\delta}$  and  $\bar{\beta}$ , used for filtering candidate concepts during the Concept and Datatype annotation phase (Section 4.1.3). The best results are obtained with  $\bar{\delta} = 0.5$  and  $\bar{\beta} = 0.4$ .  $\bar{\delta}$ . As described above,  $\bar{\delta}$  is the threshold relating to the frequency of a candidate type (considering different ontologies) while  $\bar{\beta}$  is the threshold relating to the number of cells that can be associated with that particular type.

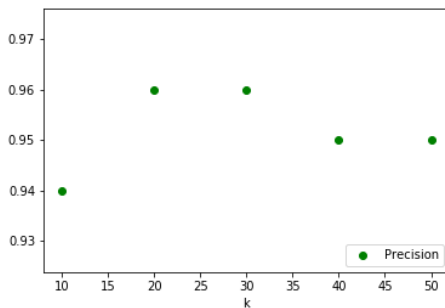


Figure 5.5 Precision of the approach when the  $k$  value changes.

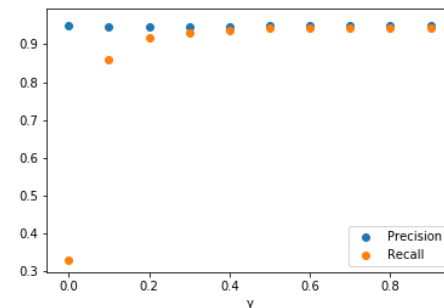


Figure 5.6 Precision and Recall of the approach when the  $\bar{\gamma}$  value changes.

The ECF threshold (Formula 4.4) refers to the number of words in the name of the corresponding entity. Considering the way our score is calculated, ECF will be lower if the entity name has only one word and will be higher if the entity name is composed of two or more words. At the same time entities with more than 2 words will get a global ECF score similar to the one with 2 words because the context will get a higher weight. In order to determine the best ECF threshold for entities with more than two words we analysed

<sup>16</sup>[dbpedia.org/sparql](http://dbpedia.org/sparql)

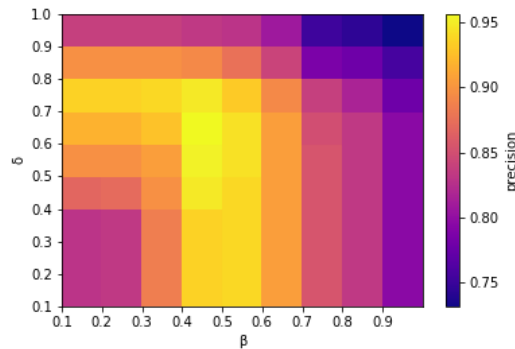


Figure 5.7 Precision of the approach when the  $\bar{\delta}$  and  $\bar{\beta}$  value change.

the worst performing table without using any threshold. The T2Dv2 table considered is 1438042986423\_95\_20150728002306-00329-ip-10-236-191-2\_805336391\_10 that is about Swimmers and only one entity per column has a corresponding DBpedia entity. With this analysis we determined that cells containing name and surname should both simultaneously find a match with the corresponding entities of DBpedia in order to be accepted. Therefore, the threshold we considered for entities composed of 2 or more word is 2.4. The best content to test a valid threshold for 1 word entities in T2Dv2 is a column with a list of countries (they contain 1 word cell for a great part of the columns), hence we consider table 24859353\_0\_7027810986004269522 that as a subject has a list of countries. In this way we determined that all the candidate entities with an ECF score lower than 1.5 are not correctly assigned, thus, these entities should not be considered in the process.

## 5.2.4 Evaluation results and Discussion

**Subject Column Detection .** Table 5.2 shows the evaluation of the Subject Column Detection phase. We test the MantisTable for the Subject Column Detection with four different configurations: i) with acronym features, ii) with context match features, iii) without web search score features, and iv) MantisTable without the first two features and with the third one, that is the web search score. As can be noticed from the results, we outperform the state of the art approaches [122] which use additional features such as the acronym and the context. MantisTable (fourth configuration) provides a slightly better result than the other approaches. However, there can still be problems to reach a precision equal to 1 as shown by the examples in Table 5.3. In general, the problems regard the disambiguation of entities containing unique values, or two different columns with very similar entities.

Method	T2Dv2
Zhang et al. [123]	0.87
MantisTable	0.98
MantisTable - fraction of empty cells feature	0.9444
MantisTable - fraction of cells with unique content feature	0.9401
MantisTable - distance from the first NE-column	0.9444
MantisTable - average number of words in each cell	0.93

Table 5.2 Results of the Subject Column Detection step.

Table name	GS	Example	Topic	Note
12125836_0_1134348206297032434	T2Dv2	AM 900; KALI AM; Spanish News/Talk; 747 E Green St, Pasadena 91101; (626) 844-8882	Radio	Both the frequency and the name of the radio station are unique and have a similar score
6310680_0_5150772059999313798	T2Dv2	Afghanistan; Afghani; AFA; 4	Currency	Possible T2Dv2 error
80184932_0_4240003884724905487	T2Dv2	Diversified Industrials; 3M Co.; Solutions videos, demo videos	Companies and Industries	The two NE columns are very similar to each other

Table 5.3 Problem on Subject Column Detection step.

**Concept and Datatype Annotation.** Table 5.4 shows the results of precision and recall for Concept and Datatype Annotation. The table-to-concept annotations of the T2Dv2 Gold Standard can be associated with the Concept and Datatype Annotation defined above. This association is based on the assumption that the annotation made by the technique is considered valid, according to the Gold Standard, if the table-to-concept annotation present for a specific table corresponds to the concept annotation made for the subject column of the same table. The first issue in this phase regard heterogeneous entities for which it is not possible to identify a common and specific concept. For example, table 48456557\_0\_3760853481322708783 (T2D) contains animal of various species including frogs (e.g. Bullfrog), deer (e.g. Elk) and birds (e.g. Gray Catbird). Our algorithm finds as the winning concept the one having the greater cumulative frequency which, in the example, corresponds to the Bird concept. If we change the parameters thus taking the general concept for this particular example, it will provide errors with other tables for which we should take the most specific concept. Another issue occurs when there is no corresponding entity in the KG. This is due to entities that are not popular as shown in table 1438042989018\_40\_20150728002309-00067-ip-10-236-191-2\_57714692\_2 (T2Dv2) containing the surnames of cricket players. As a consequence it is not possible to identify the concept.

Method	T2Dv2			Limaye200		
	P	R	F1	P	R	F1
Zhang et al. [123]	0.9	0.89	0.9	0.63	0.56	0.59
Ritze et al. [88]	0.89	0.88	0.89	0.51	0.26	0.35
Baseline	0.83	0.45	0.58	0.68	0.60	0.63
<b>MantisTable</b>	0.95	0.95	0.95	0.77	0.93	0.84

Table 5.4 Results of the Concept Annotation.



**Property Annotation.** The T2Dv2 Gold Standard attribute-to-property annotations can be traced back to the previously defined Property Annotation. Table 5.5 shows the results of Property Annotation, respectively for the Baseline, T2K [87] and MantisTable. While we get better results for the property annotation between two NE-columns, lower results are provided for the property annotation between an NE-column and a L-column. As shown in Table 5.5, all the approaches get lower results for L-columns (having numerical values). The reason for that is that connecting correctly an entity to a literal through a predicate is not straightforward because is necessary to have a context in order to better disambiguate literals.

Method	T2Dv2		
	P	R	F1
Zhang et al. [123]	0.43	0.31	0.36
Ritze et al. [88]	0.23	0.73	0.35
Baseline	0.49	0.38	0.43
<b>MantisTable</b>	0.57	0.45	0.51

Table 5.5 Results of the Predicate Annotation for NE-column.

Method	T2Dv2
	P
Neumaier et al. [73]	0.32
MantisTable	0.16
<b>MantisTable + Neumaier et al. [73] + schema</b>	0.43

Table 5.6 Results of the Predicate Annotation for L-column.

## 5.2.5 Challenge Evaluation Results

Tasks	Challenge Datasets	
	F1	P
CEA	1.0	1.0
CTA	0.929	0.933
CPA	0.965	0.991

Table 5.7 Results of the Challenge Tasks.

Table 5.7 shows the results of precision for the three tasks of the challenge. The main issues faced in these tasks were: i) tables with few rows (1-2) are sometimes difficult to be linked; this may be addressed by trying to integrate some other data sources (maybe considering Wikidata) for the entity linking; ii) some target columns are very complex to annotate because the cell contents cannot be directly linked to entities in the KG, so we decided to exclude the columns that our tool identified as L-columns; iii) tables about people with only surnames are frequently linked to homonymous entities, some specific solutions to expand the context in this kind of situation need to be adopted.

### 5.2.6 STILT tool

To support an automatic evaluation we developed STILT tool<sup>17</sup>. The STILT tool allows to evaluate MantisTable annotations for T2Dv2 and Limaye200 Gold Standard tables. The tool takes a JSON as input, that can be downloaded from the MantisTable tool via the download annotation feature. The input file contains information about the tables of the Gold Standard and the annotations that MantisTable obtained on various tables. The output returned by STILT tool is an analysis of the results obtained by MantisTable which shows the correctness of the annotations. In particular, it is possible to compare MantisTable annotations and those of the Gold Standard for each table. This comparison can be done by calculating evaluation metrics. STILT tool is developed in NodeJs and Meteor. The code is freely available at a Git repository<sup>18</sup>.

---

<sup>17</sup>[mantistable.disco.unimib.it](http://mantistable.disco.unimib.it)

<sup>18</sup>[bitbucket.org/disco\\_unimib/ti\\_eval/](https://bitbucket.org/disco_unimib/ti_eval/)

## **Part II**

# **API Composition using Semantic Annotations**



# Chapter 6

## State-of-the-art

The ability to provide appropriate and complete API descriptions, to let users discover services that satisfy a set of requirements and compose them to fulfil more complex needs, is critical for the success of any modern ICT solution. Extensive researches have been conducted with the intention to create automatic integration of Web Services and APIs. Most of these approaches deal with making candidate APIs communicate to each other, which is problematic because of the lack of semantic matching between input and output data. Although implementing APIs has become common practice, meta-level API definition and implementation have yet to be settled to widely-accepted standards [91]. To automate the interactions between APIs, a semantics description of the exchanged data is needed. Approaches to achieve this goal are: creating API descriptions in a logic-based language (e.g. RDF), or linking existing descriptions to shared domain vocabularies or ontologies (e.g. DBpedia). As the former demands expertise in logic-based languages, its adoption proved to be limited; the latter is more approachable, and enriching existing descriptions reduces the effort required.

There are many active initiatives to promote the creation and publication of descriptions associated with APIs. However, a shortcoming is the lack of support to add detailed information that qualifies the properties of them (e.g. classification of input and response data). As a result, these formats are suitable to complete simple tasks, but inefficient in automatic API discovery and composition due to the lack of machine processable semantics [111]. A critical aspect is the capability of including metadata, which can be interpreted by machine agents in a bottom up way (i.e. information structure should be in pieces to whole) [112]. In the real world, a developer may need to compose APIs that refer, for example, to location information. He or she may search directories such as Programmable Web<sup>1</sup>, collect descriptions, and understand the meaning of involved terms, e.g. understand that *address* refers to *city* and

---

<sup>1</sup>[www.programmableweb.com](http://www.programmableweb.com)

*street*, and *latitude/longitude* refer to a geographic *area*; but a machine agent is unable to understand those links without a shared representation of property semantics. The use of links to concepts in shared vocabularies allows machine agents to address the issue.

In the last decade, we have witnessed the evolution of web services models from the WSDL/SOAP to the REST. This change is tangibly visible, for example, by searching ProgrammableWeb<sup>2</sup>. One of the reasons for this evolution is the need to simplify the service reference model to enhance comprehensibility and standardisation, and therefore provide the bases for automatic management of descriptions and composition. A similar evolution is needed in the realm of semantic web services. As a matter of facts, well-defined proposals that deliver machine-readable descriptions, such as OWL-S: Semantic Markup for Web Services [64], Semantic Annotation for WSDL and XML Schema (SA-WSDL) [54], Micro Web Service Model Ontology (MicroWSMO) [51] and Semantic Annotations for REST (SA-REST) [38], failed to become widely used mainly because of their complexity that requires the involvement of experts.

The current description models address services accessible through API REST, and provide meta-languages to describe services as documents based on *property-value* pairs. OpenAPI Specification<sup>3</sup>, also known as Swagger<sup>4</sup>, API Blueprint<sup>5</sup> and RAML<sup>6</sup> are the most representative. However, these models do not support semantic annotations to make *property-value* inter-operable pairs. In this chapter, we discuss an extension of the popular OpenAPI model, to add semantic annotations on input parameters and output properties of services. Such annotations are compliant to the JSON-LD<sup>7</sup> format, following the REST philosophy in order to minimise the user involvement in many practical situations.

The availability of semantic descriptions of APIs enables the development of automatic techniques and tools to support services composition [89]. A general definition states that a process of composition is defined as the aggregation of different Web services into a single compound service to perform more complex functions [91]. In this context, we refer to information services and the mash-up of *results* obtained from independent services in delivering comprehensive answers to users' requests, or about preparing data coming from a set of services to invoke another service. We call the former *merge composition* and the latter *sequence composition*. Merge composition involves more services that are invoked in parallel with the same input data, whose answers are then composed. Sequence composition

---

<sup>2</sup>[www.programmableweb.com/apis/directory](http://www.programmableweb.com/apis/directory)

<sup>3</sup>[www.openapis.org](http://www.openapis.org)

<sup>4</sup>[swagger.io](http://swagger.io)

<sup>5</sup>[apiblueprint.org](http://apiblueprint.org)

<sup>6</sup>[raml.org](http://raml.org)

<sup>7</sup>[json-ld.org](http://json-ld.org)

involves a service which is invoked using input data coming from the composition of answers from one (adaptation) or more (mash-up) services.

The goal of our project is to (semi)automatically create semantic descriptions that correlate properties at a semantic level to enhance inter-operability and composition by machine. The adopted methodology is: (i) evaluate the current approaches to create API descriptions to identify a reference format; (ii) integrate MantisTable approach 4.1 to collect sample data from existing APIs and associate them to appropriate concepts from shared vocabularies; and finally (iii) develop methods to support automatic composition.

## 6.1 Approaches Supporting Service Descriptions and Composition

### 6.1.1 Service Descriptions

Descriptions have been classified as: functional, when they deal with provided APIs and exchange parameters to state what a service provides and how to access it, and non-functional, when they deal with meta information that allow potential users to understand how a given service provides its service [57]. A further classification splits descriptions in syntactic and semantic. The former deals with the format of calls and exchanged messages, and the latter adds a meaning to the description terms.

The most popular syntactic description model is WSDL 2.0 (Web Services Description Language) [15], which defines an XML format for describing Web services by separating the abstract functionality offered by a service from concrete details such as “how” and “where” that functionality is offered. Although it supports descriptions of both SOAP-based services, and REST/API services, it is the de-facto standard for the former, but is rarely adopted for the latter. The Web Application Description Language (WADL) [43] is a machine-readable XML format that was explicitly proposed for API services. WADL was also proposed for standardisation, but there was no follow-up.

More recently, *user-friendly* and *easy-to-use* metadata formats have been introduced, along with editors to support developers in the creation of descriptions for REST APIs. Among others, popular description formats are the Open API Specification (OAS)<sup>8</sup> (also known as Swagger specification), which provides human-readable API descriptions based on YAML and JSON. RAML is a YAML-based language for describing RESTful APIs. API Blueprint is a documentation-oriented web API description language, which provides a set of semantic assumptions laid on top of the Markdown syntax. The Hydra specification, which is currently under heavy development, tries to enrich current web APIs with tools and techniques from the semantic web area.

The OAS is the most promising choice at the moment [105], since (i) a simple format to specify descriptions, and (ii) a large set of vendor-neutral API tools, supported by a very large community of active users, are provided. Such tools provide great support to almost every modern programming languages to create and test APIs. Moreover, the Open API Initiative is an open source project sustained by relevant stakeholders, such as Google, IBM, Microsoft and PayPal<sup>9</sup>.

---

<sup>8</sup>[www.openapis.org/specification/repo](http://www.openapis.org/specification/repo)

<sup>9</sup>[www.openapis.org/membership/members](http://www.openapis.org/membership/members)



The description formats discussed so far are mainly syntactic, which means that little support to automate operations such as services discovery and composition, and verification of coherence to given interaction and building patterns is provided. Although there are many approaches proposed to enrich services descriptions with semantics, the manual work required to create descriptions, and the lack of inter-operability standards limit their adoption. The initial approach proposed by the semantic web community was to define a global ontology to include model, definitions and descriptions in a coherent system that can be used to make discovery and automatic composition. The most popular proposals are OWL-S (Ontology Web Language for Services) [64] and WSMO (Web Service Modelling Ontology) [89]. The major problem with these approaches is the expertise required to build and manage such descriptions. Although some approaches have been proposed in the past to automate the creation of descriptions [4], nobody currently uses both these approaches and descriptions. Anyway, the knowledge gained with these semantic studies has led to the definition of simpler and easier models that use the annotation approach introduced by hRESTS and RDFa.

Table 6.1 illustrates the characteristics of API description models with respect to the supported type of services (SOAP and/or REST), the capability of hosting semantic annotations, the serialisation language to publish the descriptions, the availability of supporting tools, and finally the human readability of the descriptions.

Table 6.1 Comparison of API description standards.

<i>Description</i>	<i>Service type</i>	<i>Semantics</i>		<i>Serialization</i>	<i>Tool</i>	<i>Human Readable</i>
		<i>Yes/No</i>	<i>Format</i>			
WSDL [15]	v1.1 SOAP v2.0 REST	No	-	XML	Yes	No
WADL [43]	REST	No	-	XML	Yes	No
hREST [51]	REST	No	-	Microformat	No	Yes
RDFa [2]	REST	No	-	HTML+RDF	No	Yes
OpenAPI Specification	REST	No	-	YAML, JSON	Yes	Yes
RAML	REST	No	-	YAML	Yes	Yes
API Blueprint	REST	No	-	Markdown	Yes	Yes
OWL-S [64]	SOAP REST	Yes	OWL	OWL	No	No
WSMO [89]	SOAP REST	Yes	MOF <sup>a</sup>	MOF	No	No
SA-WSDL [54]	v1.1 SOAP v2.0 REST	Yes	RDF	XML	No	No
Micro WSMO [51]	REST	Yes	RDF	RDF	No	Yes
SA-REST [38]	REST	Yes	RDF, OWL	RDF	No	Yes

<sup>a</sup> Meta-Object Facility

Within the Semantic Web, several existing approaches recognise the value of combining REST services and Linked Data [53, 95, 113]. Among these, a valid alternative is represented by Data-Fu<sup>10</sup> [97]. Data-Fu is a data and resource-driven programming approach leveraging

<sup>10</sup><https://linked-data-fu.github.io/>

the combination of REST with Linked Data. Data-Fu enables the development of applications built on semantic web resources with a declarative rule language. The main goal of Data-Fu is to minimise the manual effort to develop web based applications and the preservation of loose coupling by:

- leveraging links between resources provided by Linked Data, and
- specifying desired interactions dependent on resource states, which is enabled by a uniform state description format (i.e. RDF).

### 6.1.2 Service Composition

In the last decade, the composition of services has been widely investigated without achieving effective results for many reasons. Among others, the most relevant are the use of different architectural styles, the unexpected evolution of services, and the use of different description languages and different conceptual models [84]. Moreover, composition may occur at the design stage, leading to *static* compositions, or at runtime, leading to *dynamic* composition. The latter is best suited to address the issues in real environments that change continuously and requires automatic tools to search for, select and compose Web services automatically. The main issue affecting automatic composition is the limited number of available machine-readable descriptions associated with services.

A traditional way to compose services is the use of orchestration languages, such as BPEL (Business Process Execution Language) [118] or OWL-S (Ontology Web Language for Services) [64], which support the manual definition of abstract processes that can be implemented by actual services. On the other side, dynamic composition in *automatic* way can be achieved by exploiting the semantic Web and the planning techniques. However, the realisation of a completely automatic composition process is complex and presents several issues [91]. Some approaches try to exploit the languages described above (e.g. WSDL) to create semi-automatic composition frameworks [93]. The main problems are the missing of semantics associated with services, and the capability of understanding the semantics even when present.

Table 6.2 is an extension of the one presented in [105] to compare the number of questions posed in Stack Overflow and the number of Git stars (showing appreciation to a project) received by the four description models under study. The increasing number of available descriptions highlights the growing popularity of descriptions, and the relevance of tools that support the creation, publication, use and maintenance of service descriptions. The common limitation of such models is the lack of semantic descriptions, which motivated our

previous paper [18]. In order to be effective, we extended the most popular model, OpenAPI, to support semantic-enabled tools for describing, discovering, and then compose APIs.

Table 6.2 Comparison of API description models.

<i>Detail/Model</i>		<i>API Blueprint</i>	<i>RAML</i>	<i>WADL</i>	<i>OpenAPI Spec</i>
<i>Format</i>		Markdown	YAML	XML	YAML, JSON
<i>Licence</i>		MIT	ASL2.0	Sun	ASL 2.0
<i>Version</i>		Format 1A revision 9	1.0.1	31 August 2009	3.0.1
<i>Initial commit</i>		Apr 2013	Sep 2013	Nov 2006	Jul 2011
<i>Pricing plan</i>		Yes	Yes	No	No
<i>StackOverflow Questions</i>	2015	88	153	86	13
	2016	61	168	84	166
	2017	40	174	74	319
	2018	15	56	33	218
<i>Github Stars</i>	2015	1,819	1,058	N/A	2,459
	2016	X	X		X
	2017	5,390	2,735		6,360
	2018	6566	3060		9836
<i>Google Search</i>		985K	1M	486K	8M



# Chapter 7

## New API Description Model

### 7.1 AutomAPIc Approach

In this Section we describe in details the AutomAPIc approach which consists of the following phases:

1. **Semantic Description Creation**, which aims to create a semantic description of API using OpenAPI specification;
2. **Composition Types and Rules Analysis**, which aims to identify the possible compositions between different API.

#### 7.1.1 Semantic Description Creation

The OpenAPI is the most promising description model since it defines a simple format to specify descriptions supported by a broad set of vendor-neutral API tools, whose development involves a massive community of active users. Such tools provide significant support to almost every modern programming languages to create and test APIs. Moreover, the OpenAPI Initiative is an open source project sustained by relevant stakeholders, including Google, IBM, Microsoft and PayPal. There are several repositories collecting API REST described using OpenAPI, such as SmartAPI<sup>1</sup> and APIs.guru<sup>2</sup>.

An OpenAPI description is a YAML or JSON document that contains a list of resources and a list of operations that can be applied to those resources. An example is provided in Listing 7.1, which describes the Google Books API. Notice that the API is described by *name:value* pairs of strings without any semantics.

---

<sup>1</sup>[smart-api.info/registry](http://smart-api.info/registry)

<sup>2</sup>[apis.guru/openapi-directory/](http://apis.guru/openapi-directory/)

We propose to extend such descriptions by inserting annotations (i.e. links to ontology classes and ontology properties) through the use of the JSON-LD<sup>3</sup> format. JSON-LD provides (i) a universal identification mechanism for JSON objects through the use of Internationalised Resource Identifiers (IRIs); (ii) a way to disambiguate shared keys between different JSON documents through IRIs mapping and context; (iii) the possibility to annotate the strings with indications on the used language; and (iv) a way to associate data types with values (e.g. dates, times).

```

1  "paths": {
2  "/volumes": {
3    "get": {
4      "parameters": [{
5        "name": "title", [...]
6      }],
7    },
8    "responses": {
9      "200": {
10     "schema": {
11       "title": "result",
12       "type": "object",
13       "properties": {
14         "isbn": { "type": "string"},
15         "author": { "type": "string" },
16         "title": { "type": "string" }, [...]
17       }
18     }
19   }

```

Listing 7.1 OpenAPI description of the Google Books API.

