



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MERGING HTML TABLES FOR EXTRACTING RELATIONS

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

JHOMARA TATIANA LUZURIAGA CARPIO

PROFESOR GUÍA:
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
GONZALO NAVARRO BADINO
JORGE PÉREZ ROJAS
RENZO ANGLES ROJAS

SANTIAGO DE CHILE
2019

RESUMEN

Con la aparición y evolución de la Web Semántica, las bases de conocimiento (e.g. DBpedia y Wikidata) han ido adquiriendo gran importancia como fuente de información para nuevos proyectos; existen al menos 1700 bases de conocimiento registradas en la nube de Linked Open Data, mientras se cuenta con cerca de 2 billones de sitios en la web; de aquí nace la necesidad de incrementar la información en formato estructurado. Cuanta más información con un alto grado de confianza se encuentre en dichas bases de conocimiento mayor será el beneficio para las aplicaciones que la utilizan.

En este trabajo proponemos extraer información de tablas HTML en formato estructurado (RDF) para alimentar estas bases de conocimiento, específicamente Wikidata. El lenguaje HTML permite crear documentos web en un formato semiestructurado que es interpretado por cierto software para mostrar los documentos al usuario; sin embargo, ya que el contenido carece de estructura semántica, el software está limitado para leer y explotar el contenido HTML. Extraer información de HTML y proporcionarle una estructura semántica es por lo tanto un tema de muchos trabajos de investigación.

En general, las tablas tienen una estructura relacional de donde se pueden extraer entidades, atributos y relaciones; sin embargo, en la web encontramos innumerables diseños de tablas, que plantean un desafío no trivial para extraer su información.

Nuestro trabajo de investigación se basa en la extracción de relaciones entre entidades que pueden ser identificadas en tablas HTML. Aunque una serie de trabajos de investigación ya han abordado este problema, los enfoques de extracción de relaciones existentes tienden a procesar cada tabla de forma individual. Nosotros proponemos una extensión de estos métodos basados en la agrupación de tablas con información similar, de modo que podamos aumentar el contexto de la información contenida en tablas pequeñas y complejas, que por sí mismas no proporcionan suficiente información para extraer relaciones con un buen nivel de confianza. Aplicamos el método propuesto para enriquecer Wikidata con triples extraídos de Wikipedia.

Los resultados de la tesis muestran que nuestro método para agrupar tablas obtiene mayor precisión al proporcionar características más robustas para clasificar relaciones candidatas como correctas o incorrectas, alcanzando 75% de precisión en la evaluación realizada sobre tablas individuales; mientras que al considerar las características propuestas por el método de Muñoz et al. [30] se obtuvo 71%. Además con 70% de precisión se pudo obtener más triples mediante nuestra propuesta de agrupar tablas. Consideramos estos resultados satisfactorios ya que a pesar de la gran cantidad de triples incorrectos que se pueden generar al agrupar las tablas pudimos obtener nuevos triples con similar nivel de precisión.

ABSTRACT

With the appearance and evolution of the Semantic Web, knowledge bases (e.g. DBpedia and Wikidata) have acquired great importance as a source of information for new projects; there are at least 1700 knowledge bases registered in the Linked Open Data cloud, while there are about 2 billions of websites, hence the need to increase information available in a structured format. The more information with a high degree of confidence available in these knowledge bases, the greater the benefit will be for the applications that use it. In this paper we propose to extract information from HTML tables in a structured format (RDF) to feed these knowledge bases, specifically Wikidata.

The HTML language allows to create web documents in a semi-structured format that is interpreted by certain software to show the documents to the user; however, since the content lacks semantic structure, the software is limited in terms of exploiting to read and exploit the HTML content. Extracting information from HTML and providing it with a semantic structure is therefore the subject of many research papers.

In general, the tables have a relational structure from which we can extract entities, attributes and relationships; nevertheless, on the web we find innumerable designs of tables, which pose a non-trivial challenge to extract their information.

Our research work is based on the extraction of relationships between entities that can be identified in HTML tables. Although a number of research papers have already addressed this problem, existing relationship extraction approaches tend to process each table individually. We propose an extension of these methods based on the grouping of tables with similar information, so that we can increase the context of the information contained in small and complex tables, which by themselves do not provide enough information to extract relationships with a good level of confidence. We apply the proposed method to enrich Wikidata with triples extracted from Wikipedia.

The results of the thesis show that our method for grouping tables obtains greater precision by providing more robust characteristics to classify candidate relationships as correct or incorrect, reaching 75% precision in the evaluation performed on individual tables; in comparison, considering the characteristics proposed by the method of Muñoz et al. [30], 71% was obtained. In addition, with 70% precision, more triples could be obtained through our proposal of grouping tables. We consider these results satisfactory since, despite the large number of incorrect triples that can be generated when grouping the tables, we were able to obtain new triples with a similar level of precision.

A Dios que siempre me acompaña a donde voy. A mis padres: José y Gloria; hermanas y hermanos: Diana, José, Ronny, Linda, Lesly y mis sobrinas: Vivianne, Danna, María Fernanda y el más reciente José Vinicio; que los adoro con mi alma, por su apoyo incondicional y paciencia hasta culminar este objetivo. A los amigos que en este camino conocí, por animarme a seguir en esos momentos de cansancio y a los amigos que permanecieron a pesar de mi ausencia.

Agradecimientos

A mi profesor guía Aidan, por su eterna paciencia, siempre dispuesto a enseñar, ayudar y responder mis inquietudes. A todos los profesores y compañeros que conocí y de quienes aprendí mucho en este recorrido, así como a Angélica Aguirre, Sandra Gaez y demás personal del DCC por su gran empatía y ayuda con todas las inquietudes que se me presentaron durante el programa. A Emir Muñoz y Henry Rosales por su colaboración en el desarrollo de este trabajo. Al grupo de "Forasteros del DCC", por los gratos momentos compartidos. A todos ellos por hacer de este aventurado camino un recorrido de continuo aprendizaje.

A la Secretaría de Ciencia y Tecnología de Ecuador, que me proporcionó la beca para acceder al programa y al Instituto Milenio de Fundamentos de los Datos de Chile por permitirme participar en sus proyectos y eventos de investigación, y poder fortalecer mis conocimientos.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Motivating Example	6
1.3	Objective	9
1.4	Hypothesis	9
1.5	Results	11
1.6	Structure of this work	11
2	Background	12
2.1	Semantic Web standards	12
2.1.1	Resource Description Framework (RDF)	12
2.1.2	RDF Schema (RDFS)	14
2.1.3	Web Ontology Language (OWL)	16
2.1.4	SPARQL	16
2.1.5	Web Knowledge Bases	17
2.1.6	Wikidata Knowledge Base	18
2.1.7	Linked Data	21
2.2	Information Extraction	22
2.2.1	Information Extraction from Text	22
2.2.2	Information Extraction from Web Tables	24
2.3	Machine Learning methods	26
2.3.1	Clustering methods	26
2.3.2	Classification methods	28
2.3.3	Evaluation metrics	30
2.3.4	Text Processing	32
3	Related Work	34
3.1	Table detection	34
3.2	Table interpretation	37
3.2.1	Parsing and Normalization	38
3.2.2	Entity detection	39
3.2.3	Attribute detection	40
3.2.4	Relation extraction	42
3.3	Clustering and merging tables	43

3.3.1	Clustering tables	43
3.3.2	Merging tables	45
3.4	Summary	47
4	Proposal	48
4.1	Table extraction and normalization	49
4.2	Knowledge base querying	51
4.3	Grouping tables	52
4.4	Relation Extraction	53
4.5	Classification and validation	54
5	Table Corpus	55
5.1	Tables extraction	55
5.2	Tables classification	56
5.3	Tables normalization	56
5.4	Table Interpretation	59
5.4.1	Header identification	60
5.4.2	Text processing	60
5.4.3	Data types	61
5.5	Article’s table entity	63
6	Knowledge Base Querying	65
6.1	Wikidata Access	65
6.2	Entity extraction	66
6.3	Triple extraction	67
6.4	Statistics for features modelling	68
7	Grouping tables	71
7.1	Clustering overview	71
7.2	Grouping tables proposal	74
7.3	Evaluation of table groups	75
8	Relation extraction	77
8.1	Candidate triples from individual tables	77
8.2	Candidate triples by merging tables	78
8.3	Evaluation of candidate triples	80
8.4	Dividing table groups	83
9	Triple classification	87
9.1	Features description	87
9.1.1	Adding new features	89
9.2	Classification algorithms	91
9.3	Dataset description	91
9.4	Classification models	92
9.4.1	Model A	93
9.4.2	Model B	95

9.4.3	Model C	96
9.4.4	Model D	97
9.5	Model Selection	98
9.6	Evaluation	98
	Conclusions	99
	Bibliography	103
	Appendix A Information Extracted from Wikidata	108
	Appendix B Inter-rate agreement	112
	Appendix C Model parameters description	114
	Appendix D Classification Models	117
D.1	Model A	117
D.1.1	Features selection	117
D.1.2	Hyper-parameters tuning	123
D.1.3	Model selection	125
D.2	Model B	126
D.2.1	Features selection	126
D.2.2	Hyper-parameters tuning	129
D.2.3	Model selection	130
D.3	Model C	131
D.3.1	Features selection	131
D.3.2	Hyper-parameters tuning	135
D.3.3	Model Selection	136
D.4	Model D	137
D.4.1	Features selection	137
D.4.2	Hyper-parameter Tuning	140
D.4.3	Model selection	141
D.5	Evaluation	142
D.5.1	Correct triples extracted	142

List of Tables

3.1	Example of table interpretation per Gentile’s approach	44
4.1	Example of table 4.4b normalized	50
4.2	Example of merging two tables with same headers (Colors pink and green identify the rows that come from both tables from example in Figure 4.4 .	53
5.1	Tables classification	56
5.2	Results after table normalization	58
5.3	Useful tables	58
5.4	Top 20 stemmed header names in tables	61
5.5	Example of a normalized table with article entity	63
6.1	Tables with hyperlinks	66
7.1	Cluster visualizations of sample set of 10000 tables	73
8.1	Example of candidate triples extracted based on Figure 8.3	79
8.2	Example of triple classification	81
8.3	Initial labeling of sample triples from I and $G - I$	82
8.4	Annotation agreement	85
9.2	New proposed features	90
9.3	Validation and test set	92
9.4	Results A1: Initial validation, A2: Features selection, A3:Balancing training set, A4: Hyper-parameters tuning	94
9.5	Results B1: Initial validation, B2: Features selection, B3:Balancing training set, B4: Hyper-parameters tuning	95
9.6	Results C1: Initial validation, C2: Features selection, C3:Balancing training set, C4: Hyper-parameters tuning	96
9.7	Results D1: Initial validation, D2: Features selection, D3:Balancing training set, D4: Hyper-parameters tuning	97
9.8	Results of classifying triples extracted from individual I tables (Model B and C) and triples extracted by merging tables (Model D)	99

A.1	Top 20 predicates from existing triples	110
A.2	Predicates with fewer than 10 triples	111
B.1	Inter-rate agreement for triples from I	112
B.2	Inter-rate agreement for triples from F-I	113
B.3	Inter-rate agreement for triples from G-I	113
D.1	Features with high correlation for Model A	119
D.2	Examples of triples with maximum similarity between column names and predicate	121
D.3	Top AUC values with Random Forest parameters Model A.1	123
D.4	Top AUC values with Bagging Decision Tree parameters Model A.1	123
D.5	Top AUC values with XGBoost parameters Model A.1	124
D.6	Features with high correlation for Model B	127
D.7	Top AUC values with Random Forest parameters Model B.1	129
D.8	Top AUC values with Bagging Decision Tree parameters Model B.1	129
D.9	Top AUC values with XGBoost parameters Model B.1	129
D.10	Features with high correlation for Model C	132
D.11	Top AUC values with Random Forest parameters Model C.1	135
D.12	Top AUC values with Bagging Decision Tree parameters Model C.1	135
D.13	Top AUC values with XGBoost parameters Model C.1	135
D.14	Top AUC values with Random Forest parameters Model D.1	140
D.15	Top AUC values with Bagging Decision Tree parameters Model D.1	140
D.16	Top AUC values with XGBoost parameters Model D.1	140
D.17	Examples of triples classified as correct to feed Wikidata	142

List of Figures

1.1	Relational table	3
1.2	Vertical table (Infobox)	4
1.3	Matrix table	4
1.4	Route map for metro bus, designed using a table structure	4
1.5	Table with combined cells and multiple headers	5
1.6	Table with no main entity inside	5
1.7	Table with ambiguous header	5
1.8	Extracting relations from tables according to Muñoz et al. [29]	7
1.9	Example of incorrect candidate triples	7
1.10	Example of small tables with no available candidate triples	8
2.1	RDF Graph for listing 2.1	13
2.4	Resource linked to Wikidata and DBpedia	21
2.5	Abstract Information Extraction Architecture [12]	23
2.6	HTML table design	24
2.7	Example of a dendrogram in hierarchical clustering	27
3.1	Listings: vertical and horizontal tables	35
3.2	Attribute-value table	35
3.3	Table Matrix	36
3.4	Enumeration table	36
3.5	Table used for navigational content	36
3.6	Table used as organization chart	37
3.7	Nested Table	38
3.8	Split Table	38
3.9	Multivalued Table	38
3.10	Table with colspan and rowspan	38
3.11	Normalized table from figure 3.10	39
3.12	Entity detection	40
3.13	Example of attribute detection	41
3.14	Relation extraction from tables	42
3.15	Example of horizontal merging	45
3.16	Example of vertical merging	46
3.17	Example of full merging	46

4.1	Proposal: Table relations extraction	48
4.2	Complex table design	49
4.3	Table with different type of data	49
4.4	Two example HTML tables from Wikipedia	50
4.5	Example entity extraction from HTML tables	51
4.6	Mapping entities from a normalized table	52
5.1	HTML files extraction	55
5.2	Table normalization model	56
5.3	Table with inner tables	57
5.4	Table with inner tables used as an organizer chart	57
5.5	Table structure recognition	58
5.6	Table with empty header	60
5.7	Text processing over table headers	61
5.8	Table with numeric type of columns	62
5.9	Table with repeated name headers	62
5.10	Distribution of column data types	63
6.1	Example of inverse relations extracted	67
6.2	Example of predicate multiplicity	68
6.3	Example of object entity not in range of predicate	69
6.4	Range and domain predicate extraction	69
7.1	Cluster visualization of sample test using T-SNE algorithm (the color represents the normalized distance from each point to the closest points)	72
7.2	Mean of jaccard distances sample set	72
7.3	Distribution of tables with same headers	74
7.4	Example tables from group 1	75
7.5	Evaluation of pair of tables from top 10 groups	76
8.1	Example of candidate triples extraction from individual tables	77
8.2	Example candidate triples extraction by merging tables	78
8.3	Example of existing triples from two tables indicating the source rows	79
8.4	Number of tables by group (a) including article's entity relations and (b) with no article's entity relations	79
8.5	Application used for triples annotation	81
8.6	Example table from which triples in table 8.2 where extracted	81
8.7	Table with multiple entity cells	82
8.8	Triples generated from tables (I) with multiple entity cells	83
8.9	Triples generated by merging tables ($G - I$) with multiple entity cells	83
8.10	Tables with conflicting relations	84
8.11	Number of tables by group, conflicts analysis	85
9.1	Cross-Validation Model A.4 (including results for 100 triples from held-out test set for each fold)	94

9.2	Cross-Validation Model B.4 (including results for 100 triples from held-out test set for each fold)	95
9.3	Cross-Validation Model C.4 (including results for 100 triples from held-out test set for each fold)	96
9.4	Cross-Validation Model D.4 (including results for 100 triples from held-out test set for each fold)	97
9.5	Precision vs number of validated triples	100
9.6	Precision vs total triples classified as correct (estimated from labeled sample)	100
A.1	Word cloud of article classes from which tables were extracted	108
A.2	Number of different classes by column in tables	109
D.1	Correlation Matrix (Model A)	117
D.2	Correlation between features and class (Model A)	118
D.3	Features 32 - 34	120
D.4	Features 32 - 36	120
D.5	Features 34 - 36	120
D.6	String similarity between predicate and subject column name	120
D.7	String similarity between predicate and object column name	120
D.8	Gini score of features for Model A	122
D.9	Precision-Recall curves Model A.4	125
D.10	ROC-curves Model A.4	125
D.11	Learning curves Model A.4 (Using F1-score)	125
D.12	Correlation between features and class (Model B, (+) indicates new feature)	126
D.13	Gini score of features for Model B	128
D.14	Precision-Recall curves Model B.4	130
D.15	ROC-curves Model B.4	130
D.16	Learning curves Model B.4 (Using F1-score)	130
D.17	Correlation between features and class (Model C, (+) indicates new feature)	131
D.18	Gini score of features for Model C	133
D.19	Ratio of rows where predicate holds in table (32) and group (62)	133
D.20	Precision-Recall curves Model C.4	136
D.21	ROC-curves Model C.4	136
D.22	Learning curves Model C.4 (Using F1-score)	136
D.23	Correlation Matrix with new proposed features	137
D.24	Correlation between features and class (Model D, (+) indicates new feature)	138
D.25	Features importance for model D	139
D.26	Precision-Recall curves Model D.4	141
D.27	ROC-curves Model D.4	141
D.28	Learning curves Model D.4 (Using F1-score)	141

Chapter 1

Introduction

1.1 Motivation

The information available on the Web has grown rapidly as access to technology increases; with this growth there is then the need to have large data warehouses and tools to enable the fast retrieval of relevant information. Such tools have been developed over the years in the area of Information Retrieval (IR), which with the emergence of the World Wide Web (WWW) in the late 80's, has overseen the development of methods and techniques used by Web Search Engines to operate with large volumes of data.

To search a phrase in a set of documents it is necessary to index the content of all documents and perform efficient lookups with the query phrase. For making the system usable over millions of documents, it is necessary to identify those that are most relevant. The relevance of a document in IR is defined using measures such as TF-IDF (Term Frequency - Inverse Document Frequency), based on the word frequencies in a collection of documents. In this way, search engines based on this measure return relevant documents in a short time; however they do not know if documents really contain the information that the user needs.

Although search engines have improved in recent years, they are only capable of retrieving documents not answers as results. For example search engines are typically unable to integrate information from multiple sources when responding to a user request. The Semantic Web (SW) was proposed to facilitate and improve the automatic process of search and retrieval, bestowing on the Web a semantic structure with the use of technologies, formats and standards such as RDF (Resource Description Framework), OWL (Web Ontology Language) and RDFS (RDF Schema), making it possible for software to retrieve, integrate and process information from multiple sources before showing it to users.

The Semantic Web encourages the description and linkage of resources on the Web using structured formats like RDF, which represents content as graphs and where each resource is identified with a URI (Uniform Resource Identifier), that can be referenced from across the Web. With this structure some projects like Freebase [32], YAGO [42], DBpedia [5] and Wikidata [45] have emerged as large popular knowledge bases.

With the expansion of this technology organizations such as: the UK Data Government ¹, the US Data Government ², the National Library of Spain ³, etc., have developed websites based on the Semantic Web scheme, with the philosophy of Open Data. Such datasets have been joined by many others from different domains such as Music Brainz ⁴, Geo Linked Data ⁵, UnitProt ⁶, building the "Linked Open Data" cloud.

The ratio of content with semantic structure available on the Web has increased, however there is still a significant gap when compared with the content of the Web without structure; with the aim of reducing this gap, different methods, techniques and tools for Information Extraction (IE) have been proposed. The main challenge of Information Extraction is the extraction of structured data from diverse sources, be they documents in PDF format, documents with lightweight structure such as HTML, etc.

YAGO [42] and DBpedia [5] developed automatic processes to extract information from the infoboxes of Wikipedia, getting a large amount of data about different entities in several languages. YAGO also includes information from sources such as WordNet and GeoNames, and its information has been linked to DBpedia and other knowledge bases. These two knowledge bases have served for many years as important sources of structured data on the Semantic Web.

Freebase [42] was a collaborative project developed with the aim of allowing users to add and edit structured data, collecting information from different sources. Wikidata [45] incorporates Freebase and also adopts a collaborative platform. The broad domain of these knowledge bases has allowed them to succeed DBpedia and YAGO in becoming the main sources of structured data on the Semantic Web and also a source for different methods of information processing.

One way to increase the availability of structured data on the Web would be to automatically extract such data from legacy unstructured and semi-structured sources. Extracting structured data in formats such as RDF from text implies many difficulties such as ambiguity, lexical and syntactic errors, different languages and dialects, etc., that limit the process; hence some works have rather been interested in extracting RDF from semi-structured HTML elements such as tables. Infoboxes from which DBpedia extracts data are tables of a particular format and there are millions of other

¹<https://data.gov.uk/>

²<https://www.usa.gov/>

³<http://www.bne.es>

⁴<https://musicbrainz.org/>

⁵<http://linkedgeodata.org>

⁶<https://www.uniprot.org/>

tables on the web that contain rich meta-data, which can expand these knowledge bases.

The general process of extracting data from HTML tables involves some important tasks: 1) *table detection*, where tables have to be identified, extracted, cleaned and classified to reject useless tables and 2) *table interpretation* to detect concepts, entities and relations within tables.

The main objective of this thesis is to explore the extraction of structured data (as RDF) from web tables using and enriching knowledge bases, for achieving better results on extraction tasks. These tasks entail diverse challenges due to the diversity of formats in which content is arranged. Many of these difficulties arise from the fact that Web tables (unlike relational tables in databases, for example) are primarily used to make documents easier for humans to read, but not necessarily for machines to read. Some of these difficulties are listed below.

1. Diversity across tables

There are several types of tables whose content is organized in a human readable way such as relational tables (Fig. 1.1), vertical tables like infoboxes (Fig. 1.2), and others that arrange their content in matrices (Fig. 1.3).

Territory ↕	Fatalities ↕	Missing ↕	Damage (2017 USD) ↕	Sources
Sri Lanka	26	N/A	Unknown	[4] [27]
Tamil Nadu	203	N/A	\$155 million	[28] [29] [30] [31]
Kerala	89	141	\$286 million	[28] [32] [33] [34]
Lakshadweep	N/A	N/A	>\$77.5 million	[35]
Totals:	318	141	>\$518 million	[3]

Figure 1.1: Relational table

Also many tables are used to build organizational charts, or as layouts to build aesthetic Web pages as is shown in Figure 1.4

2. Diversity within tables

Table interpretation is also a complex task, due to the existence of tables with multiple headers, combined cells, etc., which complicate the automation of the extraction process. The table in Figure 1.5 gives examples of cells merged vertically (colspan) and horizontally (rowspan).

3. Context

Personal details	
Born	Barack Hussein Obama II August 4, 1961 (age 57) Honolulu, Hawaii, U.S.
Political party	Democratic
Spouse(s)	Michelle Robinson (m. 1992)
Children	Malia · Sasha
Parents	Barack Obama Sr. Ann Dunham
Relatives	Obama family

Figure 1.2: Vertical table (Infobox)




Grand Tour general classification results timeline						
Grand Tour	2013	2014	2015	2016	2017	2018
 Giro d'Italia	—	—	—	—	—	—
 Tour de France	—	—	14	23	10	17
 Vuelta a España	38	8	—	DNF	DNF	—

Figure 1.3: Matrix table

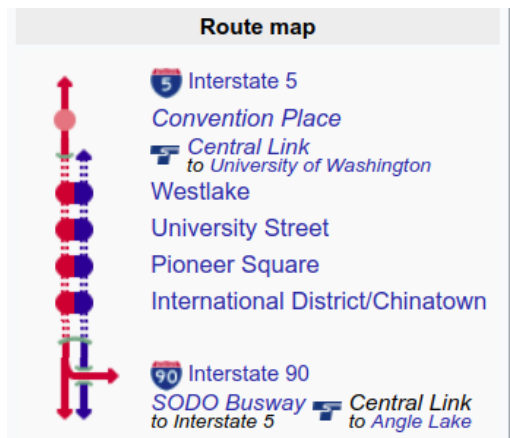


Figure 1.4: Route map for metro bus, designed using a table structure

Headers do not always tell us the full story about the content of cells. Additionally the information in tables may not make sense independently of their context. Many tables describe the main entity of the document or entities described in the content outside the table, which cannot be identified easily. For example the table in Figure 1.6 does not mention the football club that is sponsored by the brands, and in Figure 1.7, the column names say nothing about what was actually won.

Event		2013	2014	2015	2016	2017	2018
 Olympic Games	Time trial	Not Held			—	Not Held	
	Road race	Not Held			DNF	Not Held	
 World Championships	Time trial	—	—	—	—	—	—
	Road race	DNF	19	—	—	37	DNF
 National Championships	Time trial	28	—	—	—	—	—
	Road race	14	12	4	33	—	7

Figure 1.5: Table with combined cells and multiple headers

Period	Kit manufacturer	Shirt sponsor (chest)	Shirt sponsor (sleeve)
1945-1975	Umbro	—	—
1975-1980	Admiral		
1980-1982	Adidas		
1982-1992		Sharp Electronics	
1992-2000	Umbro	Vodafone	
2000-2002			
2002-2006	Nike	AIG	
2006-2010		Aon	
2010-2014			

Figure 1.6: Table with no main entity inside

Wins	Name	Years
7	Nathan O'Neill	1996, 1998, 2002, 2004, 2005, 2006, 2007
3	Rohan Dennis	2016, 2017, 2018
	Luke Durbridge	2012, 2013, 2019
2	Jonathan Hall	1997, 1999
	Cameron Meyer	2010, 2011

Figure 1.7: Table with ambiguous header

4. Representation

Due to the diversity of table formats it is not possible to establish a unique mapping process to extract information appropriately as RDF. For example, an *info-box table* (Fig. 1.2) describes attributes and values of a single entity while a *table matrix* (Fig. 1.3) describes information about different entities in each row, considering the same property in different years. Along these lines tables can

represent the same content in many ways where the extraction process should ideally give the same output for the same content in diverse table representations.

5. Efficiency

There are billions of web pages, many of which have tables with rich information, that can be extracted. Hence the extraction process ideally should be efficient to allow Information Extraction from tables at large scale.

As was mentioned before DBpedia [5] and YAGO [42] developed an automatic process of extraction from infoboxes, but other works have emerged trying to extract information from more diverse tables [47, 13, 25, 15]. These approaches address information extraction tasks over tables such as Entity Linking (EL) and Relation Extraction (RE). Entity Linking maps the recognized entities in a table with their related entities in a knowledge base, while Relation Extraction extracts relations between entities across columns. Despite advances in these topics, works do not achieve high precision and recall for large-scale data, and do not cover all available formats of tables. It is difficult to develop an automatic process to interpret table with good results for the reasons described. More recent approaches [17, 35, 8, 43, 3] develop methods using knowledge bases to guide the extraction process; however many of these approaches limit their process to some specific types of tables, such as relational tables from a specific context referring to people, places and organizations, and achieve results of varying quality (described in Chapter 3).

In order to improve information extraction over tables, in this work we propose a new approach based on merging tables and evaluate it over Wikipedia tables using Wikidata as a reference knowledge base.

We now provide an example to motivate and illustrate our proposal of grouping tables.

1.2 Motivating Example

While the mentioned works are based on identifying the structure of individual tables and extracting the correspondence of its information to entities and attributes in a reference knowledge base, we propose extracting information by first merging similar tables. This proposal aims to solve the problem of not having enough contextual information for extracting relations from each table individually. More specifically, our method extends the extraction process proposed by Muñoz et al. [29] to work with merged groups of tables.

Muñoz et al. [29] extract relations between entities across table columns which involves, detecting entities in table cell contents corresponding to a knowledge base



Figure 1.8: Extracting relations from tables according to Muñoz et al. [29]



Figure 1.9: Example of incorrect candidate triples

entity (they use DBpedia), and extracting existing relations between pairs of entities in the same row in order to propose these relations for the entity pairs in other rows of the table (generating new triples). We provide an example in Figure 1.8, where entities of each column corresponding to the links to Wikipedia articles are mapped to DBpedia entities such as Cristiano_Ronaldo (object) and Manchester_United_F.C. (subject), also the relation team between these entities is extracted from DBpedia; the predicate team is used for generating new triples between the entities of other rows in

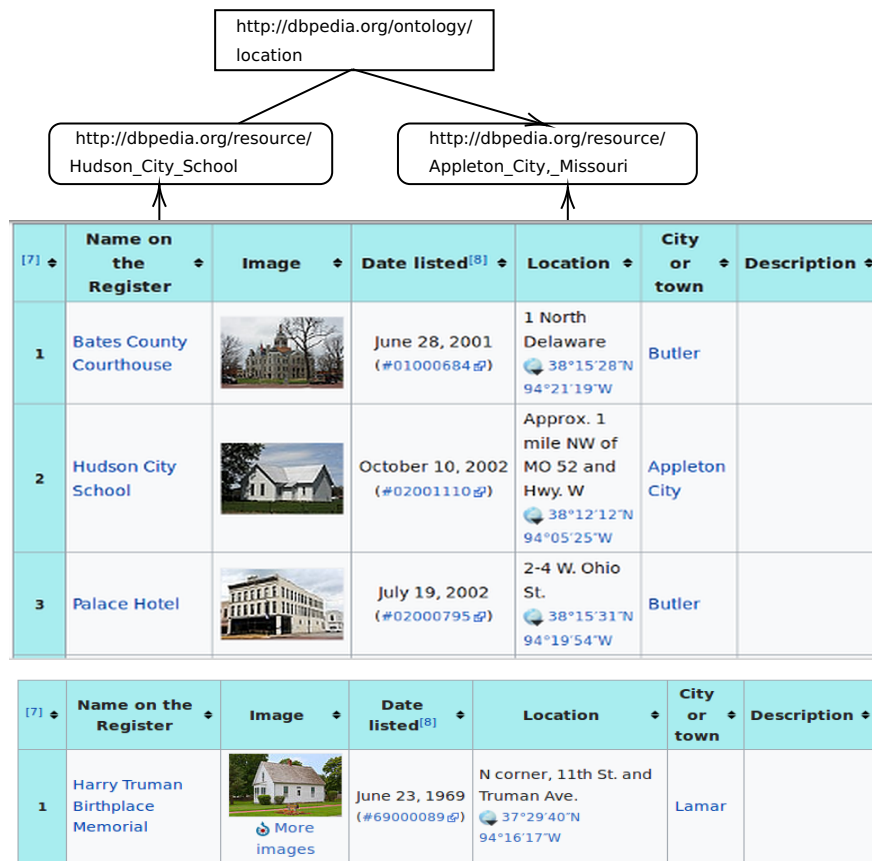


Figure 1.10: Example of small tables with no available candidate triples

the same columns $[Player, Club]$ such as the candidate triple (Lionel Messi, team, Barcelona). Candidate triples that do not already exist in the knowledge base are thus extracted from all tables of Wikipedia. We observe that the relation team proposed for the entities Lionel Messi and Barcelona based on the relation found between entities of previous row is correct, therefore with this method we can generate new triples based on the existing triples extracted from individual tables.