The union of JSON-LD and OpenAPI descriptions occurs through the introduction of the *semanticAnnotations* property (e.g. Listing 7.2, line 8 and 27), which is composed of two parts: the definition of a context, by the keyword *@context* (e.g. line 9 and 28), to set short names for the reference ontologies used throughout the description; and a list of annotations for parameters (input values) and responses (output values). Each annotation is a pair to annotate the *name*, introduced by the keyword *@id* (e.g. line 14 and 33), and the *value*, introduced by the keyword *@type* (e.g. line 15 and 34). Annotations are IRIs that uniquely identify elements.

```

1  "basePath": "/books/v1",
2  "paths": {
3  "/volumes": {
4    "get": {
5      "parameters": [{
6        "name": "title", [...]
7      }],
8      "semanticAnnotations": { /** Input semantics */
9        "@context": {
10         "dbp": "http://dbpedia.org/property/",
11         "xsd": "http://www.w3.org/2001/XMLSchema#"
12       },
13       "title": {
14         "@id": "dbp:title",
15         "@type": "xsd:string"
16       }
17     },
18     "responses": {
19       "200": {

```

<sup>3</sup>[json-ld.org/spec/latest/json-ld/#basic-concepts](http://json-ld.org/spec/latest/json-ld/#basic-concepts)

```

20     "schema": {
21       "type": "object",
22       "properties": {
23         "isbn": { "type": "string" },
24         "author": { "type": "string" },
25         "title": { "type": "string" }
26       },
27       "semanticAnnotations": { /** Output semantics */
28         "@context": {
29           "dbp": "http://dbpedia.org/property/",
30           "xsd": "http://www.w3.org/2001/XMLSchema#"
31         },
32         "isbn": {
33           "@id": "dbp:isbn",
34           "@type": "xsd:integer"
35         },
36         "author": {
37           "@id": "dbp:author",
38           "@type": "xsd:string"
39         },
40         "title": {
41           "@id": "dbp:title",
42           "@type": "xsd:string"
43       }, [...]

```

Listing 7.2 Semantic OpenAPI description of the Google Books API.

### 7.1.2 Composition Types and Rules Analysis

In this context, we consider the composition of information services and the interest in mashing up results from independent services to deliver a comprehensive answer to users' requests, or to prepare data coming from a set of services to invoke another service. We call the former *merge composition* and the latter *sequence composition*. Merge composition involves more services that are invoked in parallel with the same input data, and the results are composed [75]; while sequence composition involves a service which is invoked with input data that are coming from one (data adaptation) or more (data mash-up) services.

When dealing with automatic *sequence composition*, semantic compatibility needs to be verified. In this context, semantic compatibility occurs when a semantic relationship holds up between the semantic classes<sup>4</sup> of output properties of an API and input parameters of another API. In such cases, output properties can be used as input parameters, possibly after some transformations (Figure 7.1).



Figure 7.1 Schema of sequence composition.

To evaluate semantic compatibility, we can define four rules:

**Rule 1: single ontology, same concepts.** If annotations refer to the same ontology, and name/value pairs refer to the same concept, or two concepts in relation *owl:sameAs*,

<sup>4</sup>[www.w3.org/TR/owl2-syntax/#Classes](http://www.w3.org/TR/owl2-syntax/#Classes)

then the composition is straightforward since they are compatible. If the property and the parameter of two different descriptions contain the same semantic annotation, it means that they are composable. This case is the simplest one since the presence of the same annotation indicates that the property of an API and the input parameter of another API refer to the same class. As an example, see the composition shown in Figure 7.2.1. The API 1 produces as output an object *Book* with one property annotated with the concept `@id: dbo:author`, and the API 2 has a input parameter with the same annotation. In this case, the *Book* produced by API 1 represents the same concept of *Book* which is required as input from API 2, thus making the composition possible. If the parameter and the property have been annotated with `@type: xsd:string`, then of type literal, it is necessary to carry out a further check, considering the parameter `@id`. If the two APIs have the same value for the `@id` parameter, then they are composable.

**Rule 2: different ontologies, same concepts.** If annotations refer to different ontologies (see Figure 7.2.2), we need to verify if the annotations of involved name/value pairs are *equivalent* (i.e. they refer to the same ontology concepts or property). For example, some ontologies such as DBPedia<sup>5</sup> and Wikidata<sup>6</sup> provide the properties `owl:equivalentProperty` and `owl:equivalentClass` to address the issue. These properties, however, are not supported by all ontologies, therefore some *Ontology matching* [34] techniques may need to be exploited to check for compatibility.

**Rule 3: single ontology, different concepts in relation to each other.** If annotations refer to the same ontology, and name/value pairs refer to different ontology concepts or properties, then values' compatibility need to be checked. If between the involved concepts relations such as *subclass* and *subproperty* hold, then they may be compatible and the composition may occur. An example is shown in Figure see Figure 7.2.3, where the annotation `@type: dbp:zipCode` refers to a subproperty of `dbp:postalCode`. Therefore, API 1 and API 2 are compatible.

**Rule 4: different concepts not related to each other.** If annotations of the name/value pairs refer to different ontology concepts or properties in the same ontology or different ontologies, and among these elements none of the above rules apply, compatibility may occur after a transformation (e.g. by invoking a third-party service). For example (see Figure 7.2.4), if API 1 returns a address, and API 2 requires latitude and longitude values as input parameters, then a third API is needed to perform the conversion.

---

<sup>5</sup>dbpedia.org

<sup>6</sup>www.wikidata.org



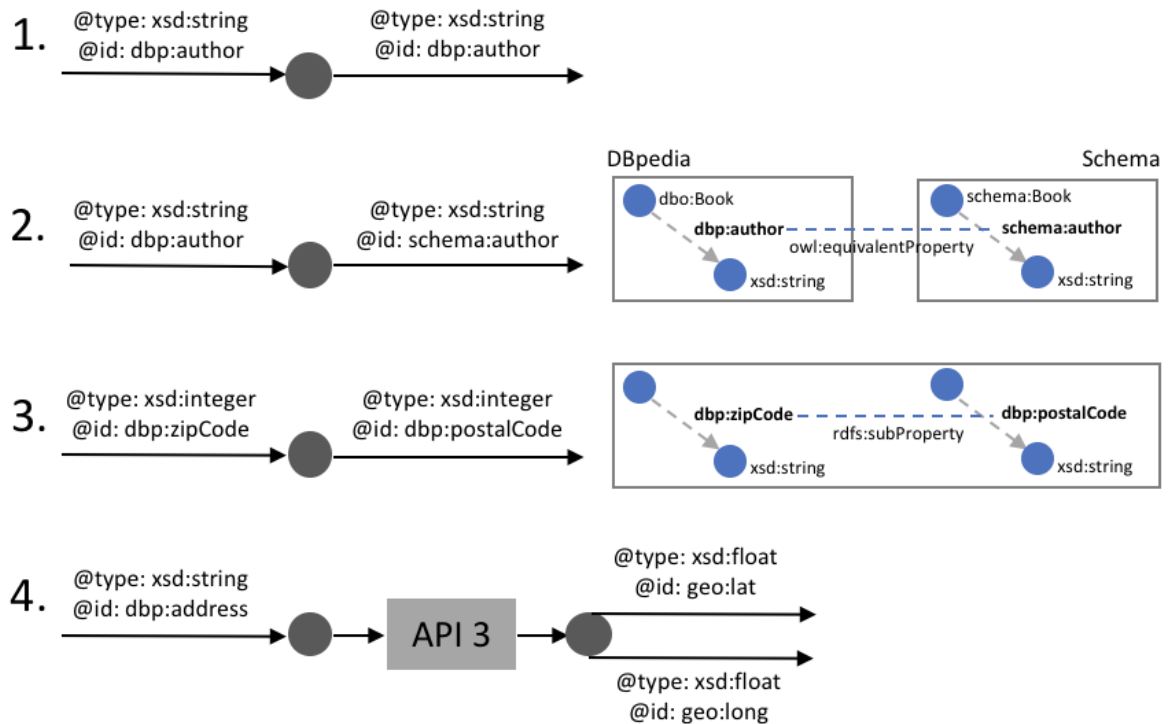


Figure 7.2 Sequence composition: examples of the four compatibility cases.

As previously described, two different methods of composition have been defined: *sequence* and *merge*. In *sequence* method is the most straightforward composition mode because it follows a linear workflow. This method happens when, all or only part of the output of an API, it is used as input to another API. To ensure a high level of automation, it is essential that the API output contains all the information needed to satisfy the input requests of the second API. If this condition is not met, it is possible to invoke an additional API that allows the conversion or integration of the data necessary to invoke the second API. The *merge* method, or composition of the outputs, is more complex since the composition is built on the outputs obtained by invoking the APIs. More precisely, the results obtained by invoking the second API are filtered through the output obtained from the first API.

Let's consider a use case to discuss the composition rules described above. Assume we seek an application that helps students to retrieve information to access textbooks. The application should provide information about different options: bookshops or e-commerce purchase, library consultation, or free download. The composition related to this use case is shown in Figure 7.3: we consider a process that starts with Google Books API, which gets a title in input and delivers a full report about accessing the requested book in output.

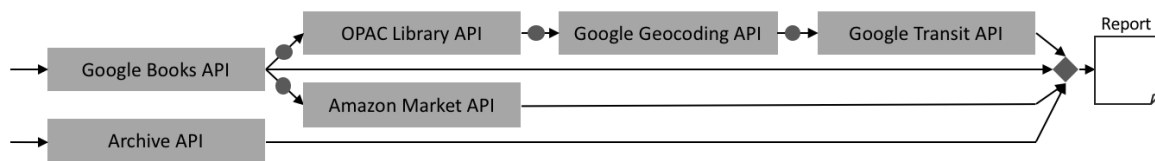


Figure 7.3 Example of a process of composition of the use case.

A first example of *sequence* composition type, is the service that collects information about a book from Google Books API<sup>7</sup> and calls Amazon Market API<sup>8</sup> to check if it is available. The Semantic OpenAPI Description of Amazon Market API is in Listing 7.3. The semantic annotation in line 6 finds a correspondence in the description of the Google Books API, in line 33 of Listing 7.2; in both descriptions the concept of *ISBN* is described with the same semantic annotation. Therefore, the services can be composed (rule 1).

```

1 "get": {
2   "parameters": [{
3     "name": "IsbnItem", [...]
4   }],
5   "semanticAnnotations": {
6     "IsbnItem": {
7       "@id": "dbp:isbn",
8       "@type": "xsd:integer"
9     }, [...]
```

Listing 7.3 The input part of the description of the Amazon Market API.

A second example is the sequence composition of the Google Books API, the Library API, and the Google Transit API: first the Library API is invoked to check the presence and availability of the book, and then the Google Transit API is invoked to check the existence of public transport to reach the library.

The composition of Google Books API and the Library API can be performed according to rule 1, and rule 3. The annotations on line 8 and line 16 of Listing 7.4 are compatible with the annotations in line 32 and 40 of Listing 7.2 (rule 1). The parameter on line 12 of Listing 7.4 is compatible with the property present in line 36 of Listing 7.2 since the relation `rdfs:SubPropertyOf` holds between them (rule 3).

```

1 "get": {
2   "parameters": [
3     { "name": "Isbn" },
4     { "name": "author" },
5     { "name": "title" }
6   ],
7   "semanticAnnotations": {
8     "Isbn": {
9       "@id": "dbp:isbn",
10      "@type": "xsd:integer"
11    },
```

<sup>7</sup>[developers.google.com/books/](http://developers.google.com/books/)

<sup>8</sup>[developer.amazonservices.it/gp/mws/docs.html](http://developer.amazonservices.it/gp/mws/docs.html)

```

12     "author": {
13         "@id": "dbp:writen",
14         "@type": "xsd:string"
15     },
16     "title": {
17         "@id": "dbp:title",
18         "@type": "xsd:string"
19     }, [...]

```

Listing 7.4 Extract from the description of the Library API.

The composition between the Library API and the Google Transit API cannot be performed directly because the first API returns the mail address of a library in text format, while the Google Transit API gets geographic coordinates as input. For this reason, between the two compositions, a third API (Google Maps geocoding API) is used to perform geocoding (rule 4). Listing 7.5 shows the annotations of the Google geocoding API.

```

1  "get": {
2      "parameters": [
3          { "name": "address" }
4      ],
5      "semanticAnnotations": {
6          "address": {
7              "@id": "dbp:address",
8              "@type": "xsd:string"
9          },
10     }
11 },
12 "responses": {
13     "200": {
14         "location": {
15             "properties": {
16                 "lat": { "type": "number" },
17                 "long": { "type": "number" }
18             },
19             "semanticAnnotations": {
20                 "lat": {
21                     "@id": "dbp:latitude",
22                     "@type": "xsd:float"
23                 },
24                 "long": {
25                     "@id": "dbp:longitude",
26                     "@type": "xsd:float"
27                 }, [...]

```

Listing 7.5 Extract from the description of Google geocoding API.

Now that all the information about the different ways to get access to the textbook have been collected, we can compose the results to deliver the requested report to the user.

Dealing with *merge composition*, we need to verify the semantic compatibility of at least two different outputs (Figure 7.4).

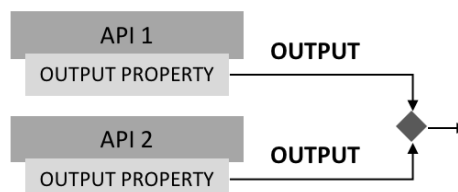


Figure 7.4 Schema of merge composition.

To evaluate semantic compatibility in the merge composition, we can define an additional rule:

**Rule 5: concepts as unique identifiers.** If two or more descriptions share compatible concepts (i.e. they are linked by properties like `owl:sameAs`, `owl:equivalentClass`, `rdfs:subClassOf`, or `rdfs:subPropertyOf`), and these concepts uniquely identify the represented resources (e.g. ISBN for a book, VAT ID for a company, BARCODE for a products), then the outputs of the APIs can be merged.

The Listing 7.6 is a fragment of the Archive.org API<sup>9</sup> description; as shown in line 10, 14, 18, respectively the annotation of the output properties, ISBN, title, author; it is possible to observe how these properties are compatible with the response of Google Books API (Listing 7.1). According to rule 5, the merge composition can occur if compatible properties allow us to conclude that outputs refer to the same resources. In the use case, the *ISBN* can be adopted as unique identifier for books, thus allowing composition of outputs into the final comprehensive report.

```
1 "200": {
2   "Book": {
3     "type": "object",
4     "properties": {
5       "ISBN": { "type": "string" },
6       "title": { "type": "string" },
7       "author": { "type": "string" }, [...]
8     },
9     "semanticAnnotations": {
10      "ISBN": {
11        "@id": "dbp:isbn",
12        "@type": "xsd:integer"
13      },
14      "title": {
15        "@id": "dbp:title",
16        "@type": "xsd:string"
17      },
18      "author": {
19        "@id": "dbp:author",
20        "@type": "xsd:string"
21      }, [...]
22    }
23  }
24 }
```

Listing 7.6 Extract from the output part of the description of the Archive API.

---

<sup>9</sup>[blog.archive.org/developers/](http://blog.archive.org/developers/)

# Chapter 8

## AutomAPIc Tool

### 8.1 Architecture and Process

AutomAPIc is a comprehensive tool created to manage semantic descriptions and input/output composition of services. In this Section, we concentrate on the the description editor, which supports semi-automatic creation of semantic descriptions, and automatic composer, which supports compatibility matching. AutomAPIc is available via Git repository<sup>1</sup>. The Figure 8.1 shows the architecture of the tool.

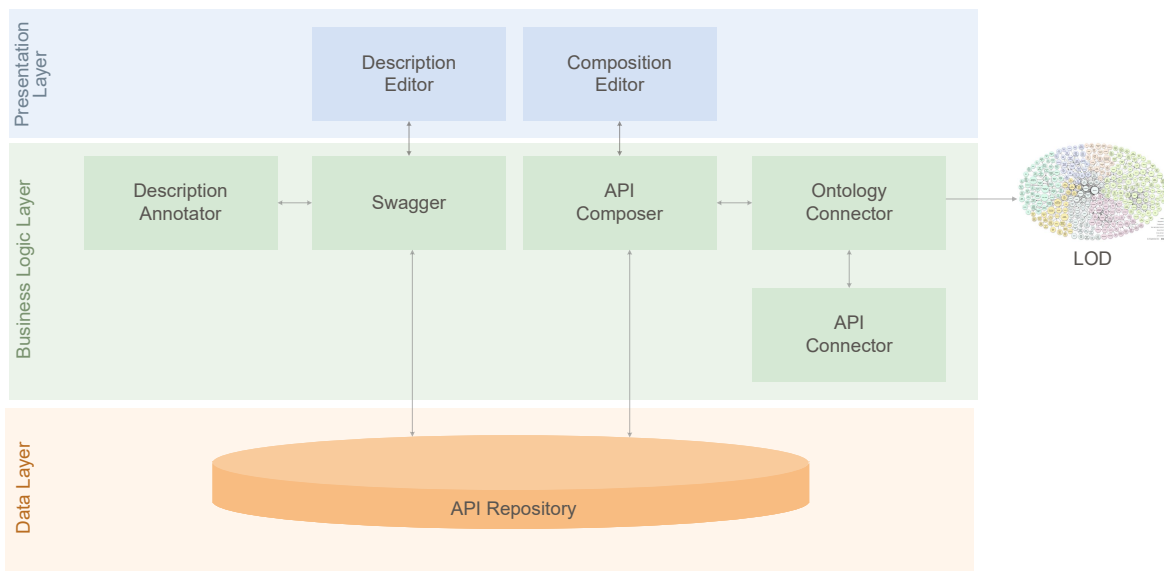


Figure 8.1 Architecture of AutomAPIc tool.

It is possible to identify 6 main components: (i) Description Editor, useful for the definition and management of API descriptions in OpenAPI format; (ii) Description Annotator,

<sup>1</sup>[bitbucket.org/disco\\_unimib/automapic-tool/](https://bitbucket.org/disco_unimib/automapic-tool/)

useful for adding semantic annotations; (iii) Composition Editor, which allows for the selection of a set of composable APIs by the user; (iv) API Connector, a component for automatic identification of the composable APIs in relation to the composition rules described above; (v) Ontology Connector, a component that extracts semantic relations by performing queries to the LOD Cloud through SPARQL; (vi) Composer API, useful for the execution of the composition previously defined by the user.

### 8.1.1 Getting OpenAPI Descriptions

The description process is semi-automatically managed by augmenting existing API descriptions, which can be retrieved from existing repositories (e.g., ApisGuru, SmartAPI), or created manually using the Description Editor. These descriptions are represented in JSON or YAML format, and include all relevant information such as available HTTP operations, the list of input parameters and output responses for each operation. The process of creating a description is detailed in Algorithm 1.

---

**Algorithm 1:** Retrieve or create API description.

---

**Result:** API description

```
1 if description is available then
2   | retrieve description from existing repositories and registries of services;
3 else
4   | create it manually using the Description Editor;
```

---

### 8.1.2 Adding Semantic Annotation

If semantic annotations are missing, we need to annotate input and output data. To annotate output data, AutomAPIc provides users with a service that collects a set of output values of GET calls into a table and apply Semantic Table Interpretation.

Table interpretation consists of associating data with semantic concepts in an ontological structure, within the LOD Cloud, which aims to represent the knowledge of a certain domain through the connections that exist between these same elements. The GET method is mainly considered since it is the most frequent. In this way API's parameters and properties can be managed by a computer. The code related to the Table Interpretation technique used in this proposal is available through a Git repository<sup>2</sup>.

The input parameters are annotated differently because it is not possible to transform the parameters into a table. AutomAPIc provides a service based on *Natural Language*

---

<sup>2</sup>[bitbucket.org/disco\\_unimib/mantistable-tool/](https://bitbucket.org/disco_unimib/mantistable-tool/)

*Processing* [16] techniques. In particular the *Stanford CoreNLP* tools<sup>3</sup> [63] has been adopted. These tools provide several libraries that allow for the extraction of entities from API descriptions, which will then be associated with concepts. The application of these techniques on hundred descriptions from the repository *APIs.guru* led to the correct identification of entities and properties for 93% of the cases. Algorithm 2 defines the process to insert semantic annotations in API descriptions.

---

<sup>3</sup>[stanfordnlp.github.io/CoreNLP/index.html](https://stanfordnlp.github.io/CoreNLP/index.html)

This algorithm revises and extends the one presented in [18].

---

**Algorithm 2:** Create and add semantic annotation to API descriptions.

---

**Data:** API description

**Result:** API description with semantic annotations

```

1 Detect all resources' end-point;
2 foreach end-point do
    | // collect data
3   repeat
4     | generate input parameters following the API description;
5     | generate semantic annotation of the input parameters using NLP technique;
6     | insert semantic annotation of the input parameters in API description;
7     | if input parameters cannot be generated then
8       | | take input parameters from the user
9       | | invoke API with input parameters;
10      | | collect results;
11   until at least N results are collected;                               /* default N=10 */
    | // create tables
12   foreach results do
13     | create a header row with API properties;
14     | fill content-cells with values from inputs and responses;
    | // add semantic annotations
15 foreach tables do
16   | apply MantisTable approach;
17   | show table to the user;
18   | if table annotation is not complete then
19     | | show related vocabularies and/or alternatives to the user;
20     | | ask the user to manually add links;
21   | if the user wants to review the annotations then
22     | | show related vocabularies to the user;
23     | | let the user confirm or modify the links;
24   | insert semantic annotation of properties in API description;

```

---



### 8.1.3 Performing Automatic Composition

The presence of semantic annotations allows the automatic identification of the composable APIs given a starting API. The API composer component automatically shows the compatible APIs. The possible combinations have been previously calculated by the API connector, through the use of SPARQL queries, in order to apply the compatibility rules (Algorithm 3).

---

**Algorithm 3:** Identification of compatibility between the APIs.

---

**Result:** Composed APIs

- 1 inserting a new API into the system;
  - 2 parsing of the description;
  - 3 extraction of semantic annotations;
  - 4 **foreach** *APIs* **do**
  - 5     creation and execution of SPARQL queries to identify the relationships between the annotations of the APIs;
  - 6     update the graph of possible compositions;
-

## 8.2 Validation

To verify the validity of the composition approach we propose, we collected a set of APIs (Table 8.1) for the creation of a *benchmark* with characteristics that cover all possible cases. The chosen APIs comes from various domains, including public transport, films, books, music and events.

Table 8.1 Validation dataset.

<i>API</i>	<i>Description</i>	<i>Source</i>
GEOCODING	Converts an address into latitude and longitude	Google Maps
MARINE CONDITION	Forecast of marine conditions	World Weather Online
WEATHER FORECAST	Weather forecasts	Weather Underground
PHOTOS	Photos geolocated in a specific position	Flickr
NEWS	List of news	NewsAPI
BOOK	List of information about a book	Google
MOVIE	List of information about a film	OMDb API
POI	Points of interest of a city	Syctic API
LIBRARY	List of information about the availability of a book	Opac Unimib
E-COMMERCE	Information regarding the price of a product	Amazon Market
FREE EBOOK	Information on the presence of a free eBook	Archive.org
PLAYLIST	List of songs contained in a playlist	Spotify
LYRICS	Text of a song	Musixmatch API
FLIGHTS	Airport information	Ryanair API
BIKE SHARING	List of bicycles available	City Bike
EVENTS	List of events in a city	EventiFul
HOTEL BOOKING	List of hotels available on a specific date on a certain day	HotelsCombined API
REVIEWS	List of reviews of places and events	TripAdvisor Content API
PUBLIC TRANSPORT	List of information about public transport in a particular place	Google Transit
RESTAURANTS	List of restaurants in a specific city	Zomato API

In a second phase the descriptions and their annotations were analyzed, to identify the possible compositions. Through a combinatorial calculation it is possible to calculate the maximum number of combinations. In particular, given 20 APIS, using provisions without repetitions (since an API cannot be composed with itself), the maximum number of compositions is 380.

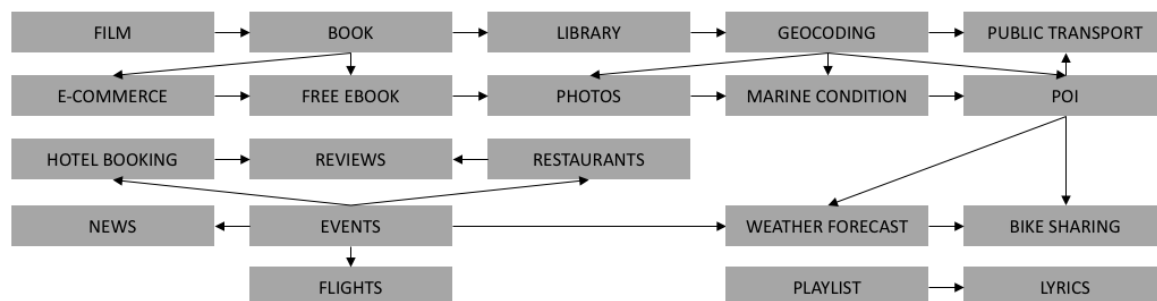


Figure 8.2 List of the possible compositions.

As shown in Figure 8.2, depending on parameters and annotations, the actual combinations are twenty four. AutomAPIC was able to identify the 85% of them. Table 8.2 reports the *confusion matrix* of the results, where attributes are: (i) TP: number of correctly composed APIs, (ii) FP: number of APIs that were composed but which should not be composed, (iii) FN: number of APIs that were not composed but that had to be composed, (iv) TN: number of APIs that were not to be composed and were not composed. The *accuracy* of the system is  $(TP + TN)/Total = 0.99$ . Going into detail, the combinations that led to composition failures are mainly three: weak support to manage concepts connected by the owl :subProperty relation, incomplete relationships between ontologies (e.g. DBpedia and KBpedia), and inaccurate semantic annotations of parameters returned by table interpretation techniques.

Table 8.2 Confusion matrix.

Tot. = 380	Composed	Not - Composed
Composed	<b>TP = 17</b>	<b>FP = 0</b>
Not - Composed	<b>FN = 3</b>	<b>TN = 360</b>



## **Part III**

# **Natural Language Generation with Neural Networks**



# Chapter 9

## State-of-the-art

NLG is a “sub-field of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information” [85, 86].

NLG’s field of action is about creating a textual output that contains information from various formats of input resources, like tables, graphs, numeric data and many others [68]. Its aim is to make the output text as indistinguishable as possible from text generated by a human author. Modern technologies also allow more sophisticated and complex outputs, like interactive info-graphics and conversational interfaces for chatbots. It is rather challenging to give a precise definition of NLG, because if the output of these systems’ classes is text, the input may vary, so we can distinguish data-to-text generation and text-to-text generation. In the former, input data could include semantic representations, numerical data and structured knowledge bases. Instead, examples of text-to-text generation are machine translation, summarization or simplification of text and paraphrases generation [37].

We can classify NLG technologies in different subfields, according to different communication goals and different formats of data received as input. To understand the functioning of NLG algorithms we must remember that, given structured data as input, there are three logical steps to achieve output text: the first is the *planning of information* that we want to express; the second is *the concatenation of information* in the output text; the third is *the actual expression of the designed messages*.

A common practice in Natural Language Generation is dividing the process in several sub-task, to make them more straightforward to manage. The architecture proposed by Reiter and Dale [85] is constituted by a pipeline divided into different steps. According to [37], it is possible to identify six primary tasks:

- Content determination - What to say?
- Text structuring - Which is the preferable order?
- Sentence aggregation - Could some phrases be gathered together?
- Lexicalization - Through which words the information can be expressed?
- Reference expression generation - What does a word refer to?
- Linguistic realisation - How can be generated the right morphological forms?

In the following Sections, for each task, we will present a brief description and some examples extracted from a scenario concerning the highest mountains.



## 9.1 Sub-tasks of Natural Language Generation

**Content Determination .** As mentioned before, in the beginning of the Natural Language Generation process we start from data (RDF). Input data is always more detailed and richer compared to what we want to cover in the text [37]. So, one of the first aims is to filter and choose what to say. It's probably the most important task in this process. The content of a text is particularly relevant to users, because they pay a lot of attention on what they're reading. If there are some mistakes or some linguistic form problems, readers can still understand text, however when content is incomplete the whole meaning might not be understood [29]. In this phase, data are filtered and abstracted to form a set of preverbal messages, often represented in a formal language, such as logical or database languages, attribute-value matrices or graph structures [37]. Usually, approaches are related to the domain of application. That being said, in the last years, researchers have focused on data-driven techniques [37], a trend that we observe in almost all the NLG task. The reason for that is that domain-dependent and handcrafted techniques are usually more time-consuming and challenging to implement in other domains. Data-driven approaches, when implementable, are the most elegant solution to reduce time and to design more versatile systems. Starting from the key idea that clustering could help to develop a content determination strategy, many authors tried to extract rules from the comparison between text and data. Some researches used Hidden Markov Models, a class of probabilistic graphical models that enables the prediction of a sequence of the unknown variable from a set of existing ones. These models are used to cluster topics together (represented as hidden states), for example, in earthquake domain [7]. Other authors used texts paired with a knowledge base in order to understand main rules in the biography domain. The idea behind this approach is that a system can automatically learn content selection rules from human written corpus data: the algorithm compares desired outputs with related semantic data in order to understand which input is relevant [29]. Often, the target text is explicitly written to make a comparison with data, while other times researchers used Wikipedia articles associated with metadata provided by Wikipedia itself [110]. In any case, human written corpora helps to understand what is relevant and meaningful.

Listing 9.1 shows an example of content determination results, encoded in RDF/XML. We are selecting some relevant information about Mont Everest, to prepare a text to describe it.

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <rdf:RDF
3     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5     xmlns:dbo="http://dbpedia.org/ontology/" >
6
7     <rdf:Description rdf:about="http://dbpedia.org/resource/Mont_Everest">
8         <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
```

```

9     <dbo:firstAscendYear rdf:datatype="http://www.georss.org/georss/point">
10     1956
11 </dbo:firstAscendYear>
12 <dbo:moutainRange rdf:resource="http://dbpedia.org/resource/Himalayas" />
13 <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
14     8848
15 </dbo:elevation>
16 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
17     Mont Everest
18 </rdfs:label>
19 <rdfs:comment xml:lang="en">
20     Mount Everest, known in Nepali as ...
21 </rdfs:comment>
22 </rdf:Description>
23
24 <rdf:Description rdf:about="http://dbpedia.org/resource/K-2">
25 <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
26 <dbo:firstAscendYear rdf:datatype="http://www.georss.org/georss/point">
27     1954
28 </dbo:firstAscendYear>
29 <dbo:moutainRange rdf:resource="http://dbpedia.org/resource/Karakoram" />
30 <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
31     8611
32 </dbo:elevation>
33 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
34     K-2
35 </rdfs:label>
36 <rdfs:comment xml:lang="en">
37     K2, also known as Mount Godwin-Austen or Chhogori, is ...
38 </rdfs:comment>
39 </rdf:Description>
40
41 </rdf:RDF>

```

Listing 9.1 Example of the output of Content Determination in RDF/XML format.

When the content determination is complete, the next step is text structuring.

**Text Structuring.** A clear text structure and the order of presentation of information are critical for readers. The second phase of Natural Language Generation, so, is about the order of presentation to the reader. There are different possible parameters to determine the disposition, such as temporal, based on importance and based on relatedness. The application domain may force constraints on ordering choices. For this reason, pre-defining the templates is necessary. For example, in football, it could be useful to give a general description during the first phase, and after the match to report events in a chronological order [107]. In neonatal care domain, instead, specific events could be highlighted, and others could be grouped [79]. In the history of approaches for text structuring, we can see three main stages of developments: a handcrafted past, a Rhetorical Structure Theory (RST) stage and a more recent Machine Learning approach. The first attempts relied on domain-dependent and handcrafted structuring rules (called schemata, a sort of template). The advantage was that its straightforward implementation, but it was strictly domain-dependent and could not be extended to other fields. The second phase was based on the idea that order could arise from relations between messages. Many authors focused on the so-called Rhetorical Structure Theory, which uses a hierarchical structure to understand relations among parts of the text, even though this approach is also domain-dependent and relies on specific rules. The last and current phase of the text structuring task is based on machine learning techniques [37].

After this phase, our data present a more clear structure and a defined order (Listing 9.1).

```

1 <rdf:Description rdf:about="http://dbpedia.org/resource/Mont_Everest">
2   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
3   <dbo:firstAscendYear rdf:datatype="http://www.georss.org/georss/point">
4     1956
5   </dbo:firstAscendYear>
6   <dbo:moutainRange rdf:resource="http://dbpedia.org/resource/Himalayas" />
7   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
8     8848
9   </dbo:elevation>
10  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
11    Mont Everest
12  </rdfs:label>
13  <rdfs:comment xml:lang="en">Mount Everest, known in Nepali as ...</rdfs:comment>
14 </rdf:Description>

```

Listing 9.2 The output of text structuring for data seen in Listing 9.1.

After this step, information need to be aggregated.

**Sentence aggregation.** Having defined the structure and the contents, we need to group the text blocks. The definition of this step is not always clear in literature: some authors see it as redundancy elimination, but it could also be considered as a linguistic structure combination. Anyway, the final aim is to have a fluid and readable text. As for the other tasks, we observe a shift from handcrafted domain-dependent methods to the recent data-driven approach where the comparison with corpus data allows to pinpoint linguistic rules [37]. Paralleling between the corpus of sentences and a corresponding database can make a system learn how to aggregate, combined with grammar constraints and global rules (some examples of this could be “which order should have the sentence” and “how many sentences should be aggregated in a document” [6]). Recently, some authors have started to consider an automatic acquisition of rules from text. It is interesting to note that sometimes aggregation can be avoided to enforce emotion or comprehension through redundancy [37].

After the sentence aggregation step, our information will be structured (Listing 9.3).

```

1 <rdf:Description rdf:about="http://dbpedia.org/resource/Mont_Everest">
2   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
3   <dbo:moutainRange rdf:resource="http://dbpedia.org/resource/Himalayas" />
4   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
5     8848
6   </dbo:elevation>
7   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
8     Mont Everest
9   </rdfs:label>
10  <dbp:highest rdf:resource="http://dbpedia.org/resource/Himalayas" />
11 </rdf:Description>
12 <rdf:Description rdf:about="http://dbpedia.org/resource/Himalayas">
13   <dbo:country rdf:resource="http://dbpedia.org/resource/Bhutan" />
14   [...]
15 </rdf:Description>

```

Listing 9.3 The output of sentence aggregation. Note that, from the DBpedia resource “Himalayas”, we retrieved new knowledge about the countries where it is located.

In the next phase, our data will undergo the lexicalization process.

**Lexicalisation .** One of the most critical phases of NLG process is about how to express message blocks through words and phrases. Once the content of the sentence is defined, it is possible to convert it in natural languages. This task is called lexicalisation. This process involves the decision of which words and phrases are to be used. It is not easy to make this choice because a single event can be expressed in many different ways. One of the aims is the generation of a non-repetitive text, by varying the chosen wording to express something. To achieve this, the system can randomly select a lexicalisation option among a set of possible solutions. There are, however, different constraints, not only the contextual ones but also the stylistic ones: not all the possible ways to express a concept can elegantly describe the situation in question [37]. When we deal with well-defined domains, lexicalisation can operate on preverbal messages, directly converting concepts into lexical items. In many other cases, however, lexicalisation encounters more difficulties. One is that it can involve selection between semantically similar, near-synonymous or taxonomically related words. Another is the potential vagueness which we encounter when we deal with terms indicating gradable properties, such as adjectives like “wide” or “tall”. In this case, the system would have to reason about the relative value of the property when the object is compared to other members of the same class (for example “tall glass” is shorter than a “short man”) [37]. Gatt [37] notes that many issues related to lexicalisation have also been discussed in the psycholinguistic literature about lexical access. For years, researchers have investigated on how speakers choose the right word to use, and when they tend to make errors, given the complexity of the mental lexicon. It is a densely connected network and contains items with different levels of connection between different kinds of information.

Lexicalisation will produce an output like in Listing 9.4.

```
1 Mount Everest belongs to the Himalayan mountain range.
2 Mount Everest rises 8848 m above sea level.
3
4 Mount Everest is the highest peak in the Himalayas.
5
6 Himalayas is located in Bhutan.
7 Himalayas is located in China.
8 Himalayas is located in Pakistan.
9 Himalayas is located in Nepal.
10 Himalayas is located in India.
```

Listing 9.4 Output of lexicalization process has a more human-friendly form.

This text, already understandable by humans, has to be further refined if we want to present it in an optimal fashion.

**Reference Expression Generation .** To avoid repetitions, selecting ways to refer to entities using different methods (such as pronouns, proper nouns, or descriptions) is essential. Referring Expression Generation (REG) is defined as “the task of selecting words or phrases

to identify domain entities” [85]. When the system needs to decide how to refer to an entity, it needs to consider if they have already been mentioned. In this case, a pronoun can be used to indicate the entity, but another thing that has to be considered is whether they need to be distinguished from similar entities. If that is the case, the system performing Referring Expression Generation has to convey enough information to allow the receiver of the message to discriminate one entity from others of the same domain. The way by which entity is expressed, be it with a pronoun, a name or a description, is called referential form. In every part of the text, to choose the referential form, it is essential to identify which is the salient entity, i.e. the one “in focus”. When the chosen form is a description, we need to determine what is called the referential content. This can be seen as a search through the known properties of the referent until the right combination is found. The ideal formula is the one that will distinguish the entity from the others, following the Gricean maxim of quantity [40], which states that speakers have to be as informative as possible and to give as much information as is needed, but no more than that. Different algorithms have accomplished this task in various ways. Some of them, like the one of Full Brevity Procedure [23], conduct a thorough search through the possible description, choosing the least number of properties that can guarantee discrimination. Others select the properties incrementally, choosing every time the one that rules out most distractors. In this way, there is less possibility of including no relevant information. This type of solution is implemented in the Greedy Heuristic algorithm [23]. A third possibility is to select properties incrementally but on the base of domain-specific preference or cognitive salience [24].

Having undergone the REG process, our output will be like in Listing 9.5.

```
1 Mount Everest belongs to the Himalayan mountain range.  
2 It is rises 8848m above sea level.  
3 It is the highest peak in the Himalayas.  
4 Himalayas is located in Bhutan, China, Pakistan, Nepal and India.
```

Listing 9.5 The output of REG: "Mount Everest" has been replaced with anaphoras in two sentences.

**Linguistic Realisation .** Linguistic realisation is all about combining relevant words and phrases to form a sentence. To perform linguistic realisation, systems need to choose an order of constituents of the sentences. Moreover, the right morphological forms, as verb conjugations, have to be generated, together with function words such as auxiliary verbs and prepositions. An issue within this process is represented by the fact that output will comprise some linguistic components that may not be present in the input (this is a case of what is called “generation gap”) [37]. Historically, there have been different approaches to the linguistic realisation:

- human-crafted templates;

- human-crafted grammar-based systems;
- statistical approaches.

The templates approach is convenient when the domain of application is small. In this case, the realisation can be relatively easy, and templates can be used to have full control over the quality of the output, avoiding the generation of poorly grammatical sentences. Recent versions of the template-based approach incorporate syntactic information and complex rules for filling the gaps. Thus, these methods are becoming difficult to distinguish from more sophisticated approaches [25]. The main disadvantage of these systems is the hand-work required during their realisation [37]. Unlike the templates ones, grammar-based realisation systems are general-purpose and domain-independent. These approaches make their choices following a hand-written grammar that refers to the language under consideration. One of the problems of this kind of approach is the difficulty to create hand-crafted rules with the right sensitivity to context [37]. The idea at the foundation of statistical approaches is to obtain probabilistic grammars by acquiring them from large corpora. The main advantage is the critical reduction of manual labour and the maximisation of coverage.

With linguistic realisation, the process of converting RDF information in natural language is finally complete (Listing 9.6).

```
1 Mount Everest belongs to the Himalayan mountain range and rises 8848m above sea level.  
2 It is the highest peak in the Himalayas that it is located in Bhutan, China, Pakistan,  
3 Nepal and India.
```

Listing 9.6 The final output of NLG.

As we stated above, lexicalisation is one of the most critical and complex tasks in the NLG process. Natural language vagueness and the need to choose the right words to express a concept are intricate issues to manage. Analysing the state-of-the-art, we see that recent research on this topic shows that an attractive solution in these cases is based on Machine Learning (ML) [37]. Moreover, a recent challenge in the NLG field, launched and published in 2017, called WebNLG [36] confirms the idea that not only we need to combine ML methods to generate language, but we can also use KGs to enrich sentences with additional contextual information (e.g. contextual information about a player).

## 9.2 Application Scenarios in Natural Language Generation

With the enormous variety of data that can be given in input to a NLG algorithm, we are not surprised to notice many application scenarios in which a data-to-text generation can be extremely useful. The main advantages come from the possibility to free humans of repetitive

tasks, allowing them to focus on more specific and stimulating activities. Application Scenarios in NLG include [37]:

- *Business Intelligence*: currently, there are many tools for data management and analysis, but there is not a standardized technology that can adapt to the custom needs of businesses. With NLG techniques, every company can autonomously interpret its data and manage it with a practical dashboard;
- *Finance and Insurance*: each client is unique, he has his own needs and takes different risks to fulfill them. With NLG, a consultant can produce an automatic generation of the personalized report even without knowing the story of the client;
- *Health and Science*: in clinical contexts, NLG technologies can swiftly generate summaries of patient information;
- *Media and Entertainment*: from the possibility of a personalized communication via e-mail and in-app to the generation of a report for sports performances and results: data-driven text generation could be a game-changer technology;
- *Industry*: the advantages in this field go from the generation of landing pages on e-commerce sites to automatic report for data from IoT sensors.

After an analysis of the possible solutions that can be used for the implementation of Natural Language Generation, the ones employing Neural Network turned out to be particularly promising [37], in particular for the lexicalization task. Since it is a supervised approach, however, it is necessary to create a proper training dataset. In the next Section will be presented a review of the main state-of-the-art datasets.

### 9.3 Training Dataset

The dataset contains an alignment between data triples and natural language. During this training phase, the network learns which outputs have to be associated with determined inputs.

In Table 9.1 we can see a comparison between various state-of-the-art datasets for NLG [33].

NYT-FB is a dataset created with alignments between New York Times articles and Freebase data triples. New York Times corpus contains 1,855,658 articles, published between the 1987 and June 19, 2007. In the first step for the creation of dataset, the Stanford named

Dataset	Authors	Year	KG	Domain	Corpus	Unique predicates	Aligned Triples	Available
NYT-FB	Mintz et al	2009 2011	FB	News from NYT	1.8M sentences	258	39K	partially
TAC KBP	Li et al	2012	Custom	News wire and web forums	1.7M documents	41	122K	closed
Google-RE	-	2013	-	Wikipedia	60K sentences	5	60K	partially
FB15K-237	Toutanova et al	2015	FB	ClueWeb	200M sentences	237	2.7M	public
Wikireadings	Hawlett et al	2016	WikiData	Wikipedia	4.7M articles	884	n.a. (no alignment)	public
T-Rex	Elsahar et al	2018	DBpedia	Wikipedia	3.09M	633	11M	public

Table 9.1 Comparisons between different training datasets for NLG.

entity recognizer is employed to find mentions of entities in text. Each token is classified into four categories: “person”, “organization”, “location” and “none”. Found entities are then associated with Freebase entries, matching entity mention phrases and the canonical names in Freebase. The results are 39K aligned triples with 258 unique predicates, but this dataset is prone to bias and coverage issues since the Named Entity linking used for its generation is based on keyword matching with Freebase labels [33]. This entails that there are many alignments for the same keyword. For example, 30.7% of the alignments are only for the predicate “/location/country”.

TAC KBP [58] is a dataset built from a corpus of of 1.7 million documents, composed of newswire articles, audio transcript and other miscellaneous web data, and a knowledge base, made of 800,000 KB entries. The task of the author was to develop linguistic resources for Named Entity Linking (NEL) system training and evaluation. The main effort is directed towards the creation of a system able to perform cross-lingual NEL (both with English and Chinese). Training and evaluation data are created using human annotations. Training data are represented by sets of queries that correspond to different name mentions of people, organisations or geopolitical entities with high levels of ambiguity. We can see some examples in Figure 9.1.

Each query is characterized by a QueryID, a string corresponding to the name mention of an entity, and a document containing that mention. The document constitutes an important reference as it provides context for the disambiguation in the knowledge base, especially for ambiguous names that could potentially refer to multiple entities. TAC KBP’s dimensions are limited, as they contain only 41 predicates, representing 5 classes. Moreover, the dataset is not publicly available.



```

<query id="EL_CLCMN_02111">
  <name>Abbas Moussawi</name>
  <docid>LTW_ENG_19960311.0047</docid>
</query>
<query id="EL_CLCMN_02173">
  <name>丁一汇</name>
  <docid>XIN_CMN_20030323.0033</docid>
</query>
<query id="EL_CLCMN_02174">
  <name>Yihui Ding</name>
  <docid>XIN_ENG_20030327.0034</docid>
</query>

```

Figure 9.1 Examples of queries in TAC KBP.

Google-RE is a Google dataset created with 60K sentences from Wikipedia, manually aligned with Freebase. It is considered high quality, but it is limited because it is labelled for only five Freebase relations.

FB15-237 is a dataset derived from FB15K, the latter being an adaptation of the Freebase Knowledge Graph. From FB15K, the authors of FB15-237 have removed redundant relations, adding textual mentions derived from the ClueWeb 12 web-scale document collection [103]. The authors then realized a system of knowledge base completion, able to predict new relations (links between entities) from a given knowledge base. In Figure 9.2 we can see an example.

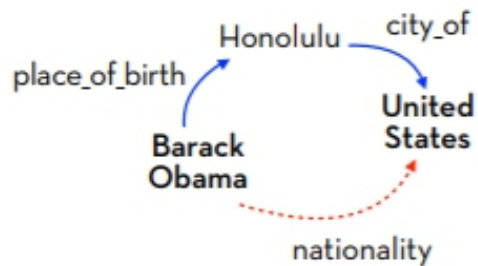
The authors employ statistical models to score the existence of any possible triple, taking into consideration either observed features from the knowledge graph or latent features of the triple's three elements. In this dataset, the knowledge graph is paired with textual relations, derived from sentence co-occurrences of entity pairs. Textual relations are represented as full lexicalized dependency paths. After pruning, FB15-237 includes 2.74 million textual relations, occurring in 3.9 million text.

Wikireadings is a dataset created by altering the Wikidata triples (item, property, answer). In these triples, the subject (item) is replaced with the whole text of the corresponding Wikipedia article. The aim of this dataset is the prediction of textual values from the open knowledge base Wikidata, using corresponding articles on Wikipedia. An example of instance of Wikireadings is seen in Table 9.2:

The approach exploits Recurrent Neural Networks to predict the probability that a word at a given location is part of an answer. Although it has a large size (18 million instances), the dataset does not contain actual alignments between text and KB triples.

One of the broader dataset is T-Rex [33]. This tool has proven to be the one that makes more alignments (it has achieved alignment between over 3 million texts extracted from Wikipedia

## Knowledge Base



Freebase

## Textual Mentions

Barack Obama is the 44th and current President of United States.

Obama was born in the United States just as he has always said.

...

ClueWeb

Lemur

Figure 9.2 An example of relation generation in FB15-237.

<b>Document</b>	Folkart Towers are twin skyscrapers in the Bayrakli district of the Turkish city of Izmir...
<b>Property</b>	country
<b>Answer</b>	Turkey

Table 9.2 An example of instance of Wikireadings.

and 11 million of Wikidata triples). Table 9.1 illustrates the statistics of T-Rex, compared with other datasets.

To create the alignments, T-Rex implements a pipeline with various steps: Document Reader, Entity Linking, Coreference Resolution, Date-Time Linker, Triple Aligner and Document Writer (Figure 9.3) [33]. In the first phase, Document Reader, the input text (abstract of DBpedia) is divided into sentences, for each of which the start character and the end of sentence character are saved. Each of these pairs of integer values is called Boundary. The second phase of the pipeline consists of the extraction of the entities present in the text. T-Rex uses DBpedia-Spotlight [66] as a tool to search the given input text for the various entities and link to the corresponding resource. In the Coreference Resolution phase, pronouns or in general mentions referring to entities previously mentioned are searched for in the text. This

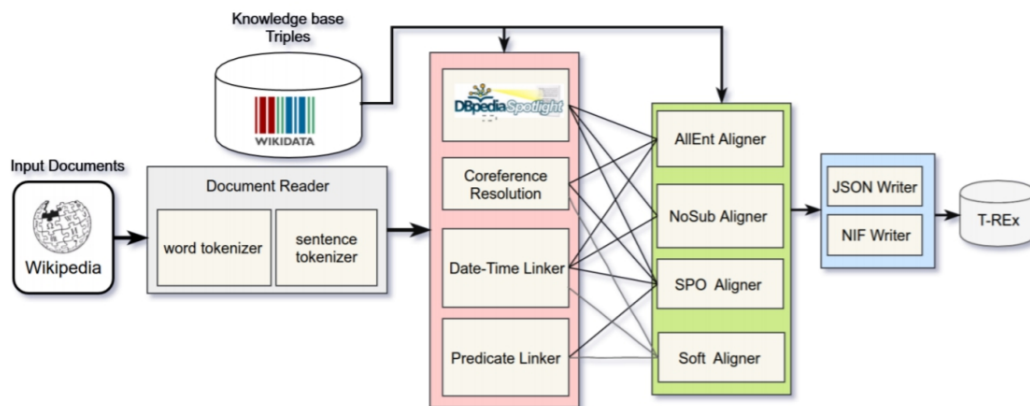


Figure 9.3 The pipeline of T-Rex [33].

step uses Stanford CoreNLP co-reference resolution<sup>1</sup> [63]). For the fourth phase, Date-Time Linker, entities referring to temporal expressions are extracted from the text. T-Rex also in this case uses an external tool, the Stanford temporal tagger SUTime [63]). Triple aligners are the main component of the pipeline. Each “Aligner” deals with aligning a set of triples expressed in the document through their position, or using the Boundaries calculated in the previous phases. In the last phase, Document Writer, the JSON or NIF format is used to represent the text and the related entities and triples aligned.

From the previous comparative analysis, we can identify a series of weaknesses in the datasets currently present in the state-of-the-art. In particular, these shortcomings are identified: (i) low quality alignments, (ii) work on specific domains, (iii) are not freely accessible, (iv) use of ad-hoc Knowledge Graphs. T-Rex [33] is a good resource, but presents some problems in identifying predicates. For this reason, in the next section, starting from the pipeline proposed in T-Rex, we hypothesised a series of improvements, in particular in the predicate alignment part. The result is a new approach called SeaLion.

<sup>1</sup>[www.rangakrish.com/index.php/2019/02/10/coreference-resolution-in-stanford-corenlp](http://www.rangakrish.com/index.php/2019/02/10/coreference-resolution-in-stanford-corenlp)



# Chapter 10

## New Pipeline for triple-text alignment

### 10.1 SeAlion approach

In this section we describe in detail the SeAlion approach, which consists of the following phases:

0. **Data Preparation** aims to clean Wikipedia abstract from brackets, apixes and some characters in order to simplify the text and improve the results of next steps;
1. **Entity Linking** finds a group of entities that are in the processed text. Wikifier and DBPedia-Spotlight are used to perform this task, and from their annotations we keep the union between them;
2. **Literal Extraction** searches for literals using regular expressions (extracted from MantisTable);
3. **Relation Extraction** finds the predicate that connects entities.

SeAlion (where the “se” and the “ali” stand for Sentence Aligner) is inspired by T-Rex [33], described in Section 9.3. T-Rex has been considered a good starting point, and SeAlion is an attempt to enhance and extend its results. SeAlion’s ultimate goal is to generate a dataset that aligns natural language text with a set of structured data (RDF triples). The very first step of the pipeline is the acquisition of text (abstract) from Wikipedia. This text has to be processed to be usable for the following phases.

To describe each phase of the approach we consider the abstract extracted from a Wikipedia page about the K2 mountain<sup>1</sup> (Listing 10.1).

---

<sup>1</sup>[wikipedia.org/wiki/K2](http://wikipedia.org/wiki/K2)