Despite the fact that the proposed relation is correct, this method cannot be generalized completely, even when a higher number of rows with the same predicate is considered. Considering Figure 1.9, if we extract the relations `capitalOf` and `country` from the entity pairs Lima, Peru and Bogota, Colombia, it is difficult to establish which relation is correct for the entities in other rows like Sao_Paulo, Brazil, since both relations have the same number of the occurrences. Furthermore since this approach is based on the existing relations in a table, if an individual table does not have any relation between entities in column pairs, it is not possible to add new triples to knowledge base from that table, as per table of Figure 1.10 where the second table has only one row, and thus no new relation can be proposed for the entities in this row.

In the mentioned work, for classifying the candidate triples as correct or incorrect, Muñoz et al. [29] consider features extracted for each triple, which are fed into a

binary classification; however their approach still does not cover the mentioned cases when there is little evidence for correct triples. More generally, they propose to extract local features from individual tables, but as we see in Figure 1.10, often little information is available given a single table. To solve this issue we propose the idea of merging similar tables where correct relations that appear in few rows will have more supporting information, and small tables will be merged with tables with existing relations; this should offer two main advantages: producing more robust statistics for the purposes of feature engineering and finding additional relations for small tables or tables that otherwise does not have any.

We now describe our main objective and the hypothesis that we propose at the beginning of this work until achieve the mentioned results.

1.3 Objective

The previous examples lead us to propose a *novel method that merge tables according to their structure prior to applying extraction, based on existing information in a knowledge base, with the objective of improving the table information extraction process and, in turn, augmenting the knowledge base.*

1.4 Hypothesis

The hypothesis of this proposal is that *Information Extraction from Web tables can be improved by first grouping and merging the tables according to their content and structure and then applying the extraction process for merged table instead of doing so individually for each table.*

This hypothesis is based on the observation that on sites such as Wikipedia, there are many tables with very similar structure, either because they are created by software automatically or by a manual process where the author of content copies and pastes a table and replaces information. We believe that this similarity in structure across tables forms a "meta-structure" that is exploitable for information extraction.

We will explore this hypothesis in a concrete setting: for extracting RDF triples from tables in Wikipedia, comparing the methods of Muñoz et al. [30] with and without grouping tables. Though we believe that our hypothesis should generalize to other settings involving extracting information from Web tables, evaluation in other settings is considered out of scope for the current work.

Given the hypothesis we pose the following research questions.

Research Question 1

How can merging tables improve the information extraction process?

We believe that merging tables could provide more information for the extraction process, as per the previous examples where using information from similar tables, it is possible to extract more accurate data; although these examples show relational tables, this approach should also be able to extract information from more complex tables as we discussed previously, though only from pairs of cells with entities (not numerical values) in the same row.

Research Question 2

What criteria should be used to merge tables?

The headers of tables can provide some information about the context of a table, but it does not ensure that columns with the same names contain similar information; it is necessary to determine if header information is sufficient to group tables or what additional information should be considered.

Research Question 3

How many tables with the same structure can be merged?

We propose this work based on informal observations of similar tables in Wikipedia; however we need more concrete information about the size of groups of similar tables and the relevance of their information, to better understand the feasibility of our proposed approach.

Research Question 4

How can the benefits of merging tables for automated extraction be evaluated?

It is difficult to evaluate automatic processes of table information extraction, as was mentioned in related works: some previous methods achieve good results but for specific contexts or types of tables. However we aim for a broader setting which considers diverse types of tables and contexts, but for which evaluation is more challenging.

1.5 Results

Our results will show that with our proposal we can increase the number of triples extracted by grouping and merging tables and we can also improve the precision by adding new features for classifying the extracted triples. The results presented correspond to the precision obtained by a classification algorithm (previously validated) that classifies the triples extracted from tables as *correct* or *incorrect*. The validation of these results was performed on a set of triples classified as correct by this algorithm. In this validation a similar precision of 70% and 71% was obtained when grouping and merging the tables and over individual tables, however about a million more triples were obtained by grouping tables. For classifying the triples the algorithm was fed with the features proposed by Muñoz et al. [30]; we also add new features achieving a better precision of 75% compared with 71% obtained by using the baseline features.

1.6 Structure of this work

Next, we describe briefly the topics covered in the next chapters:

Chapter 2: contains a description of the main concepts related to this work, mainly Semantic Web and Machine Learning methods used throughout.

Chapter 3: describes the revision of related works that implement Information Extraction over HTML tables, their contribution and the challenges involved in the proposed approaches.

Chapter 4: contains the proposal of this thesis, describing the necessary steps to achieve the proposed objectives.

Chapter 5: provides an overview of the corpus of tables used, the tasks of preprocessing and data preparation, as well as the features selected for identifying correct candidate triples.

Chapter 6: describes the process of extracting information from the knowledge base, for feature generation and triple extraction.

Chapter 7: contains a brief exploration of groups of similar tables, using clustering methods and visualizations.

Chapter 8: describes the method proposed for grouping and merging similar tables.

Chapter 9: describes the methods and techniques used for triple classification and the results obtained by applying this methods.

Chapter 2

Background

In this chapter we introduce some relevant concepts that are mentioned throughout the work. We describe the standards of the Semantic Web, the data model of the Wikidata knowledge base that was used for mapping table information, the tasks and processes involved in Information Extraction, and some Machine Learning techniques required for the evaluation of the methods proposed.

2.1 Semantic Web standards

For achieving the interoperability of Web content the Semantic Web proposes standards such as RDF, RDFS and OWL. The RDF standard provides a data model based on graphs to facilitate interoperability and file formats such as Turtle, and N-Triples to materialize the information in a structured format. The RDFS and OWL standards allow for defining the semantics used in an RDF dataset. SPARQL meanwhile is the query language used for RDF. We now describe these standards in more detail.

2.1.1 Resource Description Framework (RDF)

On the Web, each element of a Web page that can be accessed independently is considered a *resource*; it may be the entire HTML page or the elements inside it, such as images or style files, etc., that can be accessed through their Uniform Resource Locator (URL) using the transfer protocol such as *http*, *ftp*, etc. On the other hand, the Resource Description Framework considers a resource to be anything with identity, which may include not only web pages but also people, places, books, categories, etc. Given that URLs are used to identify *information resources* (web pages, images, etc.), Uniform Resources Identifiers (URIs), generalize URLs and can be used to identify anything with identity. For example, on Wikidata the URL: <https://www.wikidata.org/>

`wiki/Q9685` identifies a web page displaying data about *Diana, Princess of Wales*, while the URI `http://www.wikidata.org/entity/Q9685` identifies the person herself, and this URI may redirect to the previous URL. The use of URI(s) was extended by another Internet standard, the Internationalized Resource Identifier (IRI) which allows to include more Unicode characters in resource names, rather than just ASCII.

While RDF uses IRIs to identify resources, it defines a data model based on triples of the form `<subject, predicate, object>`, to describe them.

The predicate in an RDF triple provides the property, characteristic or attribute used to describe the subject; it can also be seen as a relation between the subject and object terms.

While the subject and predicate are typically resources identified by IRI(s), the object can also be a literal with a primitive value such as number, date, string, etc. RDF uses many of the same data types as XML: *integer*, *numeric*, *string*, etc. The syntax to define an integer is `"45"^^xsd:integer`; if the data type is not defined, type *string* will be assigned. Furthermore we can establish the language of a string literal with a suffix, for example: `"Princess_Diana"@en`.

The subject and object can also be blank nodes that represent unknown resources and they can also be used to define complex relations. Blank nodes are often serialized with a blank namespace like `_:b`.

```
@prefix ex: <http://example.org/>

ex:PrincessDiana ex:hasName      _:anon1 .
_:anon1          ex:firstName    "Diana"^^xsd:string .
_:anon1          ex:lastName     "Frances_Spencer"^^xsd:string .
```

Listing 2.1: Example RDF Graph in Turtle Syntax

In the example shown in Listing 2.1 the resource `ex:Princess_Diana` has the predicate `ex:hasName` with a complex value represented by a blank node. Figure 2.1 shows a graph resulting from these statements, where the predicates are represented as edges and the subjects and objects form the nodes of the graph. Along these lines, a set of RDF triples is referred to as an RDF graph.

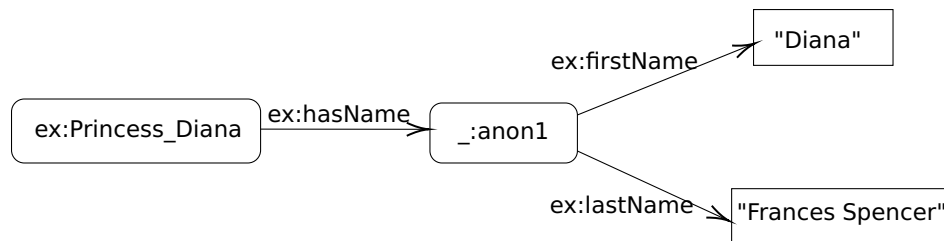


Figure 2.1: RDF Graph for listing 2.1

RDF graphs can be stored using different formats such as RDF/XML, N-Triples and Turtle. We discuss N-Triples and Turtle which describe a set of triples and are amongst

the simplest and most readable of the RDF syntaxes available.

- **N-Triples**

This format represents triples without abbreviations with one triple per line, as shown in Listing 2.2.

- **Turtle**

Turtle format introduces a variety of abbreviations, Listing 2.3 shows an example where the Wikidata resource Q9685(Diana,Princess_of_Wales) is described with the properties P21(gender): Q6581072(Female) and P27(country of citizenship): Q145(United Kingdom). This format is more concise, abbreviating the prefixes of common URI(s) such as wd for entities and wdt for properties, as well as grouping triples with the same subject, for example.

```
<http://www.wikidata.org/entity/Q9685>
<http://schema.org/name>
  "Diana_Princess_of_Wales"@en .

<http://www.wikidata.org/entity/Q9685>
<http://www.wikidata.org/prop/direct/P21>
  <http://www.wikidata.org/entity/Q6581072> .

<http://www.wikidata.org/entity/Q9685>
<http://www.wikidata.org/prop/direct/P27>
  <http://www.wikidata.org/entity/Q145> .
```

Listing 2.2: Example of an RDF Graph in N-Triples format

```
@prefix schema: <http://schema.org/> .
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix wdt: <http://www.wikidata.org/prop/direct/> .

wd:Q9685    schema:name "Diana,_Princess_of_Wales";
            wdt:P21 wd:Q6581072;
            wdt:P27 wd:Q145.
```

Listing 2.3: Example of an RDF Graph in Turtle format

2.1.2 RDF Schema (RDFS)

The information described in RDF graphs can be enriched by creating ontologies. An ontology provides the definition of all concepts and relations with high semantic expressiveness. For example, we could state that since `ex:PrincessDiana` is a `ex:Person`, this means that children of Diana also are persons; these relations can be covered with the semantics of properties and classes defined in RDFS and OWL (Web Ontology Language). We describe some relevant concepts from RDFS in the following.

RDFS was created for representing the semantics of terms used in RDF based on property and class definitions.

A class represents a set of resources, with common properties, such as `Person`, `Build-`

ing, `Movie`, etc., these resources are called instances. For example: `ex:PrincessDiana` is an instance of the class `ex:Person`; RDFS describes this relation with the property `rdf:type`. A class like `ex:Person` is itself considered a resource and an instance of `rdfs:Class`.

Properties in RDF allow to define or describe characteristics of the instances of a class. For example, in the triple `ex:PrincessDiana foaf:gender ex:Female` the property `foaf:gender` describes the gender of the subject entity.

RDFS itself defines some built-in classes. We previously mentioned that `rdfs:Class` is the class of all classes. The class `rdfs:Resource` contains all resources and these are sub-classes of `rdfs:Class`. Literals are represented by the class `rdfs:Literal` and `rdfs:Datatype`; each instance of `rdfs:Datatype` is a sub-class of `rdfs:Literal`.

RDFS also defines a number of built-in properties used to define the semantics of terms. Properties are represented by a class named `rdf:Property`.

The property `rdfs:subClassOf` denotes that all instances of a subject class are also instances of the object class. For example `ex:Musician rdfs:subClassOf ex:Person` means that all instances of `ex:Musician` will be instances of the `ex:Person` class.

The property `rdfs:subPropertyOf` denotes that all pairs of resources related with the subject property are also related with the object property. For example: `ex:placeOfBirth rdfs:subPropertyOf ex:countryOfCitizenship` means that every subject and object related by `ex:placeOfBirth` also can be related with `ex:countryOfCitizenship`.

Properties `rdfs:range` and `rdfs:domain` establish the types for subjects and objects in a triple, respectively. More specifically `rdfs:domain` denotes the class of which the subjects of a property are instances, while `rdfs:range` denotes the class of resources analogously for objects on a property. In Listing 2.4 the property `ex:author` has range of class `ex:Person`, which means that the object of the triples with this property are implied to be instances of `Person`, while the subject is implied to be an instance of the class `Book`.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:Book rdf:type rdfs:Class .
ex:Person rdf:type rdfs:Class .
ex:author rdf:type rdf:Property;
  rdfs:domain ex:Book;
  rdfs:range ex:Person .
```

Listing 2.4: Example domain and range definitions

2.1.3 Web Ontology Language (OWL)

The Web Ontology Language (OWL) provides a set of RDF terms with well-defined meaning for reasoning support that are not included in RDFS. We now describe some OWL terms by way of example to illustrate the idea.

The property `owl:equivalentClass` allows to declare that two classes are synonymous. For example `ex:Person` and `ex:Human`. The property `owl:disjointWith` allows to declare that an object that belongs to a class cannot be an instance of another; for example `ex:Country` and `ex:City`. While the previous terms provide mechanisms to define relationships between classes, there also exist terms in OWL for defining relationships between properties such as `owl:equivalentProperty` and `owl:inverseOf`. For example, the property `ex:hasChild` is equivalent to `ex:isParentOf` while `ex:hasParent` and `ex:hasChild` are inverse properties.

OWL also provides terms for defining classes based on restrictions of the elements that they can contain; these include `owl:minCardinality` which determines the minimum quantity of values allowed for a property on an instance of a particular class, and `owl:maxCardinality` for the maximum quantity of values. For example, the `owl:maxCardinality` of the property `ex:headOfState` on an instance of class `ex:Country` is 1.

We consider that cardinality concept is useful to know if we can propose new objects for a given subject and predicate, for example when the predicate has a cardinality of 1 and there already exists a triple in knowledge base with that value, we may avoid proposing another value for that predicate as part of an information extraction process.

Here we have only described the most relevant features of RDFS and OWL. Other features of these standards are not used in the current work.

2.1.4 SPARQL

SPARQL is an RDF query language for retrieving information stored in RDF format. SPARQL provides a set of query operations such as **join**, **aggregate**, **sort**, and so on. This language allows to generate results in table format using the **select** operation and also in RDF format using **construct**. Listing 2.5 shows an example of a query for getting the name and country of citizenship of a person; the result of this query will be a table with three columns named *firstName*, *lastName* and *country* with their respective values.

The variables of a query take the prefix (?) and it is possible to query for multiple facts in one statement, splitting the facts by (;), and finishing the statement with (.)