```

1 K2, also known as Mount Godwin-Austen or Chhogori, is the second highest mountain
2 in the world, after Mount Everest, at 8,611 metres (28,251 ft) above sea level.
3 It is located on the China-Pakistan border between Baltistan, in the region
4 Gilgit-Baltistan of northern
5 Pakistan, and the Taxkorgan Tajik Autonomous County of Xinjiang, China. K2 is the
6 highest point of the Karakoram range and the highest point
7 in both Pakistan and Xinjiang. K2 is known as the Savage Mountain due to the extreme
8 difficulty of ascent. It has the second-highest fatality rate among the eight
9 thousands. With around 300 successful summits and 77 fatalities, about one person
10 dies on the mountain for every four who summit. It is more difficult and hazardous
11 to reach the peak of K2 from the Chinese side; thus, it is usually climbed from the
12 Pakistani side. Unlike Annapurna, the mountain with the highest fatality-to-summit
13 rate (191 summits and 61 fatalities), or the other eight thousands, K2 has never
14 been climbed during winter.

```

Listing 10.1 Abstract from Wikipedia page of K2 mountain.

### 10.1.1 Data Preparation

One of the improvements added in the new approach, when compared to T-Rex [33], is indeed in this step of Data Preparation, which swipes off from the text unusable characters, i.e. the ones comprised between parenthesis, not-English characters, apex, hyphens, etc. To accomplish Data Preparation, regex are employed. An example is showed in Listing 10.1.1. In this case, the pattern specifies to search for every round bracket or square bracket. The “|” character in the middle of the pattern counts as a boolean OR. The “^” character defines a negative set. This implies that, inside the pattern (for example, between a “[” and another “]”), all characters must be matched but not the one specified after the “ ^ ” (in our example, “]”, because that character is the one that ends the set).

```

1 Pattern(r'\([^)]*\)|\[[^\]]*\]', r'')

```

Listing 10.2 Regex pattern used for Data Preparation in SeAlion.

In Listing 10.1.1 we can see how, continuing with the K2 example, this step returns a cleaned, plain text, that is far more easy to process for the other forthcoming operations.

```

1 K2, also known as Mount Godwin Austen or Chhogori, is the second highest mountain
2 in the world, after Mount Everest, at 8,611 metres above sea level. K2 is located
3 on the China Pakistan border between Baltistan, in the Gilgit Baltistan region of
4 northern Pakistan, and the Taxkorgan Tajik Autonomous County of Xinjiang, China.
5 K2 is the highest point of the Karakoram range and the highest point in both
6 Pakistan and Xinjiang. K2 is known as the Savage Mountain due to the extreme
7 difficulty of ascent. K2 has the second highest fatality rate among the eight
8 thousands. With around 300 successful summits and 77 fatalities, about one
9 person dies on the mountain for every four who summit. K2 is more difficult and
10 hazardous to reach the peak of K2 from the Chinese side; thus, K2 is usually
11 climbed from the Pakistani side. Unlike Annapurna, the mountain with the highest
12 fatality to summit rate, or the other eight thousands, K2 has never been climbed
13 during winter.

```

Listing 10.3 Abstract of K2 after Data Preparation step.

Data Preparation includes coreference resolution. Coreference occurs when two linguistic expressions have the same referent. In our case, we see that the pronoun “it”, subject of the

second sentence, is converted in its referent “K2”. In SeAlion, coreference is implemented via API calls to external libraries, i.e. Spacy<sup>2</sup> and CoreNLP Coreference<sup>3</sup>. The former is a part of the spaCy parser, and uses a neural net scoring model. The latter is a part of the Stanford Core Natural Language Processing, a tool suite for NLP.

### 10.1.2 Entity Linking

After Data Preparation, the text is ready for Entity Linking. Entity Linking (EL) is also called Named Entity Disambiguation: this term focuses on the necessity to tackle the potential ambiguity among different possible instances [13]. Entity linking begins with Entity Recognition (ER), the task of identifying and marking entities in the text [115]. At this point, EL can disambiguate the entities, anchoring them to a KG identifier, such a DBpedia URI. To perform the linking task, SeAlion uses Wikifier’s<sup>4</sup> and DBpedia-Spotlight’s<sup>5</sup> APIs. These calls produce two sets of linked entities, one for each annotator. These sets will then be intersected and a new, abstract class which will be taken into consideration for the following steps. Once the annotation is complete, the tool’s interface (Figure 10.1) shows the results of the process by highlighting linked entities in text. Moreover, a list at the end of the page illustrates which annotator produced which linking, to allow a faster assessment from the user.

Entity marked with the black circle (“S”) are found by DBpedia-Spotlight. Entity paired with the fuchsia circle (“W”) are annotated by Wikifier. In this case, the former seems to have way more coverage.

### 10.1.3 Literal Extraction

Literal extraction is performed by SeAlion with the help of SUTime Date Linker<sup>6</sup>, but also regular expressions to match geo-coordinates, URLs, Email addresses, IP addresses, ISBNs.

As we see in Figure 10.2, in our case the tool extracts different numbers, along with a (mistaken) geocoordinate. In this case we reuse the set of regex that comes from MantisTable (Section 4.1.2).

---

<sup>2</sup>spacy.io

<sup>3</sup>stanfordnlp.github.io/CoreNLP/coref.html

<sup>4</sup>wikifier.org

<sup>5</sup>www.dbpedia-spotlight.org/demo/

<sup>6</sup>stanfordnlp.github.io/CoreNLP/sutime.html

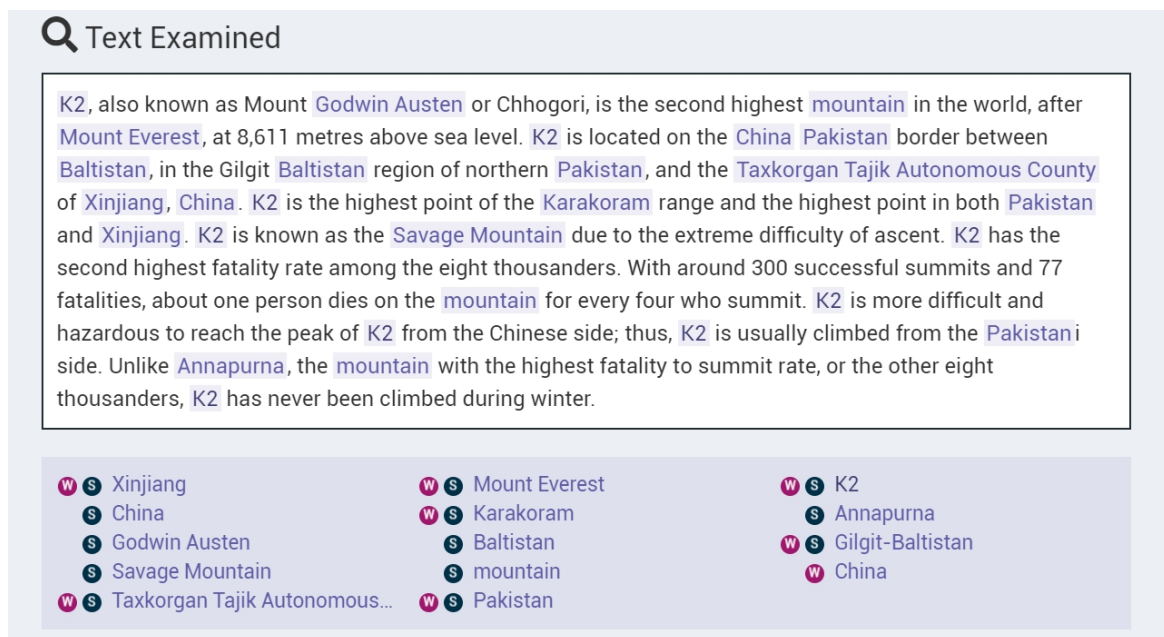


Figure 10.1 Results of Entity Linking in SeAlion.

### 10.1.4 Relation Extraction

After entities and literals are dealt with, SeAlions tries to perform predicate linking. The base approach of SeAlion for predicate linking uses ABSTAT [94]<sup>7</sup>, a distributed tool for calculating and exploring profiles of RDF data. ABSTAT takes a dataset and the related ontology (in OWL<sup>8</sup> format) as input, and produces a summary and statistics about the dataset. The summary is composed of patterns in the form of tuples  $\langle E, P, V \rangle$  which represent the presence of instances of  $E$  that are linked to instances of  $V$  via instances of  $P$ . Statistics includes frequencies for classes and predicates that are present in patterns. After entity linking, the classes of each entity get searched. After the class retrieval, for any couple of classes representing subject-object instances, an ABSTAT query is done. The result is a set of predicates between the given classes. The presence of each potential predicate is checked in text. At the end of the list of predicates, if no correspondence is found, synonyms of them are searched with the tool RelatedWords [98].

Let's now imagine that we want to extract relations and information from our abstract of K2 Wikipedia page, focusing on the first sentence only: "K2, also known as Mount Godwin-Austen or Chhogori, at 8,611 metres above sea level, is the second highest mountain in the world, after Mount Everest at 8,848 metres." SeAlion's current predicate aligner gives this output (Figure 10.3).

<sup>7</sup>backend.abstat.disco.unimib.it

<sup>8</sup>www.w3.org/OWL/



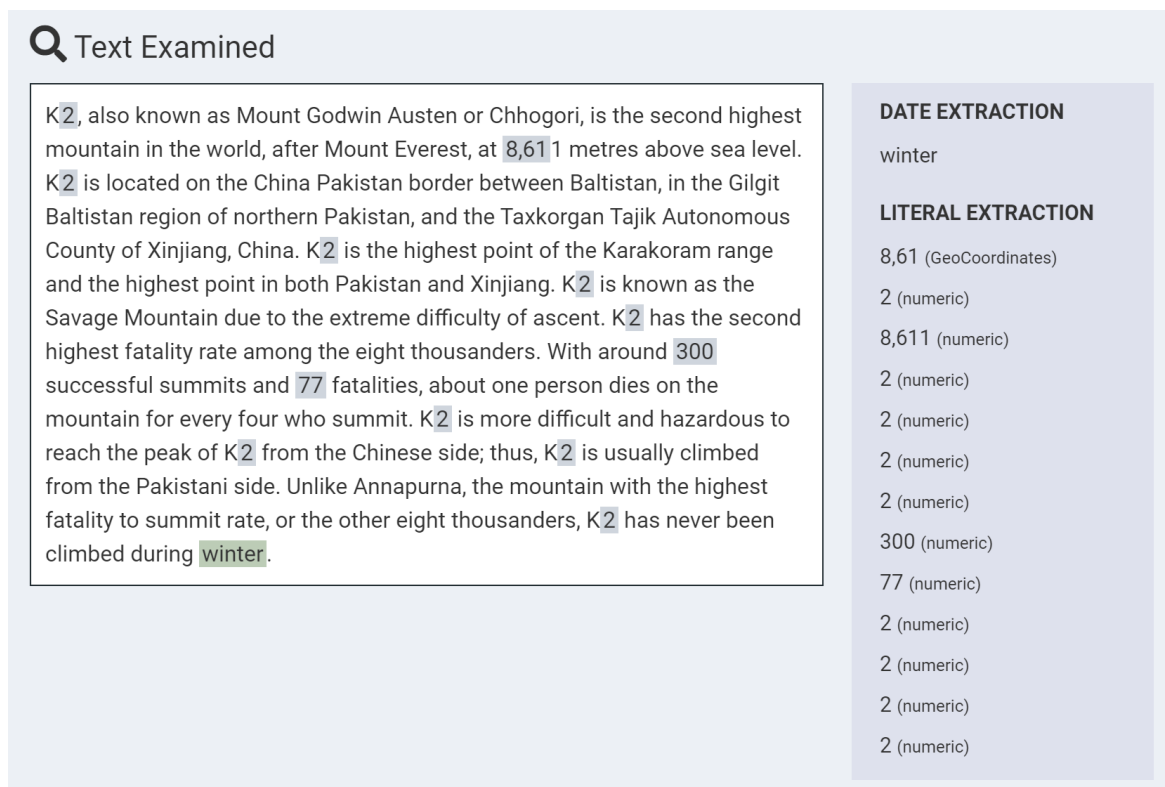


Figure 10.2 Results of Literal Extraction in SeAlion.

The sentence in output shows the abstraction of some entities, performed the ABSTAT query. We see that, in this case, SeAlion fails to abstract K2 and Chhogori, even though the former was correctly detected by entity linking. Suggested triples would therefore put in relation “Mount Godwin Austen” and “Mount Everest”, and “Mount Everest” and the literal 8,611.

**Issues with Relation Extraction.** The main problem is that this approach does not take into account the syntactic characteristics of the sentence. So, it struggles to correctly determine subjects of sentences and, likewise, the subjects of triples. A critical issue is that, without knowing the functional structure of the sentence, it is impossible to reliably distinguish which entities are actually put in relation, and which are not. SeAlion needs a new approach for the Predicate Extraction step, since the results for this phase are particularly lacking. We can note, for example, the results about these two sentences “The capital of Andorra is Andorra la Vella, and this is the highest capital city in Europe. The official language is Catalan, although Spanish, Portuguese, and French are also commonly spoken.”. In the first one, the predicate is successfully extracted, as ABSTAT considers the couple (Country, Populated Place) and correctly selects the relation capital. An example

Output text

K2, also known as Mount Godwin Austen or Chhogori, is the second highest mountain in the world, after Mount Everest, at 8,611 metres above sea level. K2 is located on the China Pakistan border between Baltistan, in the Gilgit Baltistan region of northern Pakistan, and the Taxkorgan Tajik Autonomous County of Xinjiang, China. K2 is the highest point of the Karakoram range and the highest point in both Pakistan and Xinjiang. K2 is known as the Savage Mountain due to the extreme difficulty of ascent. K2 has the second highest fatality rate among the eight thousands. With around 300 successful summits and 77 fatalities, about one person dies on the mountain for every four who summit. K2 is more difficult and hazardous to reach the peak of K2 from the Chinese side; thus, K2 is usually climbed from the Pakistani side. Unlike Annapurna, the mountain with the highest fatality to summit rate, or the other eight thousands, K2 has never been climbed during winter.

SENTENCE	PREDICATES	TRIPLES
K2, also known as dbo:Mountain or Chhogori, is the second highest mountain in the world, after dbo:Mountain at xsd:double metres above sea level.	<ul style="list-style-type: none"> <li>• mountain</li> <li>• k2</li> <li>• mount</li> <li>• everest</li> <li>• mount</li> <li>• known</li> </ul>	<ul style="list-style-type: none"> <li>• ( dbo:Mountain , dbo:mountainRange , dbo:Mountain )</li> <li>• ( dbo:Mountain , dbo:name , dbo:Mountain )</li> <li>• ( dbo:Mountain , dbo:prominence , xsd:double )</li> </ul>

Figure 10.3 An output from SeAlion’s current predicate aligner.

of wrong alignment, though, is given by the second sentence. The found triple for it is  $\langle \textit{Portuguese}, \textit{official language}, \textit{Spain} \rangle$ . This happens despite the fact that Spain is not an entity expressed in the sentence. With Spain (dbo:Country) and Portuguese (dbo:Language) marked in the sentence, ABSTAT hypothesises that their relation has to be that the latter is the official language of the former. We note two things in this example: the first is the incorrect denotation of Spanish as “Spain”. The second is that the correct object of the triple is actually not expressed in the sentence, but this system has no clue of it. From this sentence, the correct triples would be:  $\langle \textit{Catalan}, \textit{official language}, \textit{Andorra} \rangle$   $\langle \textit{Spanish}, \textit{spoken in}, \textit{Andorra} \rangle$   $\langle \textit{Portuguese}, \textit{spoken in}, \textit{Andorra} \rangle$   $\langle \textit{French}, \textit{spoken in}, \textit{Andorra} \rangle$

A human reader, having read the previous sentences of the text, would have no problem to infer that Andorra, even though not expressed in this particular sentence, is the (implied) focus of the discourse. This is due to the mental processes similar to the ones that allow us to resolve coreference. In the process of text comprehension, we form mental models that help us to identify the element of referential expressions. The factors involved are the cognitive salience of the element, semantic and syntactic constraints but also the so-called givenness hierarchy [41]. According to the givenness hierarchy, the more a referent is in focus, the less the referential expression will need to be specific. A more sophisticated system, even if it doesn’t understand that a fundamental entity is only implied in the text, at least figures out that something is missing. For example, in DBpedia the domain of official language is Populated Place, while the range is Language (the domain refers to the subject while the range refers to the object of a predicate). Given that, in the discussed sentence, no entity is a Populated Place, it should be possible to infer that the object of the triple is not assignable, unless we retrieve a Populated Place entity from some previous sentence.

**Improving Relation Extraction.** As in the state-of-the-art, good results are undoubtedly achieved with statistical, pattern-based approaches [32]. Even though they could successfully

align a satisfying percentage of relations, however, they have more difficulties when it comes to gathering other information in text. The predicate extraction tool could take various advantages also from a syntactic analysis as dependency parsing. Dependency parsing maps a sentence to its dependency tree, extracting the functional structure of the sentence. Dependency trees are directed trees: one of the nodes is called root, or head. The root has a certain number of child nodes, that can have children themselves. The result is a structure where the head node comes before all the others. Nodes are connected by functions, that specify the relation among them. We can see, for example, that the root node in the sentences above is represented by the verb. Dependency parsing, thus, can reveal which are the functional relation between words and constituents, and can help to understand the role of each part of the sentence.

In Figure 10.4 we see the dependency parsing of the sentence about K2, processed with Stanford CoreNLP<sup>9</sup>. We observe how K2 is marked as subject, depending on the “mountain” element via the function *nsubj*.

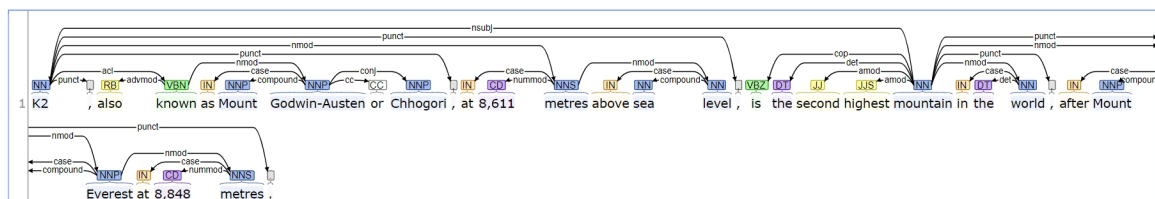


Figure 10.4 An example of how dependency parsing could help relation extraction.

K2 has a dependency that points towards the “known” verb via the function *acl*. This function indicates an adjectival clause, a clause that modifies a nominal. If we make the assumption that the verb are potential relation, we can search in the FrameNet<sup>10</sup> a corresponding entry. FrameNet is a linguistic tool provided via semantic frames, i.e. conceptual structure for the description of entities, events and relations. The FrameNet database also contains a huge number of lexical units, i.e. the pairing of a word with one of its meaning(s), with annotated examples that illustrate their usage. This resource is based on a theory of meaning called Frame Semantics [35]. According to this theory, the meanings of words can be really understood by only having all the essential information related to the concept expressed, as the event, the relation and the entities involved. FrameNet’s lexical units describe every different possible realisation for any entry, with the phrase structure and the associated semantic information. We can see how the lexical entry “known.a”, shown in Figure 10.5, contains the information on how different semantic arguments are syntactically realisable.

<sup>9</sup>corenlp.run

<sup>10</sup>framenet.icsi.berkeley.edu/fndrupal/

Frame Element	Number Annotated	Realization(s)
Entity	(29)	NP.Ext (28) NP.Head (1)
Name	(29)	AVP.Dep (1) DNL.-- (1) NP.Dep (1) PP[as].Dep (25) PP[as].Ext (1)
Type	(1)	AVP.Dep (1)

Figure 10.5 The Frame Elements and Their Syntactic Realisations for known.a in FrameNet.

In our case, we see that the Name argument for the predicate “known” is most commonly realised with a Prepositional Phrase that begins with “as”. This information can be cross-checked with the dependencies of “known” in the sentence: we see that “Mountain Godwin-Austen” depends on it, via a nmod function. Its case, “as”, confirms the fact that it is the Name element specified in FrameNet. Moreover, we can now leverage another advantage of dependency parsing: the “conj” function, that individuates conjunctions between elements. “Mountain Godwin-Austen”, indeed, is connected to “Chhogori” with an “or”. This can help us to extract not only one triple, but two:

*⟨K2, alternativeName, MountainGodwin – Austen⟩*

*⟨K2, alternativeName, Chhogori⟩*

The main difficulty, here, is the lacking of a proper mapping from FrameNet and DBpedia: we would need a system to lead back to FrameNet “known.a”, that is part of the frame “Being\_named”, to the predicate `dbo:alternativeName`.

Semantic information associated to FrameNet entries helps to characterise and contextualise the data contained in sentences, and to exploit potentially all of them. Moreover, the detection of incorrect alignment is facilitated, especially if we use ABSTAT. If the classes of the two entities, detected by ABSTAT, are radically different from the ones that the hypothesised predicate expects in the semantic description of the verb (e.g. the entities are concrete objects, but the predicate only accepts abstract concepts) we can infer that something has gone wrong.

A more linguistically deep analysis has enormous advantages. To face the complexity of human language, complex tools need to be employed. The contribute given by FrameNet-like knowledge base is essential. To tackle the problem of the lacking of a proper mapping to DBpedia’s predicates, an approach that follows the idea beyond PATTY [70] could be successful. We think that the Semantic Web needs a proper knowledge base for linguistic realizations, to be paired with DBpedia’s predicate dataset. This knowledge base would be a more DBpedia-friendly counterpart of tools like FrameNet. Each predicate should have its

syntactic and semantic realisations, so that when it is found in natural text, a system would know what to search.

Let's see a more concrete application of this approach. For example, in a sentence as: Julia, daughter of Julius Caesar, was born on 76 B.C.

We have to extract two triples:

$\langle \text{Julius\_Caesar}, \text{child}, \text{Julia\_}(\text{daughter\_of\_Julius\_Caesar}) \rangle$

$\langle \text{Julia\_}(\text{daughter\_of\_Julius\_Caesar}), \text{birthDate}, -76 \rangle$

If we dependency parse the sentence, we obtain what we see in 10.6.

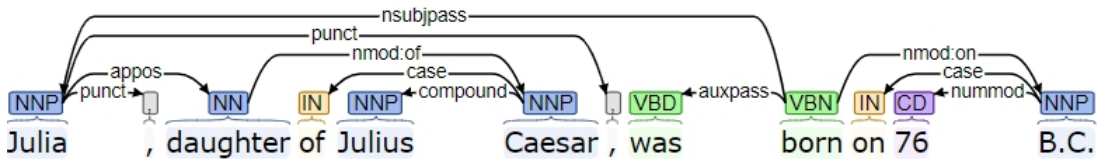


Figure 10.6 Dependency parsing for the sentence “Julia, daughter of Julius Caesar, was born on 76 B.C.”.

Let's imagine that we have a knowledge base where DBpedia predicates are represented with all their possible realisations. Among them, we will have these two (the first one for the predicate “child” and the second one for “birthDate”):

1	NP.Person.TripleObject	-(appos)->	[son/daughter]	-(nmod)->	PP[of].Person.TripleSubject
2	NP.Person.TripleSubject	<-(nsubj)-	VP.born	-(nmod)->	PP[on].Date.TripleObject

Note that the syntax is for illustrative purpose only. This simple example illustrates how the idea of such a knowledge base could work. The main difficulty would be in the realisation of such a huge project. All the data contained in existing resources, as FrameNet, VerbNet, WordNet, PropBank and many others, could be a good starting point. This would require, though, an appropriate mapping strategy. Moreover, we should elaborate proper method to syntactically parse the input sentence, assessing whether to use Constituency Parse (relying on a phrase structure parse), Dependency Parsing or a combination of both. Dependency parsing could be more efficient when it comes to extracting relations, and it seems to be more parsimonious [46]. As we see in Figure 10.5, FrameNet works with a phrase structure grammar, rather than a dependency grammar, and so it describes realisations by indicating which type of phrase could be used (i.e. prepositional phrase, nominal phrase, etc). To represent this information, this is indeed an efficient way, because it allows to swiftly represent constituents that are groups of words that make homogeneous units.



# Chapter 11

## SeAlion and GazelLex Tools

### 11.1 SeAlion Architecture and Interface

SeAlion, a tool developed with Python and the Django framework, has been created to manage and visualise each phase of the homonymous approach described in Section 10.1. Figure 11.1 shows the Architecture of this tool, which is organised in three layers: the *View Layer* that provides a graphic user interface to serve different types of tasks such as loading abstracts, executing all the step of the alignment process and exploring the results of each one; the *Controller Layer* that creates the abstractions between the View layer and the Model layer, and implements all the phases of the new pipeline; and the *Model Layer* that manages data access components to communicate with a DB connector.

SeAlion provides the following features: i) loading and storing ii) execution iii) exploration.

**SeAlion Loading and Storing.** Abstracts are imported and stored in a MongoDB database. A list of loaded abstracts is displayed on the main page. For each abstract is provided with its URI, the date of loading and a progress bar showing the percentage of the process completeness. Through the interface, it is possible to add and load new abstract providing the Wikipedia URL.

**SeAlion Execution.** Having selected an abstract, it is possible to manage the execution of the four phases described in Section 10.1.

**SeAlion Exploration.** It is possible to navigate all the executed steps by clicking on each phase and analyse the results in the visualisation mode. In *Data Preparation* a comparison is shown between the original abstract and the text after normalisation: all the removed or

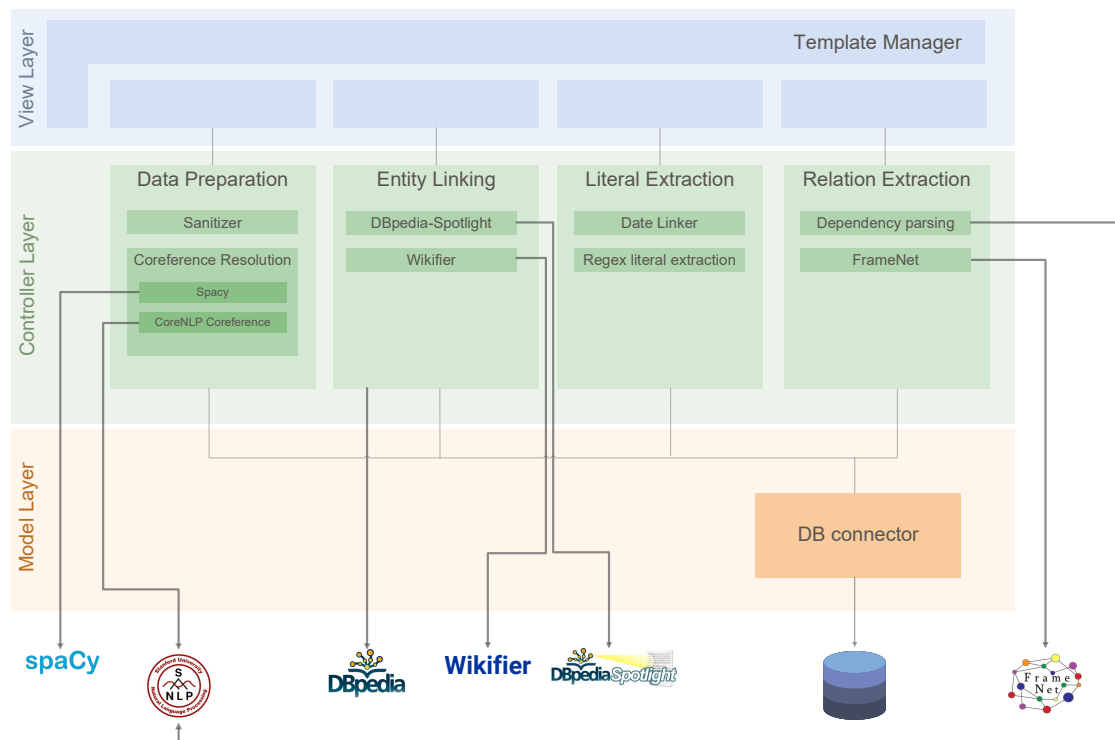


Figure 11.1 Architecture of SeAlion tool.

changed parts are highlighted; the coreference result is displayed at the bottom of the screen. In *Entity Linking*, the identified entities are highlighted in the text; each provides a summary and an image when hovered. Also, below the abstract, a list of the entities is displayed with reference to which annotator (Wikifier or DBpedia-spotlight) has found each of them. In *Literal Extraction* a list of found literal is shown. Finally, in *Relation Extraction* all the predicates found along with the final RDF triples are listed.

## 11.2 GazzelLex Tool

In order to test our proposal, we decided to apply the techniques previously described in a specific domain. In this section the GazelLex tool will be described (Gazette Lexicalization) [17], a prototype that covers several steps of Natural Language Generation, in order to create soccer articles automatically, using data from Knowledge Graphs. The project was partly commissioned by an Italian newspaper publisher. GazelLex can be considered as a tool for Automated Journalism; this new type of journalism uses algorithms to generate news under human supervision. GazelLex, through the use of deep learning techniques implements a NMT approach to generate articles (sentences) starting from data composed by RDF triples.



GazelLex is also able to generate videos containing the images and the prominent information of the article, and to generate audio using a speech synthesis module (Figure 11.2). To the best of our knowledge, our prototype is the first to provide an all-in-one integrated approach to NLG with RDF triples in the context of helping journalist in writing articles.

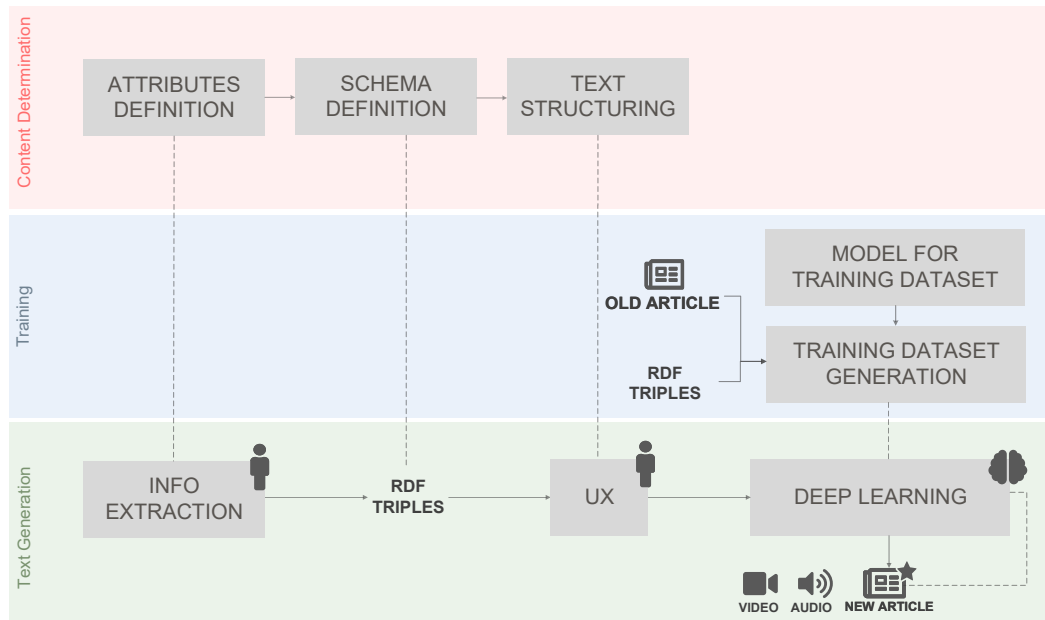


Figure 11.2 The workflow of our model.

Our approach is divided into five tasks, in order to address the five classic NLG sub-problems [37]: in the following, for each phase, implementation details will be provided.

**Content Determination.** To select the most relevant information, a handcrafted approach was chosen. To select the information to bring in the final output, we traced the most used data in soccer articles. One of the primary references was PASS, a personalised automated text system developed to write soccer articles [107]. We took the kind of information PASS used to fill its templates and enriched them with our data fields. So we have some entities of type “TEAM”, “FORMATION”, “COACH” and some predicates like “injuryAt”, “yellowCardAt”, and “violentFoulAt”<sup>1</sup>. The software used this data to create triples, that algorithms used to write the article.

**Text Structuring.** Being a domain specific process, we developed a handcrafted template, based on real articles. Aiming to get a similar output we imitated the journalist’s job in the division of text and about information contained in each part. We also considered the text

<sup>1</sup>A list of all the entities and the attributes are available here: [goo.gl/LonnQ5](http://goo.gl/LonnQ5)

structuring approach usually developed in this domain, that uses more general information and after that a chronological order [37]. In GazelLex, it is possible to find templates ( e.g. complete or short article) resulting from the process described above, but it is also possible to modify them or create new ones (Figure 11.3).

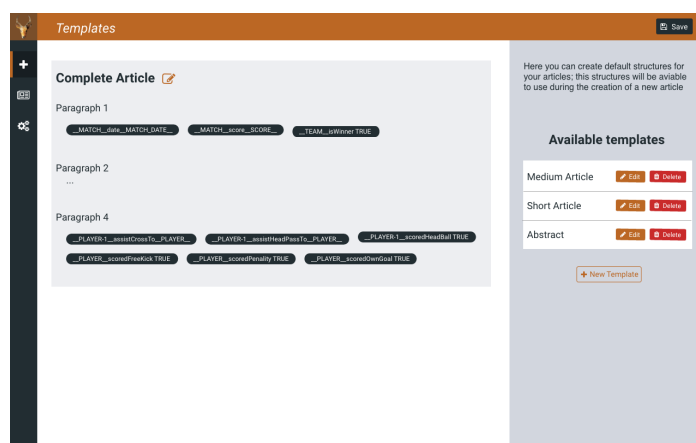


Figure 11.3 Page for creating and editing templates related to Text Structuring.

**Sentence aggregation** In soccer data, many events could be redundant when written in an article. If a player scores a hat trick in a match writing the same sentence about each goal would be unpleasant to read while grouping them in a single sentence could be more concise and coherent. This task “focused on domain- and application-specific rules” [37]. We aggregated the RDF triples defined in the preceding section to generate a group of triples that represents the content of our news article.

**Neural Lexicalization.** Like we said above, we considered lexicalization like a NMT process, converting RDF data into natural language. To achieve this aim, we used a specific kind of neural network: LSTM [45]. Their recent success in NLG field is related to many advantages they provide. Compared to the traditional neural network, LSTM do not have limitations in input and output length. Furthermore, input and output are not independent, that is a vital advantage in language generation. To predict a word in a sentence it is useful to know and consider the previous one, and the hidden states of the network keep the memory about what happened in previous timesteps. In this way, LSTM can combine the previous state, the memory collected and the input, allowing dependencies to be maintained in the long term. We experimented NMT using a now widely recognized tool for neural machine translation<sup>2</sup> [48]. Our neural architecture is based on a standard encoder-decoder structure

<sup>2</sup>opennmt.net

with 4 LSTM layers containing 200 hidden neurons on both the encoder and the decoder. Input tokenization is based on the space character (recall that our RDF triples' elements are separated by spaces).

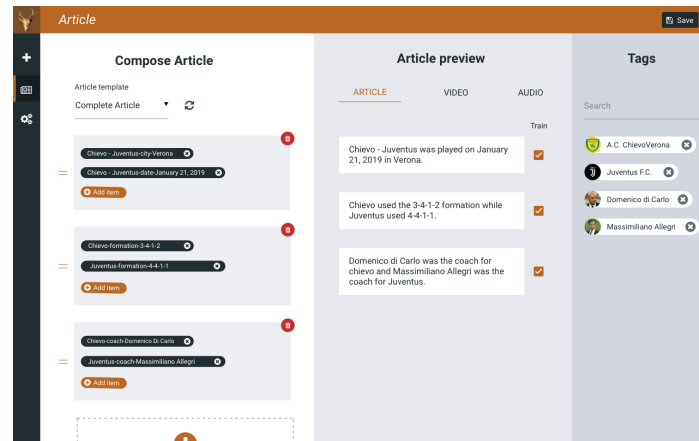


Figure 11.4 Page for the revision of the lexicalized triples.

**Reference expression generation.** We used different databases to avoid redundancy and give a fluent text to the reader. Some online resources help us to create a list of possible replacements for a team or players' name. Using DBpedia, we can find a nickname for an entity (Real Madrid players are also called *Blancos* or *Merengues*). Other resources we used are Wikidata list of soccer teams nicknames and Topend Sports database.

**Analysis.** In the following section, we shall show some insights into our tool and on how it works. We shall present a use case, a recent soccer match, for which the generation process and the resulting text will be shown. The initial dataset for the training was created manually and consists of 4387 pairs of triples and lexicalizations. We drew inspiration from the state-of-the-art to devise the architecture of our network [36, 104]. From our primary experiments the best performing model required two layers of bidirectional LSTM, but still, the model suffers from some limitations (outlined in the related Section).

**Use Case Exploration.** To show the valid output of GazzelLex, we took an example match and generated its lexicalization. We considered the football match played by Juventus F.C. and A.C Chievo on the 21st of January. Our application gathered data from an online provider and converted data in a triple format. A journalist can edit settings using a form (Figure 11.4): the journalist is in charge of deciding what is worth writing in the article and how it should appear to the end-user; we recall that we can also define templates for our articles (Figure 11.3). The final output of this process looks like the one that is shown in Figure 11.5.

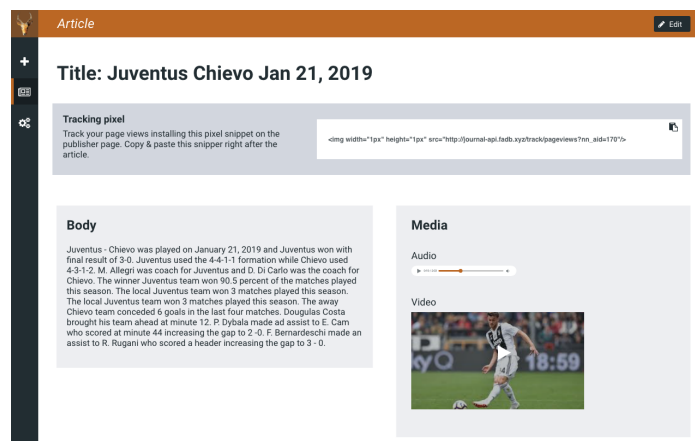


Figure 11.5 Lexicalization of triples from the Juventus-Chivevo football match.

GazelLex, in order to improve the quality of the sentences and to obtain results as close to the style of the journalist as possible (i.e. style transfer), cyclically re-executes the training phase using the sentences validated by the journalist. The following is an example of lexicalization of triples relative to the use case (Listing 11.1).