```
PREFIX ex: <http://example.org/>
SELECT ?firstName ?lastName ?country
WHERE {
  ex:PrincessDiana ex:hasName ?name;
  ex:countryOfCitizen ?country .
  ?name ex:firstName ?firstName;
  ex:lastName ?lastName.
}
```

Listing 2.5: Example SPARQL query

2.1.5 Web Knowledge Bases

Knowledge bases are large networks of entities, representing their semantic types, properties, and relationships between entities. Web knowledge bases are published online and are accessible to the public; the Semantic Web standards previously described are often used to publish such knowledge bases on the Web.

We review some of the most prominent knowledge bases in practice, that have been published on the Web using the Semantic Web standards.

- **DBpedia**

The first release of DBpedia [5] project was in 2007, and it is updated roughly once a year. The information of DBpedia is automatically extracted from infoboxes, and geo-coordinates of Wikipedia. The resources in this knowledge base also are linked to others such as Freebase, GeoNames, Musicbrainz, Uniprot among others. DBpedia contains about 13 million instances and 6721 mapped properties ¹.

- **YAGO**

The information of YAGO [42] is also extracted from Wikipedia, but it also includes WordNet and GeoNames. Even though it contains fewer properties than DBpedia, YAGO provides more context for its facts using temporal and spatial properties. The latest version, YAGO3, includes about 10 million entities and 120 million facts ².

- **Freebase**

Unlike DBpedia and YAGO, Freebase [32] was a project created for humans to edit structured data directly. It was acquired by Google in 2010 but later retired and integrated with Wikidata in 2015.

¹<https://wiki.dbpedia.org/services-resources/datasets/data-set-38/data-set-statistics>, Dic. 2018

²<https://datahub.io/collections/yago>. Dic. 2018

- **Wikidata**

The Wikimedia Foundation manages the Wikidata Project [45], whose main goal is to be a collaboratively edited knowledge base with a Creative Commons License. Aside from representing Wikipedia data, it extracts data from external public encyclopaedias, making it possible for Wikidata to also feed other sibling projects. Wikidata³ currently describes 44 million entities.

There are some comparative studies (e.g. [19, 38, 1]) of these knowledge bases, where authors contrast characteristics such as languages, domain, access, and structure.

DBpedia and YAGO extract data from Wikipedia periodically, while Wikidata is constantly updated by its collaborators. In this thesis we use Wikidata as a reference knowledge base though the methods we propose should also generalize to similar knowledge bases. The following section provides more details about Wikidata.

2.1.6 Wikidata Knowledge Base

Wikidata maintains a hierarchical structure based on item labels, descriptions, aliases and statements (see Figure 2.2). We now describe these elements in more detail.

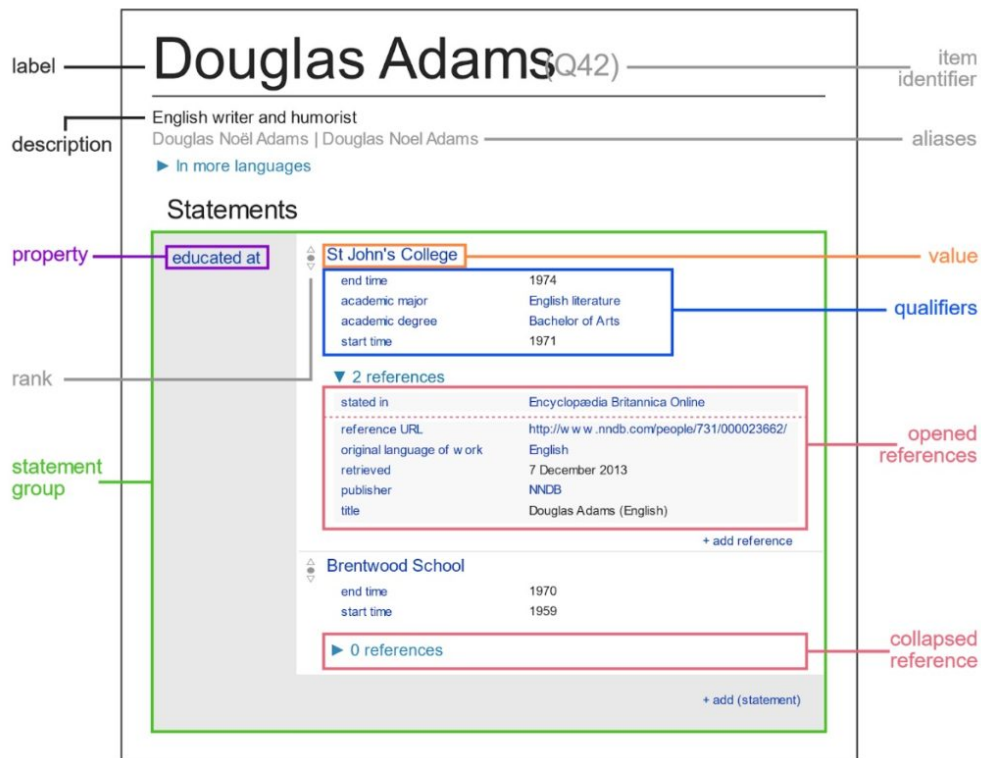


Figure 2.2: Wikidata Model⁴

³<https://www.wikidata.org/wiki/Wikidata:Statistics>. July 2018

⁴<https://www.wikidata.org/wiki/Wikidata:Glossary>

1. Item

An item is any concept, topic or object identified in the knowledge base by a unique identifier with prefix **Q** for entity items and **P** for property items.

2. Label

The name of the item in different languages.

3. Description

A textual summary of the item indicating to which context it belongs (there can be more than one item with the same label). Descriptions can also be given in different languages.

4. Aliases

An item can have more synonyms or expressions with the same meaning. Rather than create a new item, an alias can be added to an existing item. Aliases can be given in multiple languages; in each language, an item can have one label, but multiple aliases.

5. Statements

The statements are the properties of an item, describing facts that may involve relations with other entities. Such facts may be associated with qualifiers that provide additional context for the fact. For example *start date* and *end date* can be used as qualifiers for the *head of state* relation.

Wikidata provides a public SPARQL query service for querying the knowledge base as shown in the example of Listing 2.6.

The query will return a set of items with Q126826 (Aerosmith) on the P463 (member of) property. The query uses a service `wikibase:label`, which allows to get the property `rdfs:label` from the Wikibase Ontology, bound to the variable `?nameLabel`.

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX wikibase: <http://wikiba.se/ontology#>
SELECT ?item ?itemLabel
WHERE
{
  ?item wdt:P463 wd:Q126826.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en".}
}
```

Listing 2.6: Example of a SPARQL Query on the Wikidata Query Service

Wikidata also provides a dump of its content in RDF following the data model shown in Figure 2.3 for representing the information in dump files. As was mentioned before, the prefix `wdt` is used to describe direct relations but for full statements with qualifiers Wikidata uses the prefix `wds`; for example in Figure 2.2 the relation `P69` (educated at) is a full statement with the following qualifiers: *start time*, *academic major*, *academic degree* and *end time*. The statement is described with the relation `p`, and the value of the statement with the relation `ps`; on the other hand the qualifiers are described with

⁵https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format

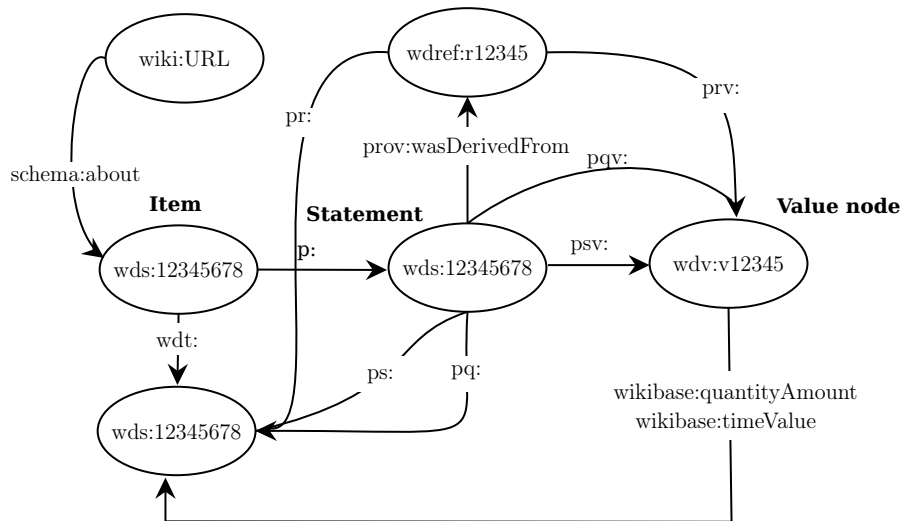


Figure 2.3: Wikidata RDF graph structure⁵

the relation pq. The values also can be new statements, in which case the relations used are psv and pqv and the value is represented with prefix wdv. Finally the relations pr and prv are used to describe references.

The representation of predicate P69(educated at) in Figure 2.2 with the Wikidata structure is shown in Listing 2.7:

```

wd:Q42 p:P69 wds:_x1.
wds:_x1 ps:P69 wdt:691283;
pq:P582 "1974-01-01T00:00:00Z"^^xsd:dateTime;
pq:P812 wdt:186579;
pq:P512 wdt:1765120;
pq:P580 "1971-01-01T00:00:00Z"^^xsd:dateTime.

```

Listing 2.7: Example Wikidata RDF syntax of dump file for a full statement

```

<http://www.wikidata.org/entity/Q42> <http://www.wikidata.org/prop/direct/P69>
<http://www.wikidata.org/entity/Q691283>.

```

Listing 2.8: Example of statement in a truthy dump file

Given the complex structure of the RDF graph representing qualified statements, Wikidata also provides a "truthy dump" with triples representing binary relations without qualifiers. Listing 2.8 shows the truthy version of Listing 2.7.

The Truthy dump file contains the "preferred" statements by predicate for an entity; the "preferred" label is selected by users to indicates the current value for that predicate; for example the entity Brazil has two objects for the predicate capital, however the the current city Brasilia will be marked as preferred, and only this relation will be part of truthy file, but if a predicate does not have a preferred statement all values will be part of this file.

2.1.7 Linked Data

In the same way that HTML pages are linked using hyperlinks forming a Web of Documents, Linked Data defines how RDF can be linked together forming a Web of Data.

Berners-Lee defined the four core principles of Linked Data ⁶.

1. Use URIs as names for things. All resources on the Web must have an unambiguous identifier.
2. Use HTTP URIs so that people can look up those names. The resources must be accessible using HTTP protocol; this makes it possible to interconnect knowledge bases over the web, which is the ultimate goal of LOD.
3. When someone looks up a URI, provide useful RDF information. Providing data in a structured format like RDF allows for software agents to automatically process such content.
4. Include RDF statements that link to other URIs, where users can discover structured data about related things. This principle encourages related descriptions of entities in RDF to be interlinked, forming the Web of Data.

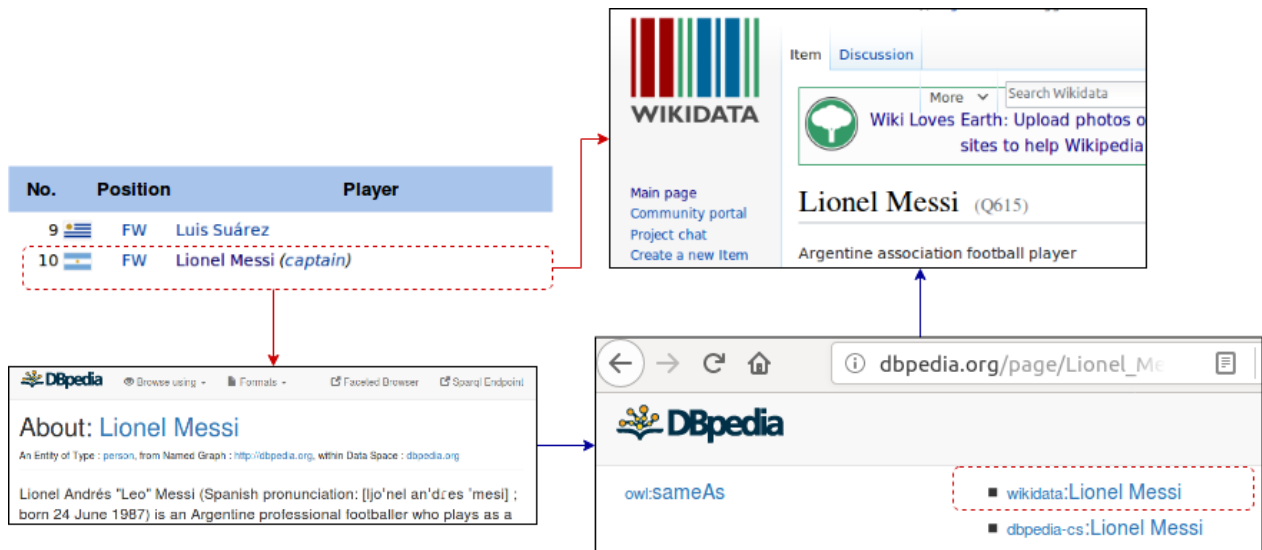


Figure 2.4: Resource linked to Wikidata and DBpedia

An advantage of having tagged items connected to as many knowledge bases as possible (see Figure 2.4) lies in improving, searching and recovering more reliable information.

Beside these principles Linked Open Data (LOD) aims to make data accessible and freely readable on the Web fulfilling the following 5-star recommendations proposed by Berners-Lee.

1. Data should be made available on the web.

⁶<https://www.w3.org/DesignIssues/LinkedData>

2. Data should be machine readable, meaning that a software application can open a document and process its information; for example a scanned document is poorly machine readable.
3. The software required for reading the data should not be proprietary software; for example CSV documents, unlike Microsoft Excel spreadsheets, can be read by any software.
4. Data should identify and describe things, preferably with open standards such as RDF.
5. Data should contain links to external resources to provide context.

The five stars of LOD indicate some simple principles for publishing data on the Web in order to improve the efficiency of searching, processing and retrieving information.

The most recent Linked Open Data cloud, tracking websites following these recommendations, has registered about 1231 knowledge bases ⁷ that accomplish these requirements. However, although LOD has grown in the past years, it still covers a small part of the web; for this reason developing methods of Information Extraction methods for enriching knowledge bases remains of great importance. In the next section we define the process of Information Extraction and describe some key tasks of this process as applied for HTML tables.

2.2 Information Extraction

The main task of the Information Extraction process is to annotate data without structure thus generating or enriching information; this task often relies on Natural Language techniques and Machine Learning processing.

2.2.1 Information Extraction from Text

For annotations over unstructured text there exist three main Information Extraction tasks: Entity Recognition (ER), Entity Linking (EL) and Relation Extraction (RE).

1. Entity Recognition (ER)

Allows to identify and tag entities from a text, e.g., "Sebastián Piñera assumed the position of president of Chile". Some of the works that implement Entity Recognition are based on identifying nouns, adjectives and verbs in a sentence; but this task is not straightforward as entities can be formed by more than one word, the text can be incorrectly spelled or can be semantically incorrect, etc.

⁷<https://lod-cloud.net> (2018-11-26)

2. Entity Linking (EL)

Associates each entity recognized by ER with its corresponding identifier in a knowledge base; e.g., Chile is identified by <http://www.wikidata.org/entity/Q298> in Wikidata and in DBpedia by <http://dbpedia.org/resource/Chile>. In some cases we can have multiple entities with the same name but with different identifiers; automated EL approaches involve Entity Disambiguation to try to differentiate entities with the same name.

3. Relation Extraction

Extracts facts that denote relationships between a pair of entities. In the example sentence "*Sebastian Piñera was born in Chile*" the text between both entities is considered as the relationship. Relation Extraction is considered a challenging task particularly because few sentences are as simple as the example provided.

Various techniques can be combined to identify relations such as: *Part of Speech (POS)* for tagging the parts of a sentence as nouns, adjectives and verbs; and *dependency parsers*, which further annotate the structure of sentences. Using methods of Machine Learning with these features it is possible to localize relations in a sentence, but it is far from straightforward, particularly for relations involving more than two entities. For example: "**Journalist Martin Bashir** interviewed **Princess Diana** for the **BBC** current affairs **show Panorama on 20 November 1995**"; in this sentence there are more than two entities and the relation identified by the verb *interview* has an additional predicate that is the *date of the interview*. Some works focus on extracting binary relations from text while others attempt to extract n-ary relations involving three or more entities. In general, relation extraction from text remains a very challenging problem and is subject to ongoing research.

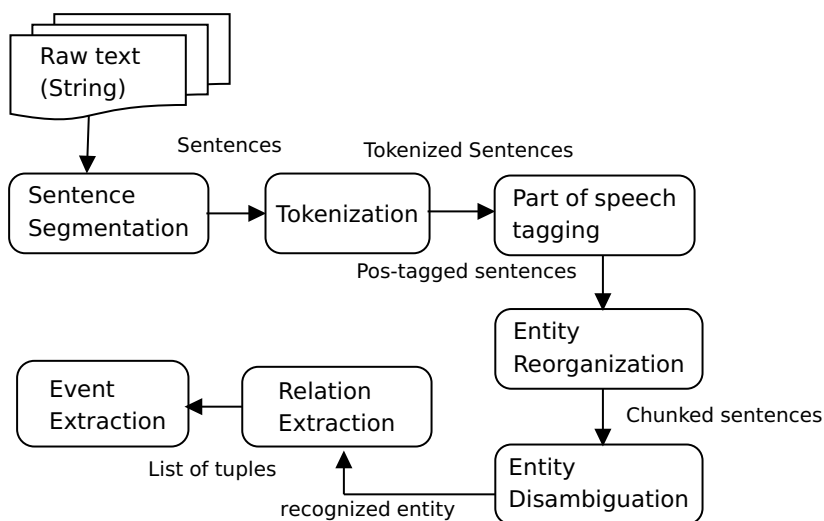


Figure 2.5: Abstract Information Extraction Architecture [12]

Singh et al. [41] defines the general architecture of an Information Extraction process (based on a previous proposal by Constantino [12]), as shown in Figure 2.5. This

architecture includes text processing techniques for identifying entities and relations in raw text. However Information Extraction over semi-structured data may require other methods given that data have lightweight annotations that can be leveraged by specialized processes for Information Extraction.

2.2.2 Information Extraction from Web Tables

Semi-structured data is based on tagged text, where tags give us some semantic information of its content; such data include markup documents in Extensible Markup Language (XML), Hypertext Language (HTML), Portable Document Format (PDF), and so on. Leveraging the hierarchical structure of these documents, many works have been developed for extracting information [2, 14, 37], using meta-tags, titles and section headers, table tags, etc., but to improve the confidence and efficiency of these methods is still a major challenge.

Extracting information from HTML tables involves (aside from the general tasks mentioned before in general process extraction), some specific tasks to recognize and match the document content with a semantic structure. There is no a single agreed upon process for extracting information from HTML tables; however next we describe some relevant concepts in this topic.

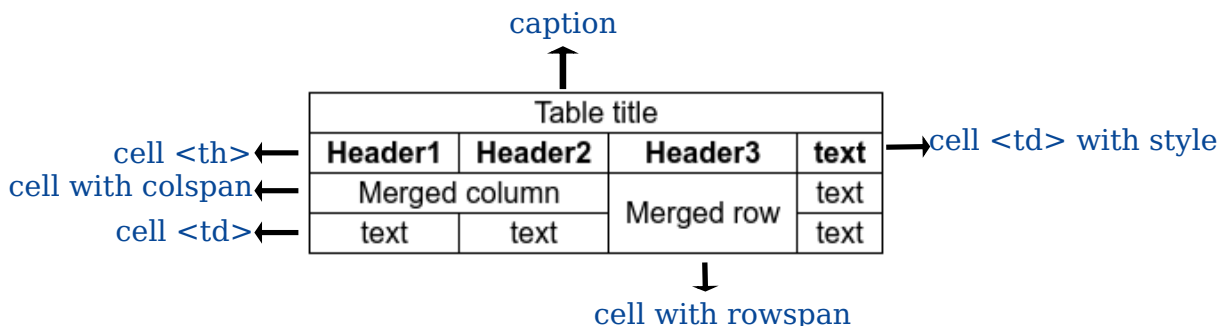


Figure 2.6: HTML table design

HTML tables are generated using the root tag `<TABLE>`, which contains specific tags to add rows and columns. Figure 2.6 shows an example based on the source code of Listing 2.9.

```

<table>
  <tr>
    <th>Header1</th>
    <th>Header2</th>
    <th>Header3</th>
    <td><b>text</b></td>
  </tr>
  <tr>
    <td colspan="2">Merged column</td>
    <td rowspan="2">Merged row</td>
    <td>text</td>
  </tr>
  <tr>
    <td>text</td>
    <td>text</td>
    <td>text</td>
  </tr>
</table>

```

Listing 2.9: Example HTML code for table design

Using the tag `<TABLE>` it is possible to identify tables in a document; however this element is used not only to represent data in web pages, but also for organizing content, such as layout and chart organizers, navigational panes, etc. Information Extraction over HTML tables thus often considers an initial classification of tables in the process, filtering tables that do not contain factual content or potentially applying different processes to different types of tables.

Automatically interpreting table information is another challenging task, where the process has to capture entities, attributes and relationships from tables. The implementation of this process will depend on the table's structure. Hurts et al. [23], describes potential challenges faced during this process, some of which are described in the following:

- Internal cell structure: `<th>` and `<td>` are not the only cell builders; sometimes a `` tag is used to design the table. For example the table in Figure 2.6, uses the tag `` to highlight a header instead of a `<th>` tag.
- Errors: `<colspan>` and `<rowspan>` are not correctly used, specifying erroneous, or empty values.
- Omissions: Empty cells or spaces are some times added to design a more aesthetic table.

In Chapter 3 we discuss in more detail the works developed for table information extraction, which form the related works of this thesis; many such works rely on Machine Learning methods described in the next section.

2.3 Machine Learning methods

There are many Machine Learning models that can be used for clustering, predicting and classifying information; these methods are used for different purposes in Information Extraction, for example for predicting the corresponding entity in a knowledge base for text in a table cell. Additionally some Natural Language processing techniques are implemented to achieve better results in these models. We now describe some of these methods and techniques as relevant for this thesis.

2.3.1 Clustering methods

Clustering is an unsupervised method used for grouping similar data in a given data space; the position of an element in this space will depend on its features. The method is unsupervised because it does not need to learn about data classes using pre-classified elements. There are multiple methods for clustering that differ by the model used, such as connectivity models, centroid models and density models. Given that the central idea for this thesis is extracting information by grouping similar tables we now introduce each high-level method of clustering.

2.3.1.1 Connectivity models

Connectivity models are based on creating initial groups, separating elements in classes and then merging classes as the distance between elements decreases.

Hierarchical clustering is an example of this model; it creates a cluster for each element, and merges near clusters according to a linkage criteria. It alternatively can apply a divisive approach, which starts by creating a big cluster with all elements and dividing it into groups recursively. The distance metric used will depend on the type of data, but the most common metric is *Euclidean* distance, measuring the distance of two points in a Euclidean dimensional space. For merging clusters this algorithm uses linkage criteria that determine the distance between a set of observations:

1. **Complete linkage:** Considers the maximum distance between the elements in a cluster and the rest of the elements. The algorithm joins clusters with minimum distance recursively.
2. **Single linkage:** Considers the minimum distance between the elements in a cluster and the rest of the elements.
3. **Average linkage:** Considers the average distance between the elements in two evaluated clusters.

A hierarchical clustering algorithm uses a dendrogram to represent the clusters

as per Figure 2.7. The final representation depends on the criteria of linkage used. For extracting the final clusters it is necessary to establish the height (distance) where the tree should be cut. For example if we decide to cut the dendrogram at a height of 0.5, it will generate the clusters $\{2,10\}, \{5,8,9\}, \{1,4\}, \{3\}, \{6,7\}$. An alternative method is to determine the maximum number of elements by cluster, where defining a maximum of 3 elements per cluster in the case of Figure 2.7 will result in the aforementioned clusters.

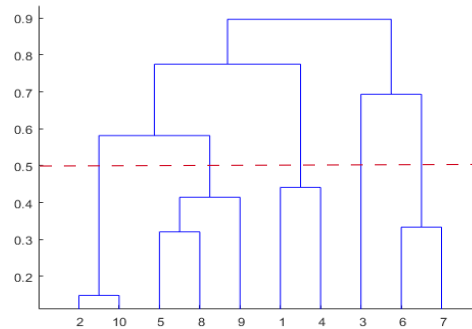


Figure 2.7: Example of a dendrogram in hierarchical clustering

2.3.1.2 Centroid models

The algorithms based on centroid models are iterative and based on the closeness of a data point to the centroid of the clusters. One of the most widely used algorithms is K-means.

K-means aims to find the best centroid position of clusters after a certain number of iterations. It requires as input the number of clusters, and selects randomly this number of points to be the initial centroids. In each iteration the algorithm groups data points to the nearest centroid, and the centroids are updated depending on the distance of all points in the cluster. This algorithm aims to minimize a square error function $\sum_{j=1}^k \sum_{i=1}^n \|x_i^j - c_j\|^2$, using the distance between each point x_i of n data points and the centroids c of each cluster j .

2.3.1.3 Density models

Using density models, clusters are formed by areas of data points with similar density. Data points are joined based on the minimum distance higher than a certain threshold. Such algorithms are based on two principles: the minimum number of points in the neighbourhood of a point (MinPts) and the maximum radius of the neighbourhood (Eps) (maximum distance from core of cluster), which allows to detect and separate border points. Points that are not directly density reachable from a point in a cluster

are considered border points: a point p is directly density reachable from a point q if p is in the set of neighbours of q ($N_{Eps}(q)$) such that the distance $d(p, q) \leq Eps$ and $|N_{Eps}(q)| \geq MinPts$.

DBSCAN is an algorithm based on this model, which allows to create clusters of different shapes; these clusters tend to have similar density, separating points of more sparse clusters as outliers. Variations of the DBSCAN algorithm have been proposed such as **OPTICS** and **HDBSCAN**. OPTICS uses the distance of neighbor points to create a reachability diagram used to separate clusters of different densities, needing to calculate the reachability distance of each point in the cluster to later sort the distances and build the clusters. On the other hand HDBSCAN [9] is an approach that uses hierarchical clustering to avoid introducing noise in clusters of different densities, generating the reachability diagram with the maximum reachability distance between two data points $d_{mreach-k}(a, b)$, having the core distance (radius of cluster given by the central point) for each point in the radius of its k nearest neighbors and the distance between the points $d(a, b)$: $d_{mreach-k}(a, b) = \max\{core_k(a), core_k(b), d(a, b)\}$. Figure 2.8 shows graphically the selection of maximum distance between red and green points. With the distance between all points it is possible to generate clusters hierarchically using single linkage criteria and a specific threshold.

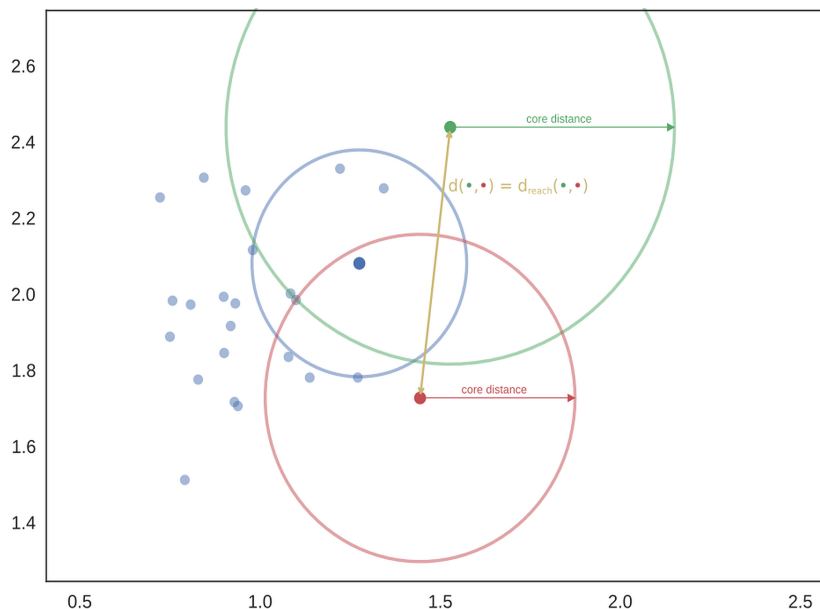


Figure 2.8: Example distance function HDBSCAN ⁸

2.3.2 Classification methods

In our proposal, we use classification methods to label initially extracted relations from tables as correct or incorrect.

⁸<https://hdbscan.readthedocs.io>

Classification methods constitute a variety of supervised algorithms that need predefined classes and data classified in those classes. The algorithms are trained with these data searching for the best fitting of the variables to the model. We will describe briefly some methods that will be used to classify relations extracted from tables.

Logistic Regression: a predictive model to estimate the relation between an dependent variable and one or more independent variables. The outcome (dependent variable) of this method could be interpreted as the probability of a sample to belong to the positive class.

Naive Bayes: is a model based on Bayes Theorem, where the class of samples is taken as the hypothesis and the predictors (features) as the set of events that have to be evaluated to obtain the probability of that sample belongs to the specified class considering that events are independent.

Nearest Neighbours: is a classification algorithm that takes a set of classified points and uses them to learn how to classify other elements closest to them; the similarity is based on the the distance between two (or more) data points in all their features. The class selected for a point is given by the most common class of the nearest neighbour points.

Decision Trees: builds classification or regression models in the form of a tree structure. The nodes in the tree are feature conditions that split the tree into smaller sub-trees. This model selects the features that reduce the impurity of the sub-trees, meaning that the samples in the final node belong to the same class.

Ensemble models: combine various estimators to get a better score for reducing the variance produced by using a single estimator. There are different types of ensemble algorithms based on Decision Trees:

- **Bagging models:** randomly selects a set of instances of the same size as the input data, with replacement (bootstrapping) [16] , and feeds the estimators with each sub-set, building the decision trees based on the best features. For getting the final classification, this model aggregates the weighted class votes from each individual tree (bagging) [6].
- **Random Forest:** is a type of Bagging model; however it builds the estimators with different subsets of randomly selected features, where each feature has the same probability of being selected by each estimator. The predicted class of the instances is selected by aggregating the "votes" predicted classes of all estimators.
- **Boosting models:** unlike Bagging models, Boosting models are built sequentially; each estimator uses the outcomes of the previous estimator and assigns weights to incorrect predictions; thus the next estimator will try to predict the samples with high weights, which are the difficult samples in the training set. Adaptive Boost (AdaBoost) [20] and Extreme Gradient Boost (XGBoost) [11] are the most well known algorithms based on this technique; both algorithms *boost*

the performance of a simple base-learner by iteratively shifting the focus towards problematic observations that are difficult to predict. With AdaBoost, this shift is done by up-weighting observations that were misclassified before while Gradient boosting identifies difficult observations by large residuals computed in the previous iterations [27]. XGBoost, in particular, provides a scalable alternative for sparse data, which has been evaluated and widely recognized in a number of machine learning and data mining challenges. Although some variation of AdaBoost have been developed to improve accuracy results XGBoost leads in some aspects such as robustness to the noise [22]. However, since these models try to predict difficult samples, developing complex models, they require special parameter configuration to deal with the overfitting problem, such as to control the stop criteria and regularization parameters.

While Bagging models attempts to reduce the variance of single estimators, Boosting models try to reduce the bias, increasing the model's ability to classify complex data [36], but both models provide better results than individual learners specially with Classification and Regression Trees (CART) as base estimators.