```

1 Paulo Dybala, assistTo, Emre
2 Emre Can, scoredAt, 44
3 Emre Can, scoredWithScore, 0 - 2
4
5 Paulo Dybala made an assist to
6 Emre Can who scored at minute 44
7 increasing the gap to 0 - 2.
```

Listing 11.1 Example of lexicalization.

## 11.3 Current Limitations and Lessons Learned

In this section we would like to outline the current limitations of our approach and also report a few lessons learned that might be useful for other researchers who are currently exploring this field. One key part of the development process comes from the definition or the selection of a good Knowledge Graph that can support the lexicalization; moreover, the definition of the new RDF predicates is a difficult process that must be done carefully in order to avoid errors in the next steps. Our NLG model is based on a deep learning architecture, and thus some of the generated sentences are not well-formed owing to the structure of the net itself. While this is a problem that has to be solved in our settings, we want to involve the user in revising the generated text: this allows us to have a model that is more flexible than standard pattern-based NLG, while the precision of the output can be controlled in a human-in-the-loop setting. Regarding the configuration of our model, we have replicated the state-of-the-art experiments (i.e. approaches explained in [36]) and we are currently experimenting those architectures on our new dataset. The results are yet to be quantitatively validated and they are preliminary, but they are promising. In particular, as reported by journalists, related to the GazelLex tool, the results are quite good. In the future, we are planning to explore various architectures and consider the use of word embeddings to solve some of our current issues.



## **Part IV**

# **Natural Language Generation from Tabular Data**





# Chapter 12

## Semantic Table Interpretation - Human Side

The advent of Big Data paved the way to the development of various technologies that allow the automatic (or semi-automatic) extraction of useful information from big quantity of data, either in the scientific or in the business domain. Let's suppose to have a table on the web, and that a user wants to know the topic without wasting much time analysing it. STI can help to generate a description for the table. The user would require, though, a software able to convert structured data into human-readable text.

The aim of this Chapter is to assess the various possible way with which an user can analyse a table, to finally delineate a process of selection for the data to lexicalise. A qualitative research resulted in the creation of a corpus of well-formed relational tables, with different domains and characteristics. These tables were then presented to a group of test user. With the aid of eye-tracking technology, ocular movements during the execution of some tasks were monitored.

The results of our analysis led us to the definition of a new tool, MantisTablex. Using semantic annotation obtained from MantisTable's approach (Section 4.1), the dataset created thanks to SeaLion (Section 10.1), the neural network tested for GazeLex (Section 11.2), it is possible to generated a natural language description of table data. We will also present the possibility to extend data to lexicalise with the use of AutomAPIC approach (Section 7.1).

With the aim to define a process of selection of data presented in a table, in preparation for the lexicalisation, we investigated how a user can interpret the table. To accomplish this, we adopted an Eye Tracker. It is a device that allows to monitor ocular movements during the execution of a task. The studies with Eye Tracker are based on the so-called Eye-Mind hypothesis [47], according to which there would be a direct match between gaze position and the point on which the attention is focused. Ocular movement, thus, can reveal in a dynamic fashion the point that receives user's attention. According different studies, indeed, it is possible to shift the attention without moving the gaze, but not vice versa. Fixations, in particular, can tell us how much time the brain dedicates to the analysis of a specific object. The registration of ocular movement, so, allows us to look into the cognitive processes of the subject, and to know what he finds interesting, important or not very clear.

## 12.1 Datasets

In this section we will describe the dataset used for the analysis. We delineated three categories: (i) Tables with concepts existing in DBpedia (ii) Tables with dynamic filters (iii) Tables extracted from T2Dv2 Gold Standard.

**Tables with concepts existing in DBpedia.** In the first group we put the tables where there was at least one NE-column that was possible to annotate with a class present in DBpedia.

**Crypto-currencies:** this table was found on an online post published on a blog in the field<sup>1</sup> and deals with a very specific topic, i.e. the implication of an attack to crypto-currencies. It is difficult to understand from who has not a pre-existing knowledge of the field.

---

<sup>1</sup>[www.cryptominando.it/2018/12/08/attacco-del-51-tabella-costi-stimati](http://www.cryptominando.it/2018/12/08/attacco-del-51-tabella-costi-stimati)

### La tabella dei costi

Name	Market Cap	Algorithm	Hash Rate	1h Attack Cost
Bitcoin	\$66.52 B	SHA-256	37,716 PH/s	\$275,613
Ethereum	\$10.51 B	Ethash	170 TH/s	\$75,390
Bitcoin Cash	\$2.07 B	SHA-256	1,080 PH/s	\$7,896
Litecoin	\$1.76 B	Scrypt	175 TH/s	\$18,203
Dash	\$662.77 M	X11	1 PH/s	\$3,851
Ethereum Classic	\$434.00 M	Ethash	9 TH/s	\$4,008
Zcash	\$358.05 M	Equihash	2 GH/s	\$15,226
Bytecoin	\$128.12 M	CryptoNight	523 MH/s	\$315
Siacoin	\$103.29 M	Sia	960 TH/s	\$0
Electroneum	\$60.00 M	CryptoNight	4 GH/s	\$2,364
Metaverse ETP	\$43.64 M	Ethash	505 GH/s	\$224
Bitcoin Private	\$40.49 M	Equihash	7 MH/s	\$47

Figure 12.1 “Crypto-currencies” tables with some related information.

**Fruit:** this table is available on a blog about tips<sup>2</sup> on how to improve one's health. The record number is  $50 < x < 100$ , so a fairly extensive informative content. The domain takes for granted a superficial knowledge of this topic by the average user.

### Frutta: calorie e valori nutrizionali

Tabella delle calorie e dei principali valori nutrizionali (carboidrati, proteine e grassi) della frutta. I valori si riferiscono a 100 grammi di parte edibile.

Alimento	Carboidrati (g)	Proteine (g)	Grassi (g)	Calorie (kcal)
Acerola	7,69	0,4	0,3	32
Acqua di cocco	3,71	0,72	0,2	19
Albicocche	11,12	1,4	0,39	48
Albicocche sciroppate	21,31	0,64	0,11	83
Albicocche sciroppate (non sgocciolate)	24,85	0,55	0,04	96
Amarena	12,18	1	0,3	50
Ananas	13,12	0,54	0,12	50
Ananas sciroppato	15,56	0,51	0,11	60
Anguria	7,55	0,61	0,15	30
Arancia	11,75	0,94	0,12	47
Avocado	8,53	2	14,66	160
Banana	22,84	1,09	0,33	89
Ciliegia	16,01	1,06	0,2	63
Clementina	12,02	0,85	0,15	47
Cocco essiccato	40,91	3,35	31,69	443
Crateva religiosa (Abiyuch)	17,6	1,5	0,1	69
Dattero (deglet noor)	75,03	2,45	0,39	282
Dattero (medjool)	74,97	1,81	0,15	277
Fichi secchi	63,87	3,3	0,93	249
Fico	19,18	0,75	0,3	74
Fico d'India	9,57	0,73	0,51	41
Fragola	7,68	0,67	0,3	32
Frutta candita	82,74	0,34	0,07	322
Giuggiola	20,23	1,2	0,2	79
Giuggiola essiccata	73,6	3,7	1,1	287
Guava (guayaba)	14,32	2,55	0,95	68
Kaki	18,59	0,58	0,19	70
Kiwi	14,66	1,14	0,52	61
Kiwi giallo (gold)	14,23	1,23	0,56	60
Kumquat	15,9	1,88	0,86	71
Lampone	11,94	1,2	0,65	52
Latte di cocco	5,54	2,29	23,84	230
Lime	10,54	0,7	0,2	30
Limone	9,32	1,1	0,3	29
Litchi	16,53	0,83	0,44	66
Macedonia	18,91	0,39	0,07	73
Macedonia di frutta tropicale	22,36	0,41	0,1	86

Mandarino	13,34	0,81	0,31	53
Mango	14,98	0,82	0,38	60
Mangostano sciroppato	17,91	0,41	0,58	73
Maracuja (frutto della passione)	23,38	2,2	0,7	97
Mela	12,76	0,27	0,13	48
Mela (con la buccia)	13,81	0,26	0,17	52
Mela cotogna	15,3	0,4	0,1	57
Mela cotta	13,64	0,26	0,36	53
Mela selvatica	19,95	0,4	0,3	76
Melagrana	18,7	1,67	1,17	83
Mele sciroppate	16,7	0,18	0,49	67
Melone	8,16	0,84	0,19	34
Mirtillo	14,49	0,74	0,33	57
Mirtillo palustre (cranberry)	12,2	0,39	0,13	46
Mora	9,61	1,39	0,49	43
Mora del getso	9,8	1,44	0,39	43
Nespola	12,14	0,43	0,2	47
Noce di cocco	15,23	3,33	33,49	354
Olive nere	6,26	0,84	10,68	115
Olive verdi	3,84	1,03	15,32	145
Papaya	10,82	0,47	0,26	43
Pera	15,46	0,38	0,12	58
Pera-mela	10,65	0,5	0,23	42
Pesca	9,54	0,91	0,25	39
Pescanocce (nettarina)	10,55	1,06	0,32	44
Pesche sciroppate	18,43	0,52	0,18	72
Pesche sciroppate (non sgocciolate)	19,94	0,45	0,1	74
Piatano	31,89	1,3	0,37	122
Pompelmo	8,08	0,63	0,1	32
Prugna	11,42	0,7	0,28	46
Prugne secche	63,88	2,18	0,38	240
Rabarbaro	4,54	0,9	0,2	21
Ribes nero	15,38	1,4	0,41	63
Ribes rosso	13,8	1,4	0,2	56
Sambuco	18,4	0,66	0,5	73
Tamarindo	62,5	2,8	0,6	239
Uva	18,1	0,72	0,16	69
Uva passa	79,52	3,39	0,46	302
Uva spina	10,18	0,88	0,58	44
Uvetta di Corinto	74,08	4,08	0,27	283

Figure 12.2 "Fruit" table that show fruit's nutritional values.

<sup>2</sup>[www.dietabit.it/alimenti/frutta/](http://www.dietabit.it/alimenti/frutta/)

**Food that contains water:** this table<sup>3</sup> has similar standards to the previous one, except that the amount of triples is larger (in the range of  $100 < x < 1500$ ), thus assuming a greater cognitive load.

Alimenti che contengono Acqua



Elenco degli alimenti che contengono Acqua, dal più ricco al meno ricco.

Il contenuto di Acqua si riferisce a 100 g di parte edibile di ogni alimento.

Alimento	Categoria	Acqua
Acqua minerale	Bevande analcoliche	99,98 g
Acqua del rubinetto	Bevande analcoliche	99,9 g
Camomilla	Tè, caffè e tisane	99,7 g
Tè nero	Tè, caffè e tisane	99,7 g
Tè nero decaffeinato	Tè, caffè e tisane	99,7 g
Cola light	Bevande analcoliche	99,54 g
Caffè	Tè, caffè e tisane	99,39 g
Caffè decaffeinato	Tè, caffè e tisane	99,3 g
Caffè solubile	Tè, caffè e tisane	98,93 g
Caffè d'orzo	Tè, caffè e tisane	98,37 g
Red Bull senza zucchero	Bevande analcoliche	98,35 g
Brodo di pollo	Zuppe e minestre	97,8 g
Caffè espresso	Tè, caffè e tisane	97,8 g
Caffè espresso decaffeinato	Tè, caffè e tisane	97,8 g
Brodo di manzo	Zuppe e minestre	97,55 g
Cetriolo	Verdura	96,73 g
Cetriolo	Verdura	96,73 g
Melone cinese da conserva	Verdura	96,1 g
Brodo di pesce	Zuppe e minestre	96 g
Germogli di bambù bolliti (con sale)	Verdura	95,92 g
Germogli di bambù bolliti (senza sale)	Verdura	95,92 g
Lattuga iceberg	Verdura	95,64 g
Lattuga rossa	Verdura	95,64 g
Lattuga	Verdura	95,63 g
Insalata	Fast food	95,51 g
Sedano	Verdura	95,43 g
Ravanello bianco	Verdura	95,37 g
Birra Budweiser light	Bevande alcoliche	95,3 g
Pomodori gialli	Verdura	95,28 g
Cetriolo (con la buccia)	Verdura	95,23 g
Fiori di zucca bolliti	Verdura	95,2 g

Luccio	Pesce e frutti di mare	78,92 g	Shake alla vaniglia (fast food)	Bevande analcoliche	69,65 g
Zenzero fresco	Aromi e spezie	78,89 g	Igname tropicale	Verdura	69,6 g
Piselli freschi	Verdura	78,86 g	Carne macinata (10% grasso)	Carne di manzo	69,5 g
Tequila sunrise	Bevande alcoliche	78,8 g	Cuori di palma	Verdura	69,5 g
Osmero americano	Pesce e frutti di mare	78,77 g	Uovo fritto	Uova	69,47 g
Insalata di patate	Fast food	78,73 g	Ketchup	Sughi e salse	69,15 g
Mais bianco in scatola	Verdura	78,73 g	Ketchup (ridotto contenuto di sodio)	Sughi e salse	69,15 g
Mais bianco in scatola (senza sale)	Verdura	78,73 g	Wasabi	Verdura	69,11 g
Mais in scatola	Verdura	78,73 g	Insaccato di tacchino affumicato	Salumi e insaccati	69,1 g
Mais in scatola (senza sale)	Verdura	78,73 g	Riso con pollo	Primi, secondi e contorni	69,09 g
Moringa (foglie)	Verdura	78,66 g	Germogli di soia	Verdura	69,05 g
Menta piperita, fresca	Aromi e spezie	78,65 g	Costolette	Carne di manzo	68,73 g
Patate russet	Verdura	78,58 g	Fianco	Carne di manzo	68,72 g
Calamari	Pesce e frutti di mare	78,55 g	Soia fresca bollita	Verdura	68,6 g
Tartaruga di mare	Pesce e frutti di mare	78,5 g	Soia fresca bollita (senza sale)	Verdura	68,6 g
Purè di patate	Verdura	78,48 g	Salmon	Pesce e frutti di mare	68,5 g
Milza di maiale	Carne di maiale	78,43 g	Fesa di tacchino arrosto	Pollame	68,4 g
Osmero americano	Pesce e frutti di mare	78,43 g	Ketchup (ridotto contenuto di sodio)	Sughi e salse	69,15 g
Insalata di patate	Fast food	78,73 g	Wasabi	Verdura	69,11 g
Mais bianco in scatola	Verdura	78,73 g	Insaccato di tacchino affumicato	Salumi e insaccati	69,1 g
Mais in scatola	Verdura	78,73 g	Riso con pollo	Primi, secondi e contorni	69,09 g
Mais in scatola (senza sale)	Verdura	78,73 g	Germogli di soia	Verdura	69,05 g
Moringa (foglie)	Verdura	78,66 g	Costolette	Carne di manzo	68,73 g
Menta piperita, fresca	Aromi e spezie	78,65 g	Soia fresca bollita	Verdura	68,6 g
Patate russet	Verdura	78,58 g	Soia fresca bollita (senza sale)	Verdura	68,6 g
Calamari	Pesce e frutti di mare	78,55 g	Salmon	Pesce e frutti di mare	68,5 g
Tartaruga di mare	Pesce e frutti di mare	78,5 g	Fesa di tacchino arrosto	Pollame	68,4 g
Purè di patate	Verdura	78,48 g	Shake alla vaniglia (fast food)	Bevande analcoliche	69,65 g
Milza di maiale	Carne di maiale	78,43 g	Igname tropicale	Verdura	69,6 g
Luccio	Pesce e frutti di mare	78,92 g	Carne macinata (10% grasso)	Carne di manzo	69,5 g
Zenzero fresco	Aromi e spezie	78,89 g	Cuori di palma	Verdura	69,5 g
Piselli freschi	Verdura	78,86 g	Uovo fritto	Uova	69,47 g
Tequila sunrise	Bevande alcoliche	78,8 g	Ketchup	Sughi e salse	69,15 g
Osmero americano	Pesce e frutti di mare	78,77 g	Ketchup (ridotto contenuto di sodio)	Sughi e salse	69,15 g
Insalata di patate	Fast food	78,73 g	Wasabi	Verdura	69,11 g
Mais bianco in scatola	Verdura	78,73 g	Insaccato di tacchino affumicato	Salumi e insaccati	69,1 g
Mais in scatola	Verdura	78,73 g	Riso con pollo	Primi, secondi e contorni	69,09 g
Mais bianco in scatola (senza sale)	Verdura	78,73 g	Germogli di soia	Verdura	69,05 g
			Costolette	Carne di manzo	68,73 g

Figure 12.3 Excerpt of the "Food that contains water" table, concerning data about the amount of water present in some groceries.

<sup>3</sup>www.dietabit.it/alimenti/acqua/

**Tables with dynamic filters.** In the second group we have included a set of tables on which it is possible to apply a filter.

**America's Top 50 Women In Tech:** this table shows a ranking from the American magazine Forbes<sup>4</sup> about women that work in technology fields. Despite containing only 50 rows, there are several filters.

**America's Top 50 Women In Tech**

**The List**

Filter list by: Sectors Categories Search by name

Name	Age	Sector	Category
Anne Aaron	41	Consumer Technology	Engineers
Tamar Bercovici	37	Enterprise Technology	Engineers
Perianne Boring	31	Digital Currency	Innovators
danah boyd	41	Digital Advocacy	Writers
Joy Buolamwini	39	Artificial Intelligence	Innovators
Ursula Burns	40	Consumer Technology	Megafa
Amy Chang	41	Enterprise Technology	Megafa
Sarah Clatterbuck	43	Enterprise Technology	Engineers
Cindy Cohn	55	Digital Advocacy	Writers
Morgan DeBaun	38	Media	Writers
Jennifer Doudna	54	Biotech	Innovators
Rana el Kaliouby	40	Artificial Intelligence	Founders
Li Fan	45	Consumer Technology	Engineers
Judy Faulkner	75	Technology	Megafa
Kathryn Finney	42	Digital Advocacy	Writers
Celeste Fraick	62	Cybersecurity	Engineers
Limor Fried	39	Consumer Technology	Founders
Jocelyn Goldfein	43	Venture Capital	Innovators
Diane Greene	63	Enterprise Technology	Megafa
Arquay Harris	40	Enterprise Technology	Engineers
Rachel Haurwitz	33	Biotech	Founders
Alyssa Henry	48	Enterprise Technology	Engineers

Ayanna Howard	47	Robotics	Innovators
Mary Lou Jepsen	53	Consumer Technology	Innovators
Maria Klawe	67	Education	Writers
Daphne Larose	30	Games	Engineers
Fei-Fei Li	42	Artificial Intelligence	Innovators
Holly Liu	43	Venture Capital	Megafa
Komal Mangtani	43	Enterprise Technology	Engineers