2.3.3 Evaluation metrics

For getting the performance of methods used for classifying data, four common measures are used: Precision, Recall and F-score and Accuracy. Clustering methods are sometimes also generally evaluated using these metrics, in which case it is necessary to have pre-classified instances in each cluster.

- **Precision** is obtained using the formula: $\frac{|TP|}{|TP|+|FP|}$ where TP are the correct evaluations of a positive class and FP are the elements that do not correspond to the evaluated class. Classification approaches aim to get good precision by avoiding the assignment of erroneous elements to the evaluated class.
- **Recall** is obtained using the formula: $\frac{|TP|}{|TP|+|FN|}$ where FN are the elements not classified in the evaluated class. Classification approaches aim to get good recall by assigning as many correct elements as possible to the evaluated class.
- **F1-score** is the harmonic mean between recall and precision: $2\left(\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}\right)$; this metric penalizes lower values, meaning that if one of the metrics reaches a higher value (1) and the other one a very small value, F1-score also will be small (relative to an arithmetic mean of the two values, for example).
- **Accuracy** is the proportion of correct predicted instances over the total instances evaluated: $\frac{|TP+TN|}{|TP|+|TN|+|FP|+|FN|}$.

The mentioned metrics are used for evaluating and selecting models, looking for the best metric value according to the goal of the problem, but there are also metrics used for comparing the performance of models.

- **Kappa Cohen** is a metric for comparing the instances agreement between two

annotators. In machine learning methods this measure is most often used for comparing the output of a specific model and a random classifier, where a value close or equal to 1 means that the evaluated model is not better than a random classifier. The metric is often also used to measure the agreement score between two selected classifiers, comparing the agreement between both and a random model.

- **Receiver Operating Curve (ROC)** shows the **True Positive Rate (TPR)** or Recall versus the **False Positive Rate (FPR)** reached by the classifier. The ideal point of the curve is the maximum TPR and minimum FPR. The ROC curve is often used to quantify the performance of a classifier for different thresholds levels, thus allowing to understand the trade-off between precision and recall for different confidence levels.
- **Area under the ROC (AUC)** allows to measure if classes can be separate correctly using a probability threshold, where $AUC=1$ indicates that the classes are perfectly separable and $AUC \leq 0.5$ indicates that the model is not working. The AUC metric thus extracts from the ROC curve a single value that can be compared across models.

The mentioned metrics allow to evaluate the methods but we can also improve classification performance by applying some techniques like features selection. Next we explain the metrics used in features selection.

2.3.3.1 Feature importance

Classification is often performed with respect to features extracted for the instances under analysis. Using classical Machine Learning methods, an expert will have to define the set of features to be used according to what they think will help the classification model to discriminate the classes. However, often the expert will not know for sure which features will end up being useful for classification. Having too many (useless) features may add noise, affecting the models ability to discriminate the classes. Hence feature selection is used to select (only) the most important features for classification.

The importance of individual features can be obtained by using different measures; specifically for classification models, the Information Gain measure is applied. Information Gain is defined as how much a feature can tell us about the data, meaning that a feature can divide the instances by class very well. To measure the Information Gain two basic metrics are used: Entropy ($E = \sum_{i=1}^n -p_i \log_2 p_i$) and Gini Index ($GI = 1 - \sum_{i=1}^n p_i^2$); there is no major difference between them besides perhaps the time of computation, since Entropy uses a logarithm function [31].

Given a set of features, a feature is evaluated with a value condition separating the instances into subsets (forming leaves of a tree); it is expected that each leaf only has instances of the same class, in which case the node will have an entropy close to 1.

Since each feature represents a node, the final score is the entropy of the parent node minus the sum of the weighted entropies of its child nodes.

Using the Scikit-Learn python library, feature importance (FI) is obtained using the equation 2.1.

$$FI = \frac{Nc}{N} * [current_{GI} - (\frac{Nr}{Nc} * right_{GI}) - (\frac{Nl}{Nc} * left_{GI})] \quad (2.1)$$

Where Nc is the number of instances in the current node, N the total number of instances, Nr and Nl the number of instances in the right and left nodes (two classes) resulting from splitting the current node, with their respective Gini value (GI). The features with higher score will take more importance in the construction of models.

2.3.4 Text Processing

Since identifying entities, attributes and relations from any source involves reading and understanding text, Natural Language Processing techniques become necessary. We likewise will pre-process text for extracting similar column names, removing suffixes. We now briefly describe two of the most common tasks implemented for text processing as specifically relevant for this thesis work.

2.3.4.1 Stop word removal

There are different methods for removing non-relevant words from a corpus; the basic method consists of removing common words such as articles, prepositions and pronouns that do not give meaning to documents; others include measures such as the method based on Zipf's law that uses frequency of words in the document, removing the most frequent words, singleton words and words with low Inverse Document Frequency (IDF). Methods based on Mutual Information (MI) measure how much information a term can offer about a class of documents, given by the probability of a random term being in a class of documents, etc.; low Mutual Information suggests that the term does not help to discriminate documents as it appears in different classes, and thus it should be removed.

2.3.4.2 Stemming

Methods used for stemming text consist of removing all possible suffixes of a word, thus reducing the vocabulary size, and improving the recall of matching terms. There are various algorithms for stemming text: ones based on clipping words such as Porter's Stemmer, others based on statistical methods such as the N-Gram Stemmer

which evaluates similar words depending on the number of letter grams they share, and algorithms based on co-occurrence of word variants [39], etc.

The Porter Stemmer is the most used stemming algorithm. It is based on rules about the suffix of a word and the number of syllables to know whether the word is long enough to apply the rules. For example when a word ends with *IES*, the end of the word is replaced with *I* (e.g. parties for parti). The disadvantage is that it can produce inappropriate words like *gener* derived from *general*. This algorithm has had some modifications in a second version called Snowball, which includes more rules and exceptions to improve stemming quality.

Chapter 3

Related Work

In the previous chapters we discussed some of the foundational techniques that will be used in this work. We now discuss works directly related to the main theme of the thesis: extracting information from tables.

3.1 Table detection

The first task in table information extraction is to identify the physical structure of tables. Wang et al. [47] propose an approach to classify tables into two groups: *Genuine* and *Non Genuine*; the first group includes tables with some relations between cells and the second one includes those for layout. To label these tables they use two groups of features based on the table structure such as *number of rows and columns*, and *others based on the table content such as data types (numeric, images, hyperlinks, input controls and so on)*; these features feed some machine learning algorithms achieving a 96% F1-measure over 11,477 total tables using cross validation.

A similar classification was proposed by Crestan et al. [13] where the tables are classified as *Relational knowledge tables* and *Layout tables*, with relational knowledge tables further divided into sub-groups.

We now describe the different types of tables identified in such works and that we can find on the web and which will be mentioned in the rest of this work:

- **Relational tables:** The relational table concept is defined as a table with attributes that describe an entity on each row (see Figure 1.1); in some cases one column contains the main entity, however in other cases it is not specified or may be composed from various columns.

- **Listings:** tables where content is arranged in a horizontal or vertical way (e.g. Infoboxes).

Player	Former Team
Parris Duffus	Undrafted Free Agent
Jason Simon	New York Islanders
Stewart Malgunas	Philadelphia Flyers

(a) Horizontal table

The Beatnuts discography	
Studio albums	8
Music videos	9
Singles	21
Instrumental albums	1

(b) Vertical Table

Figure 3.1: Listings: vertical and horizontal tables

- **Attribute-Value:** tables listing properties and values for an entity that often is not inside the table, as per Figure 3.2.
- **Matrix:** tables with both row and column headers, like Figure 3.3.
- **Enumeration:** tables with one row or column that has a list of objects, like the table in Figure 3.4.

Language	Percent of population	Number of speakers	Number who speak English very well	Number who speak English less than very well
English (only)	~80%	237,810,023	N/A	N/A
Spanish (including Spanish Creole but excluding Puerto Rico)	13%	40,489,813	23,899,421	16,590,392
Chinese (all varieties, including Mandarin and Cantonese)	1.0%	3,372,930	1,518,619	1,854,311

Figure 3.2: Attribute-value table

- **Form:** Some web applications use tables to arrange labels like *name*, *email*, *etc.* and input controls; such tables can be detected as tables with relational content, due to the presence of such labels.
- **Layout tables:** Tables used to organize web content. This group is divided into navigational tables like Figure 3.5 and formatting tables like Figure 3.6.

To perform this classification of tables Crestan [13] used features such as: *ratio of empty cells*, *ratio of distinct string cells*, *ratio of numeric cells*, *ratio of header cells*, *ratio of anchor text and cell length*, achieving an overall accuracy of 75%. However the model had difficulties correctly separating vertical listings and attribute-value tables due to both sharing some feature distributions like the ratio of distinct strings and

Climate data for Comodoro Arturo Merino Benítez International Airport, Pudahuel, Santiago (1981–2010, extremes 1966–present) [hide]													
Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
Record high °C (°F)	39.3 (102.7)	37.2 (99.0)	36.8 (98.2)	33.0 (91.4)	31.1 (88.0)	26.7 (80.1)	28.2 (82.8)	29.9 (85.8)	32.9 (91.2)	33.3 (91.9)	34.7 (94.5)	35.0 (95.0)	39.3 (102.7)
Average high °C (°F)	29.9 (85.8)	29.4 (84.9)	27.5 (81.5)	23.0 (73.4)	18.3 (64.9)	15.3 (59.5)	14.7 (58.5)	16.4 (61.5)	18.7 (65.7)	22.5 (72.5)	25.9 (78.6)	28.5 (83.3)	22.5 (72.5)
Daily mean °C (°F)	20.4 (68.7)	19.5 (67.1)	17.5 (63.5)	13.7 (56.7)	10.3 (50.5)	8.3 (46.9)	7.5 (45.5)	8.9 (48.0)	11.1 (52.0)	14.1 (57.4)	16.9 (62.4)	19.3 (66.7)	14.0 (57.2)
Average low °C (°F)	12.0 (53.6)	11.5 (52.7)	9.9 (49.8)	7.1 (44.8)	4.7 (40.5)	3.5 (38.3)	2.5 (36.5)	3.6 (38.5)	5.4 (41.7)	7.3 (45.1)	9.1 (48.4)	11.0 (51.8)	7.3 (45.1)
Record low °C (°F)	2.7 (36.9)	1.2 (34.2)	0.7 (33.3)	−2.6 (27.3)	−5.9 (27.3)	−6.5 (21.4)	−6.8 (19.8)	−6.2 (20.8)	−4.5 (23.9)	−2.8 (27.0)	0.7 (33.3)	3.2 (37.8)	−6.8 (19.8)
Average precipitation mm (inches)	0.4 (0.02)	0.8 (0.03)	6.1 (0.24)	12.0 (0.47)	46.1 (1.81)	68.7 (2.70)	62.5 (2.46)	44.2 (1.74)	20.1 (0.79)	10.0 (0.39)	4.6 (0.18)	1.4 (0.06)	276.9 (10.90)
Average precipitation days	0	0	1	3	5	7	7	6	5	2	1	0	37
Average relative humidity (%)	57	60	65	71	80	84	84	81	78	71	63	58	71
Mean monthly sunshine hours	362.7	302.3	272.8	201.0	155.0	120.0	145.7	161.2	186.0	248.0	306.0	347.2	2,807.9
Mean daily sunshine hours	11.7	10.7	8.8	6.7	5.0	4.0	4.7	5.2	6.2	8.0	10.2	11.2	7.7
Source #1: Dirección Meteorológica de Chile (humidity and precipitation days 1970–2000) ^{[20][21][18]}													
Source #2: Universidad de Chile (sunshine hours only) ^[22]													
Climate data for Quinta Normal, Santiago (1981–2010, extremes 1967–present) [show]													

Figure 3.3: Table Matrix

V · T · E	Presidents of the United States	[hide]
	<ol style="list-style-type: none"> George Washington (1789–1797) John Adams (1797–1801) Thomas Jefferson (1801–1809) James Madison (1809–1817) James Monroe (1817–1825) 	

Figure 3.4: Enumeration table

V · T · E	Microbiology: Bacteria		[hide]
Medical microbiology	infection · Coley's toxins · Exotoxin · Lysogenic cycle · Pathogenic bacteria · resistance		
Biochemistry and ecology	Oxygen preference	Aerobic (Obligate) · Anaerobic (Facultative · Obligate) · Microaerophile · Nanaerobe · Aerotolerant	
	Other	Extremophile · Human flora (Gut · Lung · Mouth · Skin · Vaginal (In pregnancy) · Placental · Uterine · Salivary) · Microbial metabolism · Nitrogen fixation · Microbial ecology · Primary nutritional groups · Substrate preference (Lipophilic · Saccharophilic)	
Shape	Bacterial cellular morphologies · Coccus (Diplococcus) · Bacillus · Coccobacillus · Spiral		
Structure	Cell envelope	Cell membrane · Cell wall: Peptidoglycan (NAM · NAG · DAP) · <i>Gram-positive bacteria only</i> : Teichoic acid · Lipoteichoic acid · Endospore · <i>Gram-negative bacteria only</i> : Bacterial outer membrane (Porin · Lipopolysaccharide) · Periplasmic space · <i>Mycobacteria only</i> : Arabinogalactan · Mycolic acid	
	Outside envelope	Bacterial capsule · Slime layer · S-layer · Glycocalyx · Pilus · Fimbria · Non-motile bacteria	
	Composite	Biofilm	
Taxonomy	Bacteria (classifications) · Bacterial phyla · <i>Former groupings</i> : Schizomycetes · Monera · Prokaryota (Gracilicutes · Firmicutes · Mollicutes · Mendosicutes)		
Book · Category · Commons · Portal			

Figure 3.5: Table used for navigational content

non-empty cells. Although for attribute-value tables their methods achieved a 90% recall, only 10% was achieved for vertical tables. To classify layout tables, features such as the *ratio of images*, *ratio of distinct tags*, and *number of control inputs tags* were included; with these features tables that don't belong to another class fell in the

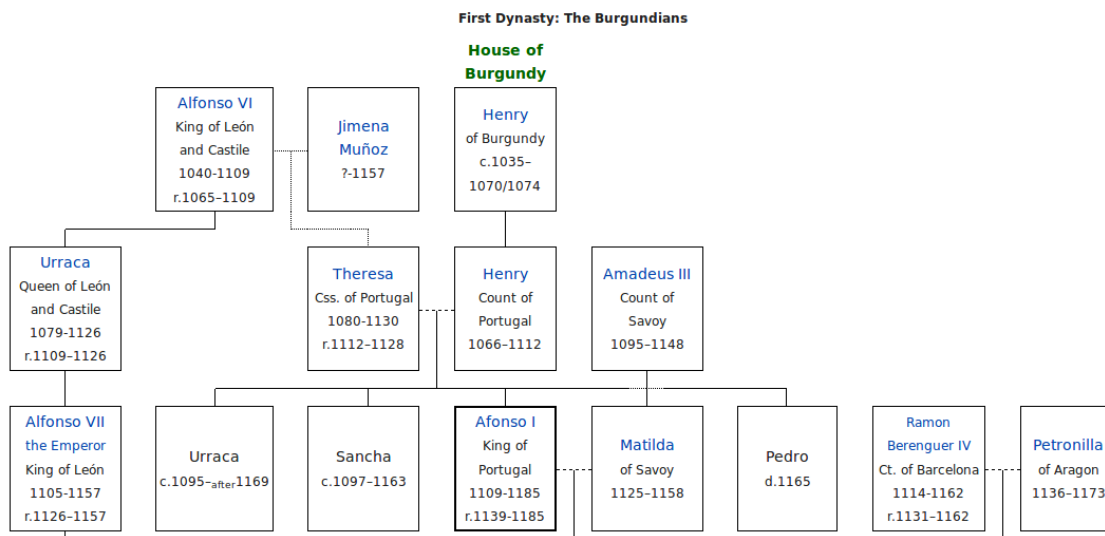


Figure 3.6: Table used as organization chart

formatting group.

Lautert et al. [25] delve further into this taxonomy by making a second classification identifying sub-groups according to their inner structure and content, such as Nested, Split and Multivalued tables. The Nested and Split groups refer to tables included inside others; nested tables are those that describe different information of the same entity in the table (Fig. 3.7) while split tables are those with the same structure arranged into different cells (Fig: 3.8). The Multivalued group gathers tables with simple and compound content, meaning that a cell may have more than one entity (Fig.3.9). Using a dataset of 342,795 tables with 68% layout tables, this work improved upon the results of Crestan et al.'s approach [13] for identifying Formatting, Vertical and Horizontal tables; however for matrices, a low F1-measure of 22% was again obtained. According to Crestan et al. [13], 88% of the tables on the Web are used for layout purposes; in their work a sample of 5,000 from 8.2 billion tables extracted from 1.2 billion web pages was used.

We aim to achieve a large volume of "useful tables", for which we will work with tables extracted from Wikipedia; given the encyclopaedic nature of Wikipedia, we expect more content rich tables and fewer layout tables than would be found on the broader web.

3.2 Table interpretation

Table interpretation can be broken down into a sequence of sub-steps: parsing and normalization, entity and attribute detection, relation detection and table clustering. Given that table interpretation has been explored in a variety of contexts and for a

Team information	
General Manager	John Paddock
Coach	Terry Simpson
Captain	Kris King
Arena	Winnipeg Arena
Team leaders	
Goals	Keith Tkachuk (50)
Assists	Keith Tkachuk (48) Teemu Selanne (48)
Points	Keith Tkachuk (98)

Figure 3.7: Nested Table

No.	Position	Player	No.	Position	Player
1	GK	Miguel Pinto	17	MF	Damián Zamogilny
5	DF	Facundo Erpen	18	DF	Jesús Arturo Paganoni
6	MF	Francisco Torres	19	MF	Saúl Villalobos
7	FW	Flavio Santos	20	DF	Hugo Rodríguez

Figure 3.8: Split Table

Title	Writer(s)
"Burning Feeling"	Jon Stevens, Stuart Fraser ^[5]
"Love Somebody"	Stevens, Fraser ^[5]
"Take Me Back"	Stevens, Brent Thomas ^[6]
"No Lies"	Stevens, Thomas ^[5]
"River of Tears"	Stevens, Fraser, Justin Stanley ^[5]

Figure 3.9: Multivalued Table

variety of applications, it is important to note that different works may focus on or omit some of these sub-steps.

3.2.1 Parsing and Normalization

Given the wide range of formats in which tables may be present and the potential use of merged cells, nested cells, etc., the first challenge is to parse and normalize the tables extracted from a document.

Club performance			League		Cup	
Season	Club	League	Apps	Goals	Apps	Goals
South Korea			League		KFA Cup	
2002	Anyang LG	K-League	0	0	0	0
2003	Cheetahs		0	0	0	0

Figure 3.10: Table with colspan and rowspan

Club performance	Club performance	Club performance	League	League	Cup	Cup
Season	Club	League	Apps	Goals	Apps	Goals
South Korea	South Korea	South Korea	League	League	KFA Cup	KFA Cup
2002	Anyang LG Cheetahs	K-League	0	0	0	0
2003	Anyang LG Cheetahs	K-League	0	0	0	0

Figure 3.11: Normalized table from figure 3.10

TARTAR is an approach [35] that relies on parsing data tables from different sources (such as HTML, PDF, Excel, text) to a different structure: logical frames. The process is based on extracting regions that have attributes (table headers) and instances (values). These regions are generated using vector distances that are built from data cell types and the position of cells. Distance measures allow to distinguish regions or merge them. Hence, it is possible to extract relations from data structured hierarchically as trees where each leaf is an attribute with the list of values of every instance; the instance then has an attribute-value in each leaf. The evaluation of this method was made comparing frames produced by the system against ones produced by humans; the system could transform 85% of 158 tables of which 50% of frames were annotated correctly, corresponding to the expert-defined frames.

Muñoz et al. [29] extract tables from Wikipedia using the TARTAR approach [35]. They further discuss normalization, which aims to generate flat $m \times n$ (m rows and n columns) tables without merged cells or nested structure. They approach this in the natural way, duplicating the content of merged cells into individual cells (see Figures 3.10, 3.11), unnesting tables, etc. However due to the presence of cells with incorrect colspan and rowspan values, they do not succeed in normalizing all tables where 4.2% from 3,923,427 tables were considered ill-formed and filtered from their extraction process.

3.2.2 Entity detection

Entity detection consists of mapping values of table cells into a specific entity in one or more knowledge bases. There are many works that attempt to annotate unstructured text in this way, called Entity Linking, where content around query text is very important to disambiguate an entity. In contrast table cells have sparse context, including for example the table in Figure 3.12, where cell values have multiple entity candidates (mapping to Wikidata entities); even considering text similarity with column names like the column *song*, disambiguating entities in a table can be challenging (in cases where links are not available).

Mulwad et al. [43] introduce the T2LD framework for interpreting and extracting entities and relations from the tables of Wikipedia. They use the Wikitology index

Song	Writer(s)	Original release	Year	Ref.
"Be My Girl – Sally"	Sting Andy Summers	<i>Outlandos d'Amour</i>	1978	[1]
"The Bed's Too Big Without You"	Sting	<i>Reggatta de Blanc</i>	1979	[2]
"Behind My Camel"	Andy Summers	<i>Zenyatta Mondatta</i>	1980	[3]

External entities linked to the table:

- Be My Girl (Q4875538) single by New Kids on the Block
- Be My Girl (Q30668168) single by Eamon
- Sting (Q20152) artefact from J. R. R. Tolkien's fantasy universe
- Sting (Q483203) (Actor) Gordon Matthew Thomas Sumner
- Sting (Q44640) (Musician) Steve Borden

Figure 3.12: Entity detection

for querying a Knowledge Base, with parameters such as: the column header, the row data, the table cell value; and the parameters used in Wikitology index such as: the Wikipedia title, the first sentence of the article, the page rank for the Wikipedia concept, a list of contents, redirects and categories associated with the concept, property values from DBpedia that appear in the infobox, and types related to the concept from Knowledge Bases (Freebase, DBpedia, WordNet and YAGO). The algorithm generates a list of pairs $[c_i, s_i]$, where c_i is a class selected from a list of candidate classes C and s_i is a string from a cell in the evaluated column; the pairs are ranked by the highest ranking instance (R) that matches with c_i ; the score is computed as $score = w \times (1/R) + (w) \times NPR$ where w is a specific weight and NPR the normalized page rank obtained from Wikitology. For example for the column with writers in the previous example, the algorithm will get a set of classes $C = \{dbpedia-owl:Actor, dbpedia-owl:Musician, \dots, c_n\}$, and it will generate the pairs $\{[Sting, dbpedia-owl:Actor], [Sting, dbpedia-owl:Musician], \dots\}$ with their respective rank. This approach is evaluated with tables of people, places, and organizations reaching 83% of accuracy for person entities, 80% for places, 61% for organizations and 29% for other types of entities. Their approach was evaluated against 15 tables with 611 entities.

Limaye et al. [26] introduce an approach to annotate table content based on "lemmas". A set of lemmas contains the entity candidates extracted from YAGO for a value cell, where they use TF-IDF cosine similarity to evaluate if a cell value matches with any of the candidates. They reached an accuracy of 83% for entity annotation.

3.2.3 Attribute detection

The attribute detection task consists in identifying the content of cells that correspond to the attributes of one or more entities in the table. In this setting table headers are considered as attributes, for example in Figure 3.13 the columns *Date of birth*,

Height and Weight can be considered attributes of the entities in the first column; in this case, the names of columns describe perfectly their content, however this work often requires differentiating columns by their name or content and to detect to which entity the attributes correspond.

No.	Name	Date of birth	Height	Weight	1990 club
1	Andrea Gardini	1 October 1965 (aged 25)	202 cm (6 ft 8 in)		Messaggero Ravenna
2	Marco Martinelli	9 October 1965 (aged 25)	200 cm (6 ft 7 in)		Philips Modena
4	Ferdinando De Giorgi	10 October 1961 (aged 29)	178 cm (5 ft 10 in)		Gabeca Montichiari
5	Paolo Tofoli	14 August 1966 (aged 24)	188 cm (6 ft 2 in)		Sisley Treviso

Figure 3.13: Example of attribute detection

Wang et al. [46] use the Probase taxonomy to identify table attributes. The function used receives the set of row content and it returns a triple (*entity_type*, *<set_possible_attributes>*, *confidence*), where confidence indicates how many attributes match with the given query. The evaluation results reached 91% in the average confidence score where 53 of 90 queries returned all-correct results. The main weakness of this approach is the coverage of the taxonomy used, which will not reach a good confidence for unknown contexts. Furthermore, there is the problem of common attribute names; for example in a table with headers [*Name*, *Location*, *Date*], where the information in column *Name* may mention stadiums, hospitals, artworks, etc., it is not possible to assign one of these labels without first exploring the cell content and table context.

Another problem is selecting which column contains the main entity of a table for identifying related attributes. Figure 3.12 shows a table with songs, writers and albums; the column year may be identified as an attribute for song and also for album. Some works use a simple heuristic based on selecting the leftmost column as the main column entity including approaches by Cafarella et al. [7] and Pinto et al. [33], who introduce this approach for querying information over tables. Venetis et al. [44], compare the leftmost heuristic (select the leftmost column) with a classification method using Support Vector Machines (SVM) incorporating features such as: *cells with unique content*, *cells with numeric and date content*, and *the index position of the evaluated column*. They reached an accuracy of 83% selecting the leftmost column and 93% with the classification method.

Not all approaches for table information extraction rely on identifying a main entity. Approaches such as that described by Muñoz et al. [29] can rather extract relations from tables without identifying or requiring a main entity column to be present; we refer to these approaches as relation extraction approaches.

3.2.4 Relation extraction

The main idea of relation extraction lies in identifying relations between entities across columns. We differentiate relations from attributes where we consider that attributes represent specific values of a main entity while relations always are given between entities; Figure 3.14, shows the relation *written by* between columns *songs* and *writers*, and the relation *part of* between *songs* and *original release*. The column headers do not always describe the name of the relation as per the second case, where the relation between the *song* and *album* columns is represented as *part of* in the Wikidata Knowledge base.

Song	Writer(s)	Original release	Year	Ref.
"Be My Girl – Sally"	Sting Andy Summers	<i>Outlandos d'Amour</i>	1978	[1]
"The Bed's Too Big Without You"	Sting	<i>Reggatta de Blanc</i>	1979	[2]
"Behind My Camel"	Andy Summers	<i>Zenyatta Mondatta</i>	1980	[3]

Figure 3.14: Relation extraction from tables

Mulwad et al. [43] extract relations from DBpedia, using an algorithm to get the score of each relation based on the number of rows that have the same relation; the relation with the highest score is selected, and after that, a heuristic is used to get the best relations that describe the table, where the column involved in the most relations is deemed to be the subject column identifying the main entity.

Limaye et al. [26] propose an approach for annotating table content based on lemmas: sets of possible types, entity names and relations for a table taken from the YAGO KB; their approach, reaches an accuracy of 83% for entity annotation, 56% for type annotation and 68% for relations over a corpus of 1,691 entities, 73 types and 10 relations taken from 36 Wikipedia tables; they find similar results for a larger corpus of web tables. One limitation of this approach is that it assumes regular tables without merged cells and a small labeled dataset to evaluate results.

Other relation extraction methods based on local evidence were proposed by Muñoz et al. [30], where they extract existing relations between a pair of columns in tables, further incorporating relations between columns and the article name. Instead of using heuristics, they used machine learning algorithms to predict correct triples trying to reduce the selection of incorrect candidate triples that may appear in common pair columns. For example: consider a table with the column header (*Player, Team, Country*) and with the row (*Lionel Messi, Barcenola F.C., Spain*); if most table rows have the relation *birthPlace* between the first and third column, relation *birthPlace(Lionel*

Messi, Spain) results in an erroneous candidate triple. The goal of the classification step is to distinguish correct and incorrect candidate triples based on a set of features defined on tables, columns and rows. They reached 81% precision over 34.9 million unique triples extracted from 1.1 million Wikipedia tables.

In this work, we propose that if local evidence about tables is increased, relation extraction methods could achieve better results. One such way would be through merging similar tables, which is the subject of the present work. We take the approach defined by Muñoz et al. [29], which works on individual tables, as our baseline method; we adapt it to work with merged groups of tables, comparing the results with and without merging.

3.3 Clustering and merging tables

In this section we describe some works that use techniques to cluster and merge HTML tables, which we believe may be useful for table interpretation, even though most such techniques have been explored in other settings.

3.3.1 Clustering tables

Cafarella et al. [7] presents a relation search engine to retrieval tables given a string query. The list of attributes of a table is used as a schema; for example [*Name, Date of birth, Height, Weight*] is a schema from a table of persons. The attribute *<Weight>* could occur in many schemas, including those from another context like [*Name, Weight, Diameter, Mintage*], that makes reference to Olympic medals for games. They use agglomerative clustering to join schemas with a function to measure if a shared attribute like *Name* plays the same role in similar schemas. The results of a join are presented to the user, suggesting a list of schemas related to the query text so the user can browse through them to get more specific results.

Bhagavatula et al. [3] also introduce a method for mining and matching web tables using semantic metrics. The method consists of joining two tables by similar columns. To predict the most related columns they use the Semantic Relatedness score (SR) [28], which allows to estimate the similarity of two article's context, and the number of shared column values; the pair of columns from the two evaluated tables with higher scores are selected and joined.

A different approach for clustering was proposed by Yoshida et al. [48], where they group tables that share a "unique" attribute in the tables dataset; the *uniqueness of an attribute* means that the attribute identifies a unique context of a table in all datasets; for example: a table column named *<Capital>* identifies with high probability a table

of cities or countries, while another column named *<Station name>* refers to metro transportation routes. The related attributes with high probability of co-occurrence are joined to create an ontology. They chose tables from the top 15 clusters by the number of tables formed from 35,232 tables in Japanese, where each cluster is represented by a unique attribute; the method reached a precision of 94% in a *Capital* cluster; nevertheless for more ambiguous column names as *Name*, *Cache*, etc. the method reached only 50% and 33% of precision.

Cannaviccio et al. [10] introduce a system called Mentor, which extracts RDF triples from groups of tables that share two columns like *[country, capital]*. They predict the relation and type of entities based on a ranking of existing relations between entities across columns, achieving 81% of precision using relational tables from Wikipedia. In this work, tables are decomposed into pairwise binary relations before being merged without considering the adjacent columns of the table and also filtering columns with heterogeneous information; for applying this filter they use the entity types extracted from DBpedia keeping columns with maximum coverage (> 0.8 ratio of entities of a type in a column).

Gentile et al. [21] propose an approach for clustering tables based on word embeddings where table information is transformed to sentences using the *subject column*, *attributes* and *values*, where the subject column is the left-most column with the highest number of unique values of type *string*; the attributes are extracted from the first non-empty row, and use only string cell values. For example for the table in Figure 3.12 and the attributes *[Song, Writer, Original release, Year]*, the sentence *<h:song h:writer h:original_release h:year>* is created. Table 3.1 shows an example table interpretation of table content. They cluster tables computing the cosine similarity between vectors generated for each table. To evaluate the method they compare clustering using word embeddings and bag of words, and use the *pair completeness* measure: the number of entity matches (recall) and the reduction ratio of pair comparisons. They do not evaluate the method in an information extraction process, but achieved 97% recall in 20 clusters using K-means with word embeddings and 50% using bag of words, while for pair comparison reduction ratio they reached 94% and 92% respectively. They use 233 Web tables with 50,072 entity correspondences.

h:song	h:writer(s)	h:original_release	h:year
v:be_my_girl_sally	v:sting_andy_summers	v:outlandos_d'_amour	\\$NUM\\$
v:the_bed's_to_big_without_you	v:sting	v:reggata_de_blanc	\\$NUM\\$
..