Laura Mather	47	Cybersecurity	Founders
bethanye McKinney Blount	46	Enterprise Technology	Founders
Erie Meyer	34	Digital Advocacy	Writers
Katie Moussouris	44	Cybersecurity	Innovators
Neha Narkhede	32	Enterprise Technology	Founders
Jennifer Pahlka	49	Digital Advocacy	Writers
Tal Rabin	36	Cybersecurity	Innovators
Carol Reiley	36	Artificial Intelligence	Founders
Tammarrin Rogers	49	Consumer Technology	Engineers
Ginni Rometty	64	Enterprise Technology	Megafa
Mona Sedky	53	Digital Advocacy	Writers
Gwynne Shotwell	53	Aerospace	Megafa
Kamakshi Sivaramakrishnan	43	Artificial Intelligence	Founders
Liss Su	30	Enterprise Technology	Megafa
Therese Tucker	57	Enterprise Technology	Megafa
Sherry Turkle	70	Digital Anthropology	Writers
Niniane Wang	39	Games	Innovators
Padmasree Warrior	58	Artificial Intelligence	Megafa
Tracy Young	34	Enterprise Technology	Founders
Michelle Zatlyn	39	Enterprise Technology	Founders

Figure 12.4 “America’s Top 50 Women In Tech” table, containing a ranking about women that work in technology fields.

<sup>4</sup>[www.forbes.com/top-tech-women-america/list/#tab:overall](http://www.forbes.com/top-tech-women-america/list/#tab:overall)

**Google Play Store apps 2018:** unlike the previous tables, this one is a dataset uploaded on the platform Kaggle<sup>5</sup>, obtained through a web scraping procedure. The table contains ten thousand apps found on Google Play Store<sup>6</sup>.

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated
1	ART_AND_DESIGN	4.1	159	19K	10,000+	Free	0	Everyone	Art & Design	January 7, 2018
2	ART_AND_DESIGN	3.9	967	14K	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018
3	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018
4	ART_AND_DESIGN	4.5	215044	23M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018
5	ART_AND_DESIGN	4.3	967	2.8K	100,000+	Free	0	Everyone	Art & Design;Creativity	June 28, 2018
6	ART_AND_DESIGN	4.4	167	5.8K	50,000+	Free	0	Everyone	Art & Design	March 28, 2018
7	ART_AND_DESIGN	3.8	178	10K	50,000+	Free	0	Everyone	Art & Design	April 28, 2018
8	ART_AND_DESIGN	4.1	36815	29K	1,000,000+	Free	0	Everyone	Art & Design	June 14, 2018
9	ART_AND_DESIGN	4.4	13791	33K	1,000,000+	Free	0	Everyone	Art & Design	September 20, 2018
10	ART_AND_DESIGN	4.7	121	3.1K	10,000+	Free	0	Everyone	Art & Design;Creativity	July 3, 2018
11	ART_AND_DESIGN	4.4	13080	20K	1,000,000+	Free	0	Everyone	Art & Design	October 27, 2018
12	ART_AND_DESIGN	4.4	8788	12K	1,000,000+	Free	0	Everyone	Art & Design	July 31, 2018
13	ART_AND_DESIGN	4.2	44829	20K	10,000,000+	Free	0	Teen	Art & Design	April 2, 2018
14	ART_AND_DESIGN	4.4	4326	21K	100,000+	Free	0	Everyone	Art & Design	June 28, 2018
15	ART_AND_DESIGN	4.4	1510	37K	100,000+	Free	0	Everyone	Art & Design	August 3, 2018
16	ART_AND_DESIGN	3.2	55	2.7K	5,000+	Free	0	Everyone	Art & Design	June 8, 2018

Figure 12.5 Table that contains a list of apps available on Google Play Store.

<sup>5</sup>www.kaggle.com

<sup>6</sup>play.google.com

**Tables extracted from the Gold Standard T2Dv2.** The third group is made of tables extracted from the Gold Standard T2Dv2, described in Section 5.2.1.

*Montagne - T2D Goldstandard set - (14311244\_0\_7604843865524657408.json)*

	A	B	C	D
1	<b>MOUNTAIN</b>	<b>HEIGHT IN METERS</b>	<b>RANGE</b>	<b>CONQUERED ON</b>
2	Mount Everest	8.848	Himalayas	May 29, 1953
3	K-2 (Godwin Austin)	8.611	Karakoram	July 31, 1954
4	Kanchenjunga	8.597	Himalayas	May 25, 1955
5	Lhotse	8.511	Himalayas	May 18, 1956
6	Makalu I	8.481	Himalayas	May 15, 1955
7	Dhaulagiri I	8.167	Himalayas	May 13, 1960
8	Manaslu	8.156	Himalayas	May 9, 1956
9	Cho Uyo	8.153	Himalayas	Oct 19, 1954
10	Nanga Parbat	8.124	Himalayas	July 3, 1953
11	Annapurna I	8.078	Himalayas	June 3, 1950
12	Gasherbrum I	8.068	Karakoram	July 5, 1958
13	Broad Peak I	8.047	Karakoram	June 9, 1957
14	Gasherbrum II	8.034	Karakoram	July 7, 1956
15	Shisha Pangma (Gasainthan)	8.013	Himalayas	May 2, 1964
16	Gasherbrum III	7.952	Karakoram	Aug 11, 1975
17	Annapurna II	7.937	Himalayas	May 17, 1960
18	Gasherbrum IV	7.923	Karakoram	Aug 6, 1958
19	Cyachug Kang	7.921	Himalayas	Apr 10, 1964
20	Kangbachen	7.902	Himalayas	May 26, 1974
21	Disteghil Sar I	7.884	Karakoram	June 9, 1960
22	Himal Chuli	7,864	Himalayas	May 24, 1960
23	Khinyang Chchish	7.852	Karakoram	Aug 26, 1971
24	Nuptse	7.841	Himalayas	Oct 1970
25	Gasherbrum East	7.821	Karakoram	July 5, 1960
26	Nanda Devi	7.816	Himalayas	Aug 29, 1936
27	Chomo Lonzo	7.815	Himalayas	Oct 30, 1954
28	Ngojumba Ri I	7.805	Himalayas	May 5, 1965
29	Rakaposhi	7.788	Karakoram	June 25, 1988
30	Batura Muztagh I	7.785	Karakoram	July 30, 1976
31	Zemu Gap Peak	7.780	Himalayas	Unclimbed
32	Kanjut Sar	7.760	Karakoram	July 19, 1939
33	Kamet	7.756	Himalayas	June 21, 1931

Figure 12.6 “Mountains” table that contains a list of mountains.

## 12.2 An Exploratory Research through Eye Tracking

The goal of this part is trying to understand how the table interpretation process works, focusing on the type of questions that the user tries to answer and following the information extraction process by studying the factors that affect comprehension. Some experimental



qualitative researches have been conducted using an Eye Tracker, starting from the following hypothesis:

- Option I: does the output text vary depending on the familiarity with the table's domain?
- Option II: does the output text vary depending on a specific goal before dealing with the table?

The main task was to generate a brief text that contained, according to its own view, the most relevant information shown within the table.

### 12.2.1 Option I: Experts (E) – Non Experts (NE) Comparison

To perform this analysis a total of 6 people had been involved: 3 categorised as “experts”(E), because they were distinguished by a proven interest about the topic shown within the table and 3 people that said they were not experts in the field (NE), but that they had heard about the topic at best. We used Tables 12.4, 12.3 and 12.6.

**“America’s Top 50 Women In Tech” table:** NE users focus more on images and on women’s names, trying to recognise someone familiar. The fields about area of specialisation and business branches are considered at a later time. A possible explanation of this phenomenon might stem from the *availability* heuristic that occurs during the decisional process about selecting information. This is a well-known cognitive shortcut that happens when, in uncertain situations, we tend to overestimate the freshest information in our memory [106]. Instead, the E users behaviour is the opposite, since they focus straightaway on those latter fields, stating that they’re performing some sort of ‘mental clustering’ to investigate which are the most prized areas. Furthermore, the expert users identify from the start how the filters work and evaluate them.

**“Food that contains water” table:** In this case both groups of users, after an initial phase of exploratory research, had struggled to identify which information were actually important among the amount of existing fields (>1500); we didn’t infer other significant considerations.

**“Mountains” table:** Similarly to the logic of Table 12.7, the NE user mostly focuses on the identification of mountains’ names that are perceived as familiar or, as subsequently highlighted, the ones that are considered curious or funny. The expert user regards as most interesting the field about the first climb of each mountain, occasionally jumping towards their names.

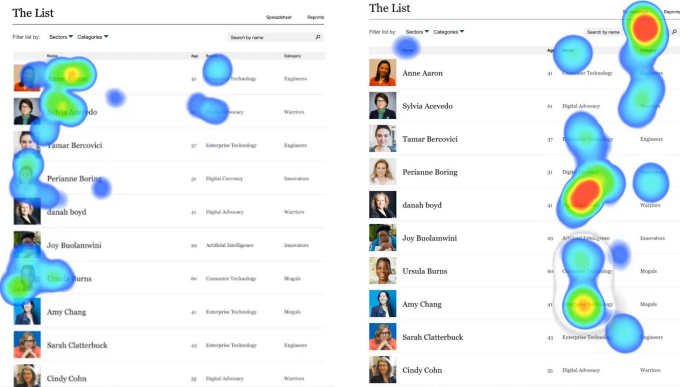


Figure 12.7 Eyetracking screen comparing NE(left)/E(right) about Table 12.4.

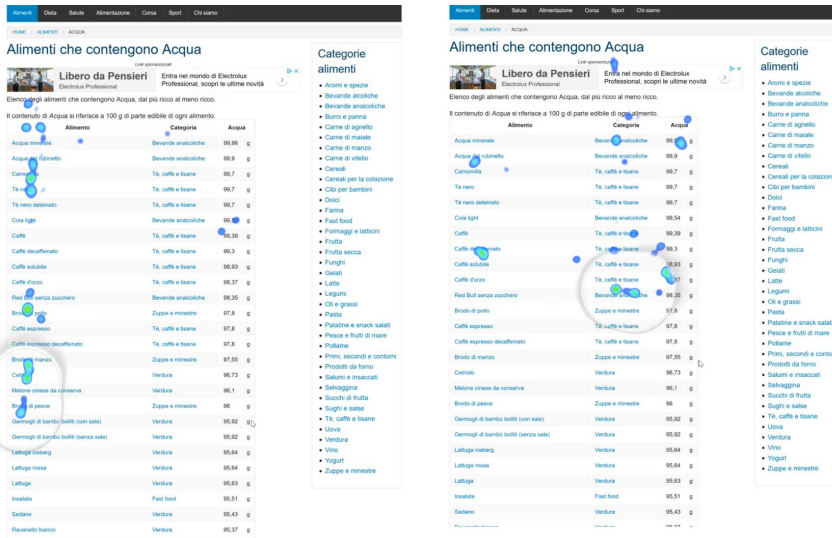


Figure 12.8 Eyetracking screen comparing NE(left) / E(right) about Table 12.3.



Figure 12.9 Eyetracking screen comparing NE(left) / E(right) about Table 12.6.

### 12.2.2 Option II: Task Oriented (TO) – Not Task Oriented (NTO) Comparison

6 people had been involved in this analysis too: 3 of them, called “ask - oriented” (TO) were notified in advance what was the purpose of the information contained inside the table; the other 3 people, called “not task oriented” (NTO), were told, like the users from the previous analysis, just to describe the table summarising all the relevant information.

To perform this analysis we selected tables 12.1, 12.2 e 12.5.

**“Cryptocurrencies” table:** NTO users declared not to know very well the domain of the table, and were able to identify the content of half of the table, either from previous knowledge or from deduction. Their attention was focused on the individuation of familiar terms, i.e. the first two records. TO users declared likewise not to know all the ontologies in the table. The fact that they had to imagine a possible future purchase led them to try to figure out what crypto-currency was more safe. Before starting to analyse the table, this users would have bought Bitcoin or Ethereum; after the reading of the table, anyway, they were not still so sure.

**Table “Fruit”:** In this case, the users’ behaviour was very similar to the one found in the E/NE analysis with table Water. In that instance, users found difficult to compare information and to remember the most important ones, due to the high number of records. TO users, after a distracting phase, performed a vertical scan of the potentially useful fields, making longer fixations and looking further on the horizontal axis, in particular on the selected fields. They then decided to generate an intermediate output, i.e. a list of selected terms with which they could produce, later, a summarising text. When they had been asked to explain this behaviour, users answered that they searched for their favourite fruits in the first place, to check then the amount of calories. This process implies a pre-existing knowledge on the ontology in question.

**Table “Google Playstore Apps 2018”:** The complexity and the high quality of data has been recognised, in this instance, by all users. Both TO and NTO users used dynamic filters on the tables. NTO users scanned the table searching some familiar name, as in the previous session, and then produced a list of apps to download. TO users focused the attention on the detection of possible relations between different categories, considering them a useful resource to assess in which field could be useful to invest.

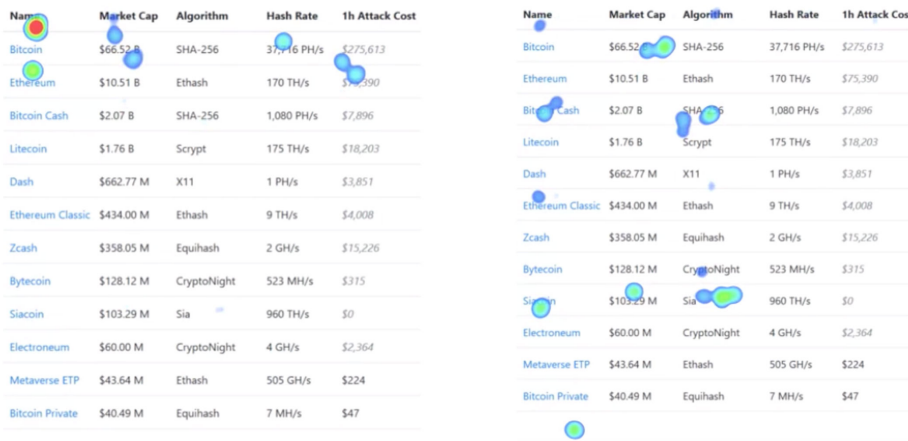


Figure 12.10 The comparison of Eyetracking Frame between NTO (left) and TO (right) on the Table 12.1.

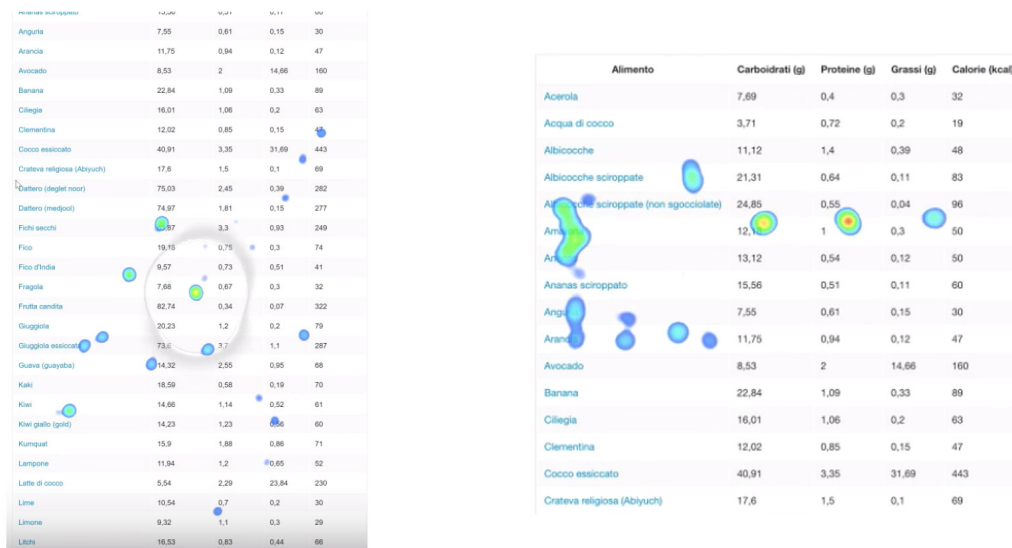


Figure 12.11 The comparison of Eyetracking Frame between NTO (left) and TO (right) on the Table 12.2.

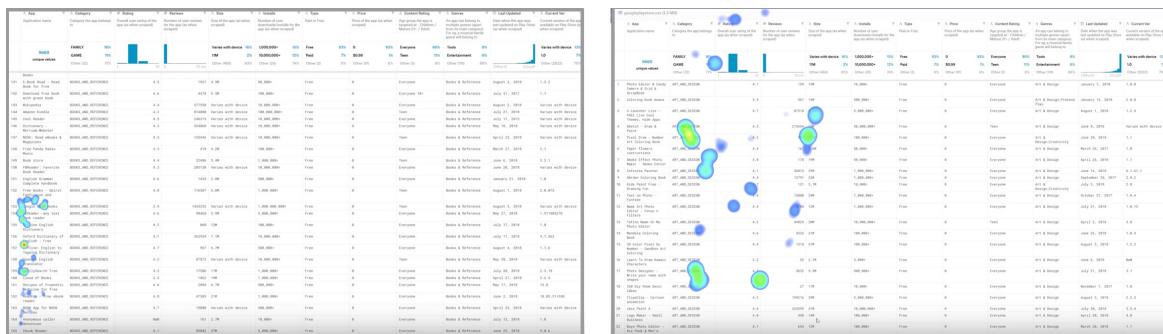


Figure 12.12 The comparison of Eyetracking Frame between NTO (left) and TO (right) on the 12.5.

## 12.3 Corpus Analysis

Below are the texts produced by users relating to the table of the mountains.

1. List of the highest mountains of the Himalayas and Karakoram. Only one peak in the Himalayas has not yet been climbed.
2. Himalayas and Karakorum are the mountain ranges with the highest mountains in the world. The highest mountains were climbed around the middle of the 900s and never in winter.
3. List of the 33 mountains among the highest in the world (7700mt +), of which only one ever reached (zemu gap peak). The others were climbed in the twentieth century for the first time.
4. In this table, there is a list of mountains with the name, height in meters, the mountain range to which they belong and the date on which they were climbed. Analyzing the range column, it is clear that all the mountains belong to the Himalayas or Karakoram mountain range.
5. This table shows the 30 highest peaks on Earth. They belong to the two mountain ranges of the Himalayas and the Karakoram. The mountains were climbed between the 30s and 80s; the most recent is the Rakaposhi of the Karakoram mountain range. It was climbed in 1988 and the Zemu Gap Peak, not yet climbed. The height of the mountains ranges between 7700 meters and 8800.
6. Most of the mountains are from the Himalayas.

Most users have included numbers in their description approximated to two digits (e.g. the years of climbing or the height of the mountains) that might suggest the difficulty of remembering a series of more than two numbers. When asked by the moderator of these tests what kind of descriptive text would have been useful to read before viewing the table (or to replace it), some users replied that they prefer a text that explains the structure of the table describing the categories, others a summary of the data inside. This division can also be observed from the type of texts produced. No one has opted for a text to list. All users have noticed the record where the information that differed from the rest was present, or the only mountain ever climbed.

The experiments shown in this Section are not to be considered significant, both for the sample size and for the methodology used. However, they allow us to set a basis for future studies and define a first set of specifications used for the implementation of a tool described in the next Section.



# Chapter 13

## MantisTablex Tool

### 13.1 Process

In this Section, we will describe the process that allows us to obtaining the representation in natural language of data in the tables. In order to make the individual steps clearer, we consider the table of mountains<sup>1</sup>, described in Section 4.1.

The input of the process is the data present in the table and the annotations obtained through the MantisTable approach (Figure 13.1).

dbo:Mountain	xsd:integer	dbo:MountainRange	xsd:year	xsd:string
Name	Height	Range	Conquered on	Coordinates
Mount Everest	8,848	Himalayas	1953	27.98785, 86.92502609999997
K2	8,611	Karakoram	1954	35.8799875,76.51510009999993
Manaslu	8,156	Himalayas	1956	28.5497107,84.55967680000003
Nanga Parbat	8,124	Himalayas	1953	35.2375038,74.58914460000005

Figure 13.1 MantisTablex input example.

With MantisTablex, we can identify two ways of selecting the data in the table: (i) single row and (ii) multiple rows. In the first case, the system lexicalises the single line. In the second, it provides a higher-level description of the elements present in the columns. Concerning this second mode, the output varies according to the level of background knowledge

<sup>1</sup>T2D table index: 14311244\_0\_7604843865524657408 - webdatacommons.org/webtables/goldstandardV2.html

of the user on the table domain. As seen in the previous Section, for the generation of a text dedicated to an expert user, the columns on the right are more taken in consideration. If the user is inexperienced, the columns on the left will be considered.

In the Listing 13.1 the triples extracted in relation to the selection by row are reported. With regards to the selection of multiple lines, in the Listing 13.2 it is possible to view the triples for the production of a text intended for an expert user, while in the Listing 13.3, triples for a not expert user.

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:dbo="http://dbpedia.org/ontology/"
5   xmlns:georss="http://www.georss.org/georss/">
6
7 <rdf:Description rdf:about="http://dbpedia.org/resource/Mount_Everest">
8   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
9   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
10     8848
11   </dbo:elevation >
12   <dbo:mountainRange rdf:resource="http://dbpedia.org/resource/Himalayas" />
13   <dbo:firstAscentYear rdf:datatype="http://www.w3.org/2001/XMLSchema#Year">
14     1953
15   </dbo:firstAscentYear>
16   <georss:point>27.988055555555555 86.92527777777778</georss:point>
17 </rdf:Description>
18 </rdf:RDF>

```

Listing 13.1 Selected triples in single-row selection mode.

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:dbo="http://dbpedia.org/ontology/">
5 <rdf:Description rdf:about="http://dbpedia.org/resource/Mount_Everest">
6   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
7   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
8     8848
9   </dbo:elevation >
10   <dbo:firstAscentYear rdf:datatype="http://www.w3.org/2001/XMLSchema#Year">
11     1953
12   </dbo:firstAscentYear>
13 </rdf:Description>
14 <rdf:Description rdf:about="http://dbpedia.org/resource/K2">
15   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
16   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
17     8611
18   </dbo:elevation >
19   <dbo:firstAscentYear rdf:datatype="http://www.w3.org/2001/XMLSchema#Year">
20     1954
21   </dbo:firstAscentYear>
22 </rdf:Description>
23 <rdf:Description rdf:about="http://dbpedia.org/resource/Manaslu">
24   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
25   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
26     8156
27   </dbo:elevation >
28   <dbo:firstAscentYear rdf:datatype="http://www.w3.org/2001/XMLSchema#Year">
29     1956
30   </dbo:firstAscentYear>
31 </rdf:Description>
32 <rdf:Description rdf:about="http://dbpedia.org/resource/Nanga_Parbat">
33   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
34   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
35     8124
36   </dbo:elevation >
37   <dbo:firstAscentYear rdf:datatype="http://www.w3.org/2001/XMLSchema#Year">
38     1953

```



```

39     </dbo:firstAscentYear>
40 </rdf:Description>
41 </rdf:RDF>

```

Listing 13.2 Selected triples for expert users in multi-row selection mode.

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:dbo="http://dbpedia.org/ontology/"
5 <rdf:Description rdf:about="http://dbpedia.org/resource/Mount_Everest">
6   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
7   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
8     8848
9   </dbo:elevation >
10  <dbo:mountainRange rdf:resource="http://dbpedia.org/resource/Himalayas" />
11 </rdf:Description>
12 <rdf:Description rdf:about="http://dbpedia.org/resource/K2">
13   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
14   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
15     8611
16   </dbo:elevation >
17   <dbo:mountainRange rdf:resource="http://dbpedia.org/resource/Karakoram" />
18 </rdf:Description>
19 <rdf:Description rdf:about="http://dbpedia.org/resource/Manaslu">
20   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
21   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
22     8156
23   </dbo:elevation >
24   <dbo:mountainRange rdf:resource="http://dbpedia.org/resource/Himalayas" />
25 </rdf:Description>
26 <rdf:Description rdf:about="http://dbpedia.org/resource/Nanga_Parbat">
27   <rdf:type rdf:resource="http://dbpedia.org/ontology/Mountain" />
28   <dbo:elevation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
29     8124
30   </dbo:elevation >
31   <dbo:mountainRange rdf:resource="http://dbpedia.org/resource/Himalayas" />
32 </rdf:Description>
33 </rdf:RDF>

```

Listing 13.3 Selected triples for non-expert users in multi-row selection mode.

Once the triples are collected, the user has the possibility to edit the groups of triples to lexicalise. Each group will result in the creation of a sentence. In the case of multi-line selection, the system will take care of performing two types of aggregations. If the object of the extracted triples contains the same value, after the lexicalization, the generated sentences will be appropriately aggregated. If the object contains a literal, we have defined several rules about the `RegexType` (Section 4.1.2). For numerical values, the data is first reworked according to the classic number aggregation functions (e.g. min, max, avg). This leads to the definition of a further triple which will then be used for the generation of a text containing the aggregate data. Please refer to the next section for an example. Currently, `MantisTablex` only supports aggregations on numbers. The development of a coordinate value management component is underway. As discussed in Section 10.1, the lexicalisation employs NMT. In particular, a neural network has been implemented using the framework `OpenNMT`<sup>2</sup>. Listing 13.4 shows the configuration of the network, according to the parameters defined in `OpenNMT`.

<sup>2</sup>[opennmt.net](http://opennmt.net)

```
1 -data data
2 -save_model model-mantistablex
3 -layers 2
4 -rnn_size 200
5 -dropout 0.2
6 -epoch 50
```

Listing 13.4 Configuration of the neural network.

From our primary experiments we noted that the best performing model required two layers of bidirectional LSTM [44]. LSTM is a variant of RNN; it adds a way to carry information across many time-steps. For this reason is a good layer to generate token sequences. A dropout was also included in the model. Dropout is one of the most effective and most commonly used regularisation techniques (to prevent overfitting) for neural networks [96]. Dropout, applied to a layer, consists of randomly dropping out (setting to zero) a number of output features of the layer during training. The dropout rate is the fraction of the features that are zeroed out. In this case, after a series of empirical tests, it was set at 0.2.

## 13.2 Architecture and Interface

MantisTablex is a web application developed with Python<sup>3</sup> and the Django framework<sup>4</sup>. A MongoDB<sup>5</sup> database acts as table and KG repository. The code is freely available through a Git repository<sup>6</sup>.

Figure 13.2 shows the modular architecture of MantisTablex, which is organised in three layers: the *View Layer* that provides a graphic user interface to serve different types of tasks such as storing and loading tables, storing and loading the semantic annotations, executing the lexicalization steps; the *Controller Layer* that creates the abstractions between the View layer and the Model layer, and implements all the lexicalization steps; and the *Model Layer* that manages mainly data access components to communicate with external data sources such as DB connector.

This tool provides the following features: i) loading and storing ii) execution iii) visualisation.

**MantisTablex Loading and Storing.** Tables are imported and stored in a MongoDB database. In MantisTablex, a list of loaded tables is displayed on the main page. For each table, a series of metadata is provided, such as name, date of loading, date of last

<sup>3</sup>[www.python.org](http://www.python.org)

<sup>4</sup>[www.djangoproject.com](http://www.djangoproject.com)

<sup>5</sup>[www.mongodb.com](http://www.mongodb.com)

<sup>6</sup>[bitbucket.org/disco\\_unimib/mantistablex-tool.py](https://bitbucket.org/disco_unimib/mantistablex-tool.py)

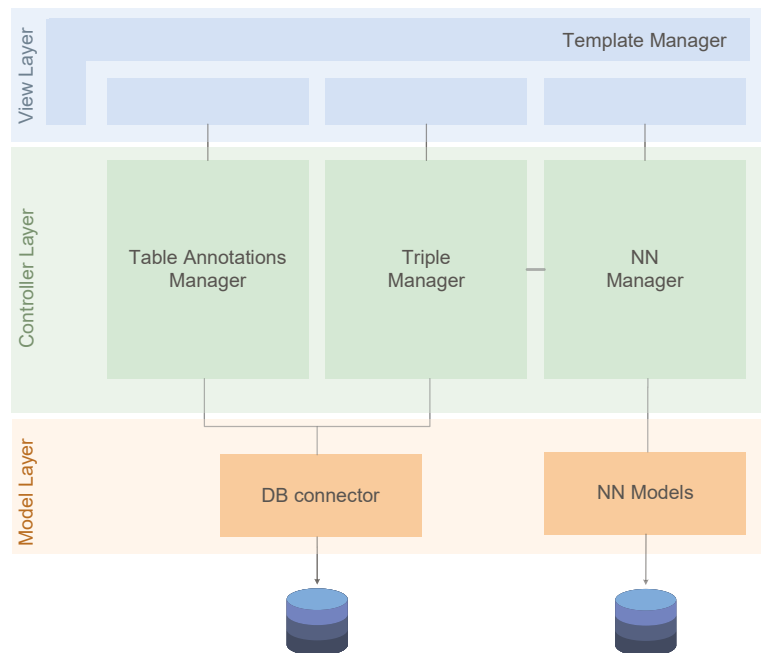


Figure 13.2 Architecture of MantisTablex tool.

modification. Through the interface it is possible to add and load new tables (in JSON format), load annotations for a table, and process a table.

**MantisTablex Execution.** After selecting a table, it is possible to manage the execution the lexicalization process.

**MantisTablex Visualization.** In the final step process all the triples the tool provides are listed: it is possible to reorder or remove them to change the final result. All the removed triples will not be taken into consideration in the lexicalization process but can be added again in a second moment. An example of the interface when a single row is selected is shown in Figure 13.3, while Figure 13.4 represents a multi-row selection scenario with low familiarity with the table domain.

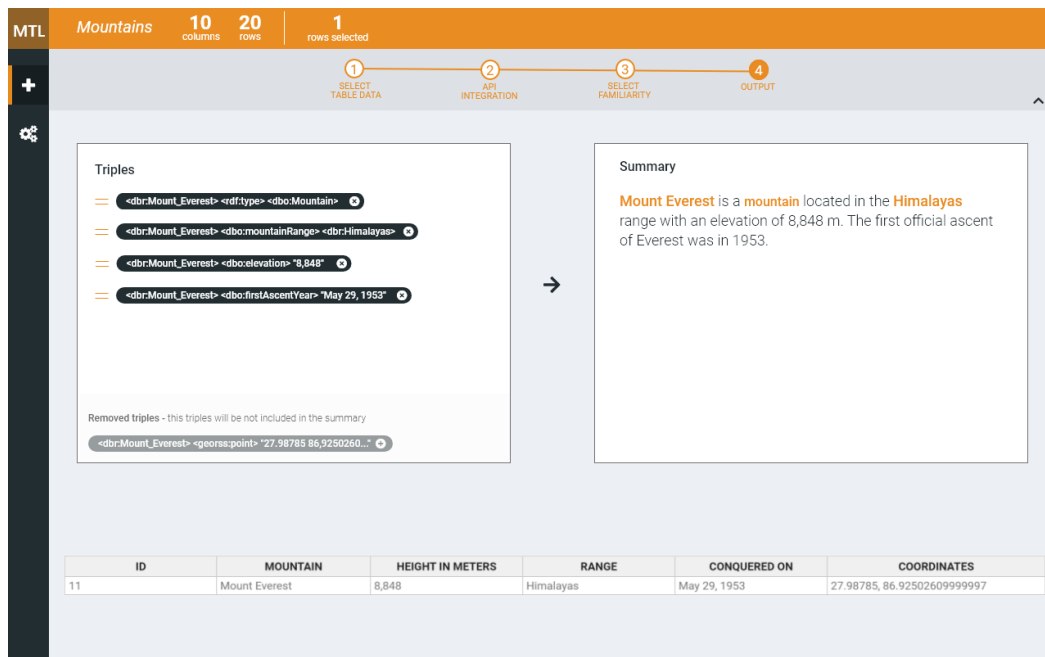


Figure 13.3 MantisTablex interface: triples selection and summary preview for single-row selection mode.

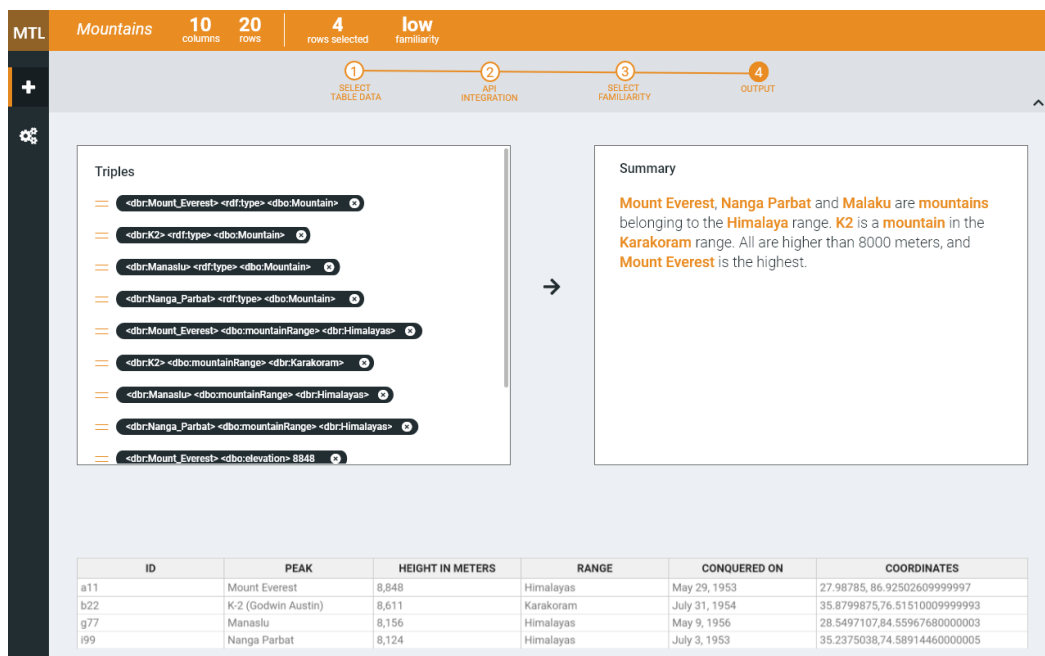


Figure 13.4 MantisTablex interface: triples selection and summary preview for multi-row selection row, low familiarity.

## **Part V**

# **Conclusion and Future Directions**



# Chapter 14

## Conclusions

One of primary objectives of this thesis was to define a method that makes tabular data more accessible, either by a human being or by a machine. To reach this aim, we decomposed it in sub-tasks: (i) define a new approach to semantically annotate tabular data; (ii) define a method to extend this data; and (iii) define a method to make semantic data more accessible and understandable by a human being.

For the first sub-task, we outlined a new STI pipeline, extending some techniques in the state-of-the-art, as for example the pipeline proposed by [123], the approach to identify column typology proposed in [87] or the approach for numeric dataset described in [73].

The resulting approach enhances the state of the art approaches, since it: (i) provides a comprehensive solution to support all annotations steps; (ii) provides an unsupervised method to annotate independent tables; (iii) generates context for disambiguation; and (iv) provides the MantisTable tool to support STI workflow and the STILTool tool to support the evaluation by providing validation indicators. Both tools are open and available.

The STI approach implemented in MantisTable has been deeply tested and validated –an activity which is still ongoing– to identify weaknesses. The first validation was conducted against the state-of-art Gold Standards, which are characterised by some problems, as the small number of tables, incomplete annotations (as they consider only Subject columns) or wrong annotations. A second, ongoing, validation led us to participate into the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching<sup>1</sup>. The final tests show that MantisTable outperforms all baselines: on the two most different datasets covering multiple domains and different table schemata, it significantly improves subject identification, concept and datatype annotation and finally property annotation for NE-column.

As a second achievement of this thesis, we propose an extension of the OpenAPI specification to support the semantic annotations of services descriptions and therefore the

---

<sup>1</sup>[www.cs.ox.ac.uk/isg/challenges/sem-tab/](http://www.cs.ox.ac.uk/isg/challenges/sem-tab/)

automatic composition of services. The goal is to help users without specific skills to manage semantics annotations, thus encouraging the delivery of semantically annotated descriptions. For this reason, two solutions have been proposed. For the annotation of input parameters, the use of Natural Language Processing (NLP) techniques has been presented, while for the annotation of output properties has been suggested the use of MantisTable. The approach has been implemented in AutomAPIc tool. The validation of the proposal through a subset of real APIs has underlined how the use of semantic annotations and the definition of a set of composition rules leads to an effective support to the composition of APIs, even if further development is necessary to improve both precision and recall.

The last achievement of this thesis is the definition of a lexicalisation process of tabular data, potentially extended with the data from APIs. Approaches that try to integrate deep networks and text generation are now common in literature, thanks to their excellent ability to generate texts. However, the use of these techniques involves a training phase through the use of a training dataset. Whilst in state-of-the-art it is possible to find different training datasets, our analysis showed that they were unfit for our objectives. For this reason, we developed SeaLion, a tool for triple-text alignment. To test the alignment approach, as well as the deep learning technique, we focused on a specific domain, i.e. automated journalism. With GazelLex, we showed an application related to the soccer domain, but the principles and the methodologies described are general, and they can be used in other fields (e.g. finance, weather reporting). We strongly believe that these tools can greatly help journalists in their daily activities (e.g. investigation, fact checking), leaving high effort, but low value tasks to computers. The current prototype should be considered a first step towards this automated process and the achieved results are surely promising.

## 14.1 Future Directions

The work presented in this thesis opens up several directions for future research.

The MantisTable approach, as shown in the evaluation Section, produces good results, but it still has some limitations. Our main goal is to improve the results using the analysis obtained with STILTool and the results of the Challenge. The STILTool allows the identification of tables that are critical and thus what needs to be improved. From the tables analysed for the Challenge, we already identify issues –failures or incorrect annotations– with tables containing entities belonging to different classes.

We are already making some revisions, such as the use of external resources to improve the content’s disambiguation within each cell during the Concept Annotation phase. During the analysis of the Gold Standard datasets, we noticed how some columns have non-annotable



elements, and detected the presence of tables with an incorrect structure (e.g. presence of multiple headers, different number of columns in the same table).

Another limit of MantisTable is about property annotation; the main issue is the difficulty to create a context to support disambiguation. In this case, also, we are developing techniques to resolve this shortcomings, achieving disambiguation via external sources, possibly employing other KGs.

We will further maintain and enhance the interface of MantisTable, for example by adding features to support the user when editing annotations. The involvement of users is fundamental for tuning up the final results. To support the annotation of huge tables we are developing a clustered version to deploy the tool in different nodes. Furthermore we are building a new Gold Standard that will provide high quality annotations especially for huge tables, so that we can also test the performance of the tool.

Relating to AutomAPIC, future work will go in that direction to consolidate the tool, along with fully integration with the Swagger interface<sup>2</sup>. In addition, a user-centric evaluation is planned in order to verify the ability of users to manage this new type of descriptions with semantic annotations. Finally, to enhance the automation of the entire process, we will study how to capture and model the user requirements.

Data are becoming more and more important for our lives and for our societies. Semantic Web promises to take a crucial role in this matter, but its exploitation is not still optimal. It is fundamental that humans have comfortable ways to interface with them. The more natural and rapid way for this is language, as it is the foundation of human communication. If this communication will be fully realised, future systems will destroy the last barrier still standing between humans and machines. The analysis presented in Section 10.1.4 describes the alignment task extensively and illustrates a possible new approach, respectful of the features, the structures and the functions that language evolved in thousands of years. It is however to be considered a starting point; as described above, it is necessary to investigate a way to create a knowledge base where DBpedia predicates are represented with all their possible realisations. This new resource could offer the opportunity to have a large number of good quality triple-text alignments.

As indicated a valid alternative to LSTM neural networks is represented by the new language representation models (e.g. BERT and GPT-2). However, introducing BERT to NMT is non-trivial, directly using BERT in NMT does not always yield promising results, especially for the huge datasets. As in many other NLP tasks, it is possible to use BERT as the initialisation of NMT encoder, or even directly replace the embedding layer of the encoder-decoder framework with the BERT embeddings. This does work in some scenarios

---

<sup>2</sup>[swagger.io/tools/swagger-ui/](https://swagger.io/tools/swagger-ui/)

with small datasets, but hardly gives encouraging results in high resource NMT benchmarks (e.g. WMT18<sup>3</sup> or WMT19<sup>4</sup>, , which always refer to large-size parallel datasets for training. Furthermore, that way to use a pre-trained model leads to remarkable improvements without fine-tuning, but gives few gains in the setting of fine-tuning in a resource-poor scenario [30]. While the gain diminishes when more labelled data become available. For these reasons, the use of Language Model (LM) within NMT scenarios is not immediate [119]. It is necessary to investigate new best practices on the use and integration of LM within machine translation techniques, in particular, to avoid the catastrophic forgetting problem (too much updating in training make the LMs forget its universal knowledge from pre-training) [39] in large datasets.

Regarding our lexicalization solution of the RDF triples, we have achieved good results in a specific domain (i.e. journalism). However, it is necessary to extend the evaluation of the models by considering more general domains and evaluation metrics present in the state-of-the-art (e.g. BLEU [61], ROUGE [60], METEOR [5]) .

In conclusion, in this thesis, the journey that allows data in a table to be “understood” both by a machine and by a human was described. The proposed solutions still have some weaknesses and be considered a starting point for future projects. However, the initial goals of the present work has been achieved as demonstrated by the extended validation phase that involved both gold standards and datasets of the “Tabular Data to Knowledge Graph Matching” challenge.

---

<sup>3</sup><http://www.statmt.org/wmt18/parallel-corpus-filtering.html>

<sup>4</sup><http://www.statmt.org/wmt19/parallel-corpus-filtering.html>

# References

- [1] Abele, A., McCrae, J. P., Buitelaar, P., Jentsch, A., and Cyganiak, R. (2017). Linking open data cloud diagram. URL <http://lod-cloud.net>.
- [2] Adida, B., Birbeck, M., McCarron, S., and Pemberton, S. (2008). Rdfa in xhtml: Syntax and processing. *Recommendation, W3C*, 7.
- [3] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [4] Azmeh, Z., Falleri, J.-R., Huchard, M., and Tibermacine, C. (2011). Automatic web service tagging using machine learning and wordnet synsets. In Filipe, J. and Cordeiro, J., editors, *Web Information Systems and Technologies*, pages 46–59, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [5] Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.
- [6] Barzilay, R. and Lapata, M. (2005). Collective content selection for concept-to-text generation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05*, pages 331–338, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [7] Barzilay, R. and Lee, L. (2004). Catching the drift: Probabilistic content models, with applications to generation and summarization. *arXiv preprint cs/0405039*.
- [8] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43.
- [9] Bianchi, F., Palmonari, M., Cremaschi, M., and Fersini, E. (2017). Actively learning to rank semantic associations for personalized contextual exploration of knowledge graphs. In Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., and Hartig, O., editors, *The Semantic Web*, pages 120–135, Cham. Springer International Publishing.
- [10] Bizer, C., Heath, T., and Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI Global.

- [11] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, pages 1247–1250, New York, NY, USA. ACM.
- [12] Cafarella, M. J., Halevy, A., and Madhavan, J. (2011). Structured data on the web. *Commun. ACM*, 54(2):72–79.
- [13] Chang, A., Spitzkovsky, V. I., Manning, C. D., and Agirre, E. (2016). A comparison of named-entity disambiguation and word sense disambiguation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 860–867, Portorož, Slovenia. European Language Resources Association (ELRA).
- [14] Chen, J., Jiménez-Ruiz, E., Horrocks, I., and Sutton, C. (2019). Colnet: Embedding the semantics of web tables for column type prediction. In *33rd AAAI Conference on Artificial Intelligence*, (AAAI 2019).
- [15] Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 1: Core language. *W3C recommendation*, 26:19.
- [16] Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1):51–89.
- [17] Cremaschi, M., Bianchi, F., Maurino, A., and Pierotti, A. P. (2019a). Supporting journalism by combining neural language generation and knowledge graphs. In *Sixth Italian Conference on Computational Linguistics 2019 (to appear)*.
- [18] Cremaschi, M. and De Paoli, F. (2017). Toward automatic semantic api descriptions to support services composition. In De Paoli, F., Schulte, S., and Broch Johnsen, E., editors, *Service-Oriented and Cloud Computing*, pages 159–167, Cham. Springer International Publishing.
- [19] Cremaschi, M. and De Paoli, F. (2018). A practical approach to services composition through light semantic descriptions. In Kritikos, K., Plebani, P., and de Paoli, F., editors, *Service-Oriented and Cloud Computing*, pages 130–145, Cham. Springer International Publishing.
- [20] Cremaschi, M., De Paoli, F., Rula, A., and Spahiu, B. (2019b). A fully automated approach to a complete semantic table interpretation. *Manuscript submitted for publication*.
- [21] Cremaschi, M., Rula, A., Siano, A., and De Paoli, F. (2019c). Mantistable: A tool for creating semantic annotations on tabular data. In Hitzler, P., Kirrane, S., Hartig, O., de Boer, V., Vidal, M.-E., Maleshkova, M., Schlobach, S., Hammar, K., Lasierra, N., Stadtmüller, S., Hose, K., and Verborgh, R., editors, *The Semantic Web: ESWC 2019 Satellite Events*, pages 18–23, Cham. Springer International Publishing.
- [22] Cremaschi, M., Rula, A., Siano, A., and De Paoli, F. (2019d). Semantic table interpretation using mantistable. In *The Fourteenth International Workshop on Ontology Matching 2019 (to appear)*.

- [23] Dale, R. (1989). Cooking up referring expressions. In *27th Annual Meeting of the association for Computational Linguistics*, pages 68–75.
- [24] Dale, R. and Reiter, E. (1995). Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263.
- [25] Deemter, K. V., Theune, M., and Krahmer, E. (2005). Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24.
- [26] Deng, D., Jiang, Y., Li, G., Li, J., and Yu, C. (2013). Scalable column concept determination for web tables using large knowledge bases. *Proc. VLDB Endow.*, 6(13):1606–1617.
- [27] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.
- [28] Duan, S., Kementsietsidis, A., Srinivas, K., and Udrea, O. (2011). Apples and oranges: A comparison of rdf benchmarks and real rdf datasets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD '11*, pages 145–156, New York, NY, USA. ACM.
- [29] Duboue, P. A. and McKeown, K. R. (2003). Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, EMNLP '03*, pages 121–128, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [30] Edunov, S., Baevski, A., and Auli, M. (2019). Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.
- [31] Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., and Christophides, V. (2017). Matching web tables with knowledge base entities: From entity lookups to entity embeddings. In *The Semantic Web – ISWC 2017*, pages 260–277, Cham. Springer International Publishing.
- [32] Ell, B. and Harth, A. (2014). A language-independent method for the extraction of RDF verbalization templates. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 26–34, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.
- [33] Elsahar, H., Vougiouklis, P., Remaci, A., Gravier, C., Hare, J., Laforest, F., and Simperl, E. (2018). T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.
- [34] Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer-Verlag, Berlin, Heidelberg.
- [35] Fillmore, C. J. et al. (2006). Frame semantics. *Cognitive linguistics: Basic readings*, 34:373–400.

- [36] Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017). The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133. Association for Computational Linguistics.
- [37] Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170.
- [38] Gomadam, K., Ranabahu, A., and Sheth, A. (2010). Sa-rest: semantic annotation of web resources. *W3C Member Submission*, 5:52.
- [39] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks.
- [40] Grice, H. P., Cole, P., Morgan, J., et al. (1975). Logic and conversation. *1975*, pages 41–58.
- [41] Gundel, J. K., Hedberg, N., and Zacharski, R. (1993). Cognitive status and the form of referring expressions in discourse. *Language*, pages 274–307.
- [42] Gupta, S., Szekely, P., Knoblock, C. A., Goel, A., Taheriyani, M., and Muslea, M. (2015). Karma: A system for mapping structured sources into the semantic web. In Simperl, E., Norton, B., Mladenic, D., Della Valle, E., Fundulaki, I., Passant, A., and Troncy, R., editors, *The Semantic Web: ESWC 2012 Satellite Events*, pages 430–434, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [43] Hadley, M. J. (2006). Web application description language (wadl). Technical report, Sun Microsystems, Inc., Mountain View, CA, USA.
- [44] Hochreiter, S. and Schmidhuber, J. (1997a). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- [45] Hochreiter, S. and Schmidhuber, J. (1997b). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- [46] Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.
- [47] Just, Marcel, A., Carpenter, and Patricia, A. (1980). A theory of reading: From eye fixations to comprehension. *Psychological review*, 87:329–54.
- [48] Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.
- [49] Knap, T. (2017). Towards odalic, a semantic table interpretation tool in the adequate project. In *Proceedings of the 5th International Workshop on Linked Data for Information Extraction co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria*, pages 26–37.

- [50] Knoblock, C. A., Szekely, P., Ambite, J. L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyani, M., and Mallick, P. (2012). *Semi-automatically Mapping Structured Sources into the Semantic Web*, pages 375–390. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [51] Kopecký, J., Vitvar, T., Fensel, D., and Gomadam, K. (2009). hrests & microwsmo. *STI International, Tech. Rep.*
- [52] Kruit, B., Boncz, P. A., and Urbani, J. (2019). Extracting novel facts from tables for knowledge graph completion (extended version). *CoRR*, abs/1907.00083.
- [53] Krummenacher, R., Norton, B., and Marte, A. (2010). Towards linked open services and processes. In Berre, A. J., Gómez-Pérez, A., Tutschku, K., and Fensel, D., editors, *Future Internet - FIS 2010*, pages 68–77, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [54] Lausen, H. and Farrell, J. (2007). Semantic annotations for wsdl and xml schema. *W3C recommendation, W3C*, 69.
- [55] Lehmborg, O., Ritze, D., Meusel, R., and Bizer, C. (2016). A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web, WWW '16 Companion*, pages 75–76, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.
- [56] Lerman, K., Plangprasopchok, and Knoblock, C. A. (2007). Semantic labeling of online information sources. *International Journal on Semantic Web and Information Systems*, 3(3):36–56. Copyright - Copyright IGI Global Jul-Sep 2007; Last updated - 2016-09-24.
- [57] Li, P., Comerio, M., Maurino, A., and De Paoli, F. (2009). An approach to non-functional property evaluation of web services. In *Proc. IEEE International Conference on Web Services, ICWS 2009*, pages 1004–1005.
- [58] Li, X., Strassel, S., Ji, H., Griffitt, K., and Ellis, J. (2012). Linguistic resources for entity linking evaluation: from monolingual to cross-lingual. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3098–3105, Istanbul, Turkey. European Languages Resources Association (ELRA).
- [59] Limaye, G., Sarawagi, S., and Chakrabarti, S. (2010). Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1-2):1338–1347.
- [60] Lin, C.-Y. and Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.
- [61] Lin, C.-Y. and Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain.

- [62] Lucky, M. N., Cremaschi, M., Lodigiani, B., Menolascina, A., and De Paoli, F. (2016). Enriching api descriptions by adding api profiles through semantic annotation. In Sheng, Q. Z., Stroulia, E., Tata, S., and Bhiri, S., editors, *Service-Oriented Computing*, pages 780–794, Cham. Springer International Publishing.
- [63] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.
- [64] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). Owl-s: Semantic markup for web services. *W3C member submission*, 22:2007–04.
- [65] Mazumdar, S. and Zhang, Z. (2016). A tool for creating and visualizing semantic annotations on relational tables. In *LD4IE@ISWC*. CEUR Workshop Proceedings.
- [66] Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11*, pages 1–8, New York, NY, USA. ACM.
- [67] Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing lstm language models.
- [68] Mitkov, R. (2014). *The Oxford Handbook of Computational Linguistics 2nd edition*. Oxford University Press.
- [69] Mulwad, V., Finin, T., and Joshi, A. (2013). Semantic message passing for generating linked data from tables. In Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J. X., Aroyo, L., Noy, N., Welty, C., and Janowicz, K., editors, *The Semantic Web – ISWC 2013*, pages 363–378, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [70] Nakashole, N., Weikum, G., and Suchanek, F. (2012). Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12*, pages 1135–1145, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [71] Narducci, F., Palmonari, M., and Semeraro, G. (2017). Cross-lingual link discovery with tr-esa. *Information Sciences*, 394-395:68 – 87.
- [72] Navigli, R. and Ponzetto, S. P. (2012). Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217 – 250.
- [73] Neumaier, S., Umbrich, J., Parreira, J. X., and Polleres, A. (2016). Multi-level semantic labelling of numerical values. In Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., and Gil, Y., editors, *The Semantic Web – ISWC 2016*, pages 428–445, Cham. Springer International Publishing.
- [74] Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.



- [75] Paulraj, D., Swamynathan, S., and Madhaiyan, M. (2012). Process model-based atomic service discovery and composition of composite semantic web services using web ontology language for services (owl-s). *Enterprise Information Systems*, 6(4):445–471.
- [76] Pautasso, C., Zimmermann, O., and Leymann, F. (2008). Restful web services vs. "big" web services: Making the right architectural decision. In *Proceedings of the 17th International Conference on World Wide Web, WWW '08*, pages 805–814, New York, NY, USA. ACM.
- [77] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- [78] Pham, M., Alse, S., Knoblock, C. A., and Szekely, P. (2016). *Semantic Labeling: A Domain-Independent Approach*, pages 446–462. Springer International Publishing, Cham.
- [79] Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y., and Sykes, C. (2009). Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7-8):789–816. AvImpFact=2.566 estim. in 2012.
- [80] Quercini, G. and Reynaud, C. (2013). Entity discovery and annotation in tables. In *Proceedings of the 16th International Conference on Extending Database Technology, EDBT '13*, pages 693–704, New York, NY, USA. ACM.
- [81] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.
- [82] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).
- [83] Ramnandan, S., Mittal, A., Knoblock, C. A., and Szekely, P. (2015). *Assigning Semantic Labels to Data Sources*, pages 403–417. Springer International Publishing, Cham.
- [84] Rao, J. and Su, X. (2005). A survey of automated web service composition methods. In Cardoso, J. and Sheth, A., editors, *Semantic Web Services and Web Process Composition*, pages 43–54, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [85] Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87.
- [86] Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.
- [87] Ritze, D. and Bizer, C. (2017). Matching web tables to dbpedia - a feature utility study. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, pages 210–221, Konstanz. OpenProceedings.

- [88] Ritze, D., Lehmborg, O., and Bizer, C. (2015). Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS '15*, pages 10:1–10:6, New York, NY, USA. ACM.
- [89] Roman, D., Kopecký, J., Vitvar, T., Domingue, J., and Fensel, D. (2015). Wsmo-lite and hrests: Lightweight semantic annotations for web services and restful apis. *Web Semantics: Science, Services and Agents on the World Wide Web*, 31:39 – 58.
- [90] Roman, D., Nikolov, N., Putlier, A., Sukhobok, D., Elvesæter, B., Berre, A., Ye, X., Dimitrov, M., Simov, A., Zarev, M., et al. (2018). Datagraft: One-stop-shop for open data management. *Semantic Web*, 9(4):393–411.
- [91] Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition: A decade's overview. *Information Sciences*, 280:218–238.
- [92] Shi, B. and Weninger, T. (2016). Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-Based Systems*, 104:123 – 133.
- [93] Sirin, E., Hendler, J., and Parsia, B. (2003). Semi-automatic composition of web services using semantic descriptions. In *1st Workshop on Web Services: Modeling, Architecture and Infrastructure*, pages 17–24.
- [94] Spahiu, B., Porrini, R., Palmonari, M., Rula, A., and Maurino, A. (2016). Abstat: ontology-driven linked data summaries with pattern minimalization. In *International Semantic Web Conference*, pages 381–395. Springer.
- [95] Speiser, S. and Harth, A. (2011). Integrating linked data and services with linked data services. In Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., and Pan, J., editors, *The Semantic Web: Research and Applications*, pages 170–184, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [96] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.
- [97] Stadtmüller, S., Speiser, S., Harth, A., and Studer, R. (2013). Data-fu: A language and an interpreter for interaction with read/write linked data. In *Proceedings of the 22nd International Conference on World Wide Web, WWW '13*, page 1225–1236, New York, NY, USA. Association for Computing Machinery.
- [98] Stanners, R. F., Neiser, J. J., Hernon, W. P., and Hall, R. (1979). Memory representation for morphologically related words. *Journal of Verbal Learning and Verbal Behavior*, 18(4):399 – 412.
- [99] Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, pages 697–706, New York, NY, USA. ACM.
- [100] Syed, Z., Finin, T., Mulwad, V., and Joshi, A. (2010). Exploiting a web of semantic data for interpreting tables. In *Proceedings of the Second Web Science Conference*, volume 5.

- [101] Taheriyani, M., Knoblock, C. A., Szekely, P., and Ambite, J. L. (2016). Learning the semantics of structured data sources. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37–38:152 – 169.
- [102] Takeoka, K., Oyamada, M., Nakadai, S., and Okadome, T. (2019). Meimei: An efficient probabilistic approach for semantically annotating tables. In *33rd AAAI Conference on Artificial Intelligence*, (AAAI 2019).
- [103] Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. (2015). Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.
- [104] Trisedya, B. D., Qi, J., Zhang, R., and Wang, W. (2018). Gtr- lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1627–1637.
- [105] Tsouropolis, R., Petychakis, M., Alvertis, I., Biliri, E., Lampathaki, F., and Askounis, D. (2015). Community-based api builder to manage apis and their connections with cloud-based services. In *CAiSE Forum*.
- [106] Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131.
- [107] van der Lee, C., Krahmer, E., and Wubben, S. (2017). Pass: A dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 95–104. Association for Computational Linguistics.
- [108] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.
- [109] Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 4(9):528–538.
- [110] Venigalla, H. and Di Eugenio, B. (2013). The content selection challenge entry from the university of illinois at chicago. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 210–211. Association for Computational Linguistics.
- [111] Verborgh, R., Harth, A., Maleshkova, M., Stadtmüller, S., Steiner, T., Taheriyani, M., and Van de Walle, R. (2014). Survey of semantic description of rest apis. In *REST: Advanced Research Topics and Practical Applications*, pages 69–89. Springer.
- [112] Verborgh, R., Mannens, E., and Van de Walle, R. (2015). Bottom-up web apis with self-descriptive responses. In *Proceedings of the First Karlsruhe Service Summit Workshop-Advances in Service Research*, page 143. KIT Scientific Publishing.

- [113] Verborgh, R., Steiner, T., Deursen, D. V., de Walle, R. V., and Gabarro, J. (2011). Efficient runtime service discovery and consumption with hyperlinked restdesc. In *The 7th International Conference on Next Generation Web Services Practices (NWeSP 2011)*, Salamanca, Spain.
- [114] Vrandečić, D. and Krötzsch, M. (2014). Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.
- [115] Walter, S. J. (2016). *Generation of Multilingual Ontology Lexica with M-ATOLL. A corpus-based approach for the induction of ontology lexica*. PhD thesis, Bielefeld: Universität Bielefeld.
- [116] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding.
- [117] Wang, J., Wang, H., Wang, Z., and Zhu, K. Q. (2012). Understanding tables on the web. In *Proceedings of the 31st International Conference on Conceptual Modeling, ER'12*, pages 141–155, Berlin, Heidelberg. Springer-Verlag.
- [118] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Ferguson, D. F. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- [119] Yang, J., Wang, M., Zhou, H., Zhao, C., Yu, Y., Zhang, W., and Li, L. (2019). Towards making the most of bert in neural machine translation.
- [120] Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. (2017). Breaking the softmax bottleneck: A high-rank rnn language model.
- [121] Zanibbi, R., Blostein, D., and Cordy, J. R. (2004). A survey of table recognition. *Document Analysis and Recognition*, 7(1):1–16.
- [122] Zhang, S. and Balog, K. (2018). Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, pages 1553–1562, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.
- [123] Zhang, Z. (2017). Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, 8(6):921–957.