Table 3.1: Example of table interpretation per Gentile’s approach

3.3.2 Merging tables

Once tables are clustered, they can be merged to combine their content. In the mentioned works there is no a definition of table merging; however we identify and describe the methods that can be used for this task.

3.3.2.1 Horizontal merging

Horizontal merging requires finding columns on which to merge tables, similar to a join in the relational algebra, as shown in Figure 3.15. From a group of tables, one of them is selected as the source table or source schema to which new columns (attributes) will be added. In this case the type of join can be considered as a left-outer-join.

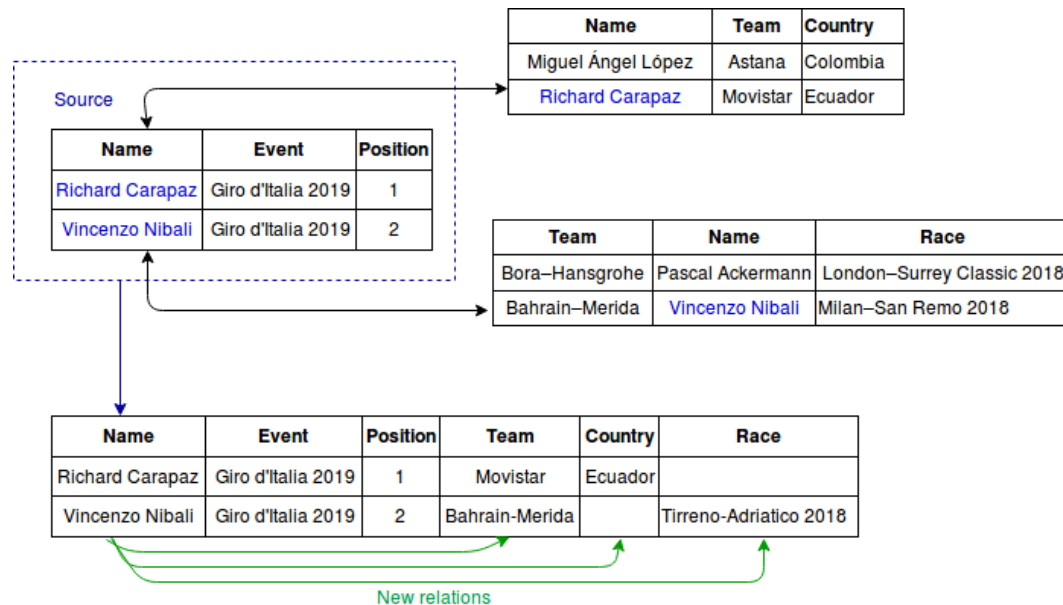


Figure 3.15: Example of horizontal merging

Google Fusion ¹ is an experimental data visualization web application to gather tables for data management using this type of merging as well as Bhagavatula et al. [3] that implements automatic joins for table search tasks, where the system receives a source table and retrieves tables that share values of the column key.

One of the advantages of this method is that it allows to merge information for entities that are in different tables with different columns; however it requires establishing a correct match between cell contents to ensure that the available information is joined correctly.

¹<https://fusiontables.google.com>

3.3.2.2 Vertical merging

Vertical merging consists of creating a new table with rows of grouped tables, similar to a union in the relational algebra. For example in Figure 3.15, the table that would result from vertical merging is shown in Figure 3.16, where a table with headers of both tables was created and all rows were added.

Name	Event	Position	Team	Country	Stage race
Richard Carapaz	Giro d'Italia 2019	1			
Vincenzo Nibali	Giro d'Italia 2019	2			
Miguel Ángel López			Astana	Colombia	
Richard Carapaz			Movistar	Ecuador	
Pascal Ackermann			Bora–Hansgrohe		London–Surrey Classic 2018
Vincenzo Nibali			Bahrain–Merida		Milan–San Remo 2018

Figure 3.16: Example of vertical merging

Vertical merging requires matching only columns rather than cell contents. Cannaviccio et al. [10] offer an example of an approach using this type of merging; they extract and merge bi-column tables (e.g. [Name, Team] from the previous example) based on the type and column names (the type is defined by the entity class identified in each column). Yoshida et al. [48] merge table contents based on a main schema (set of headers) which is generated using a special header that allows to identify the table context, for example *birthday* (a table with *birthday* header might talk about people); a table is generated with this column header and other columns that could appear in the same context such as *hobby*; finally the new table is filled with information from tables with these headers or similar header like *interest* instead of *hobby*.

3.3.2.3 Full merging

Full merging is similar to a full outer join in relational algebra where tables are joined by a key and also new rows are added. Figure 3.17 shows the resulting table of applying this type of merging. We are not aware of a work applying such a merge in the literature.

Name	Event	Position	Team	Country	Stage race
Richard Carapaz	Giro d'Italia 2019	1	Movistar	Ecuador	
Vincenzo Nibali	Giro d'Italia 2019	2	Bahrain–Merida		Milan–San Remo 2018
Miguel Ángel López			Astana	Colombia	
Pascal Ackermann			Bora–Hansgrohe		London–Surrey Classic 2018

Figure 3.17: Example of full merging

3.4 Summary

Table information extraction involves different perspectives derived from the processes mentioned in this chapter, such as entity and attribute detection and relation extraction. Each work achieves different results due to the data and techniques used. Table clustering has also been explored by a few works; the works of Cannavicio and Gentile propose techniques that promise good results, grouping tables in different ways. The former work clusters relational tables by pairs of columns for extracting relations, but does not consider the full table structure; the second work uses word embeddings from the content of tables for mapping entities but uses a small dataset and was not evaluated for the purposes of relations extraction. We believe that grouping tables can improve information extraction in this setting and we evaluate this hypothesis by extracting relations and comparing the results with and without clustering. In the next chapter we present our proposal in various steps including grouping tables.

Chapter 4

Proposal

In this chapter we introduce our proposal briefly, describing the main tasks: the extraction of the table corpus, the preparation of the knowledge base for querying and feature extraction, and finally the extraction of triples. In the next chapters we will address individual steps of the whole process in more detail.

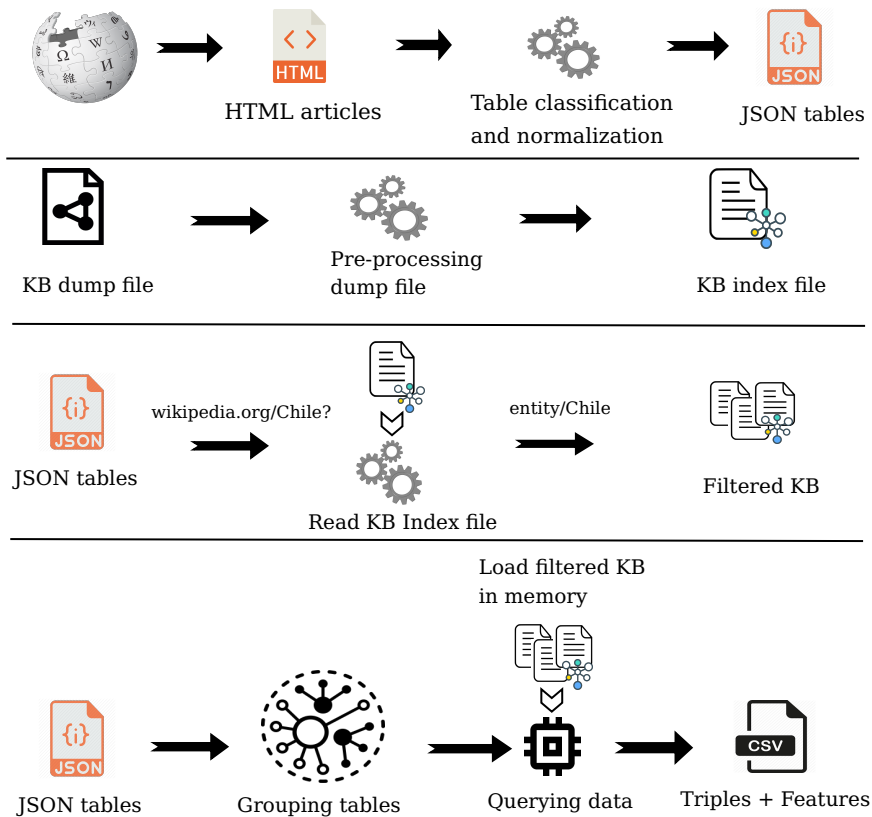


Figure 4.1: Proposal: Table relations extraction

4.1 Table extraction and normalization

The first task involves the extraction of HTML pages; in this work, we take the English version of Wikipedia. Wikipedia has not provided a dump in HTML format since 2016; currently it has dumps in XML and OpenZim formats, which can be read for extracting tables, but there is also the possibility of downloading HTML files one by one using an API, considering a filter for English articles. We extract the tables from Wikipedia by using this API.

The HTML files have to be processed for extracting tables, trying to reject useless tables such as layout tables, which often do not contain factual data and infoboxes (extraction from which has been well-covered by previous works, such as DBpedia [5] and YAGO [42]).

There are many tables with complex designs as per the example in Figure 4.2. To read these tables automatically, it is necessary to convert them to matrices splitting merged cells and extracting inner HTML tables.

Header		Tennis Club Las Terrazas Miraflores, Lima, Peru ^[1]			
	Bahamas	17 April 2019		Barbados	
	3	Clay		0	
				Header	
			1	2	3
1	Danielle Andrea Thompson		6	6	
	Melena Lopez		0	0	
2	Kerrie Cartwright		6	6	
	Tangia Riley-Codrington		0	0	
3	Sydney Clarke / Sierra Donaldson		7	6 ¹	7
	Melena Lopez / Chloe Weekes		5	7 ⁷	5

Figure 4.2: Complex table design

Name	Park Güell	Palau Güell	Casa Milà	Casa Vicens
Code, year	320-001, 1984	320-002, 1984	320-003, 1984	320-004, 2005
Coordinates	41°24′59.6″N 2°09′07.9″E	41.379183°N 2.174445°E	41°23′51.3″N 2°09′46.9″E	41°22′50.5″N 2°10′30.6″E

Figure 4.3: Table with different type of data

The challenges of this tasks are to keep the cell distribution and avoid losing the table's meaning, to support different and complex table designs, and to deal with some design errors, and different types of content (see Figure 4.3).

Year	Awards	Category	Nominated work	Result	
1995	Independent Spirit Awards	Best Supporting Male	<i>Fresh</i>	Nominated	
1995	National Board of Review	Best Cast	<i>The Usual Suspects</i>	Won	
1999	NAACP Image Awards	Outstanding Supporting Actor in a Drama Series	<i>Breaking Bad</i>	Nominated	
2011				Nominated	
2011	Saturn Awards	Best Guest Starring Role on Television		Nominated	
2012	Critics' Choice Awards	Best Supporting Actor in a Drama Series		Won	
2012	Primetime Emmy Awards	Outstanding Supporting Actor in a Drama Series		Nominated	
2012	Satellite Awards	Best Supporting Actor – Series, Miniseries or Television Film		Nominated	
2012	Saturn Awards	Best Supporting Actor on Television		Nominated	
2012	Screen Actors Guild Awards	Outstanding Performance by an Ensemble in a Drama Series		Nominated	
2013	Saturn Awards	Best Supporting Actor on Television		<i>Revolution</i>	Nominated
2019	Screen Actors Guild Awards	Outstanding Performance by an Ensemble in a Drama Series		<i>Better Call Saul</i>	Nominated
2019	Primetime Emmy Awards	Outstanding Supporting Actor in a Drama Series	<i>Better Call Saul</i>	Pending	

(a) Wikipedia article:Giancarlo_Esposito

Year	Award	Category	Nominated work	Result
2018	Screen Actors Guild Awards	Outstanding Performance by an Ensemble in a Drama Series	<i>Stranger Things</i>	Nominated
2018	MTV Movie & TV Awards	Best On-Screen Team (with G. Matarazzo, F. Wolfhard, C. McLaughlin, N. Schnapp)		Nominated

(b) Wikipedia article:Sadie_Sink

Figure 4.4: Two example HTML tables from Wikipedia

In this step the tables are extracted and normalized individually preserving the article's title, and serialized in JSON format; in this way we attempt to group similar tables that belong to the same or different files, as per the example in Figure 4.4 whose tables correspond to the articles related to the actor Giancarlo Esposito and the actress Sadie Sink respectively. The result of normalizing Table 4.4b is present in Table 4.1; the cell with the content *Stranger Things* was split, thus we can extract relations between the entities mapped from each row.

Table 4.1: Example of table 4.4b normalized

Year	Awards	Category	Nominated work	Result
2018	Screen Actors Guild Awards	Outstanding Performance by an Ensemble in Drama Series	Stranger Things	Nominated
2018	MTV Movie & TV Awards	Best On-Screen Team (with G. Matarazzo, F. Wolfhard, C. McLaughlin, N. Schanapp)	Stranger Things	Nominated

4.2 Knowledge base querying

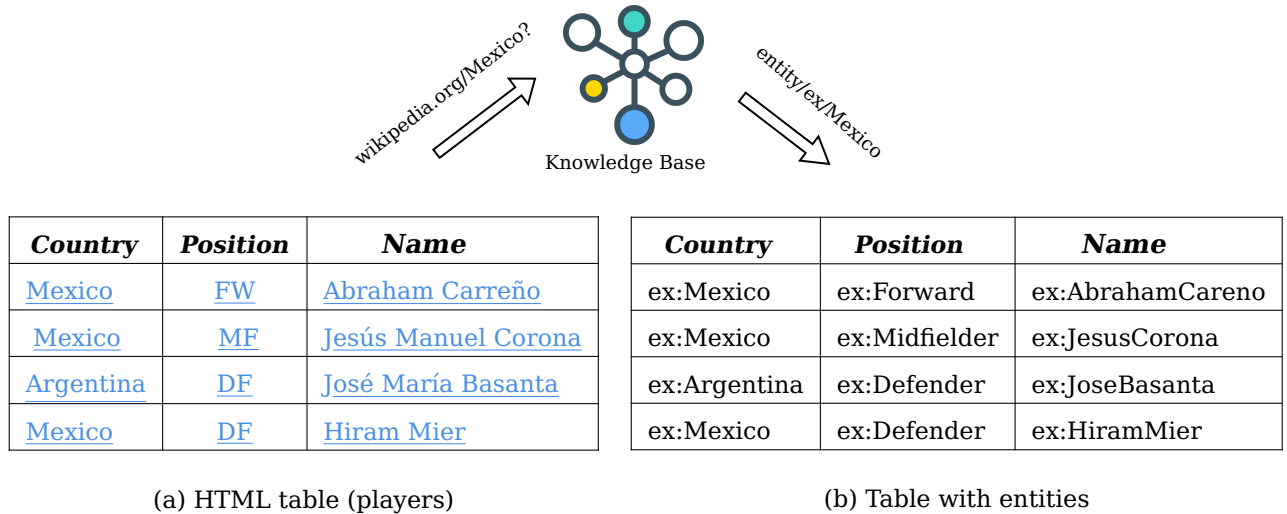


Figure 4.5: Example entity extraction from HTML tables

Our method for extracting information from tables relies on a reference knowledge base, where we use Wikidata. For each table we need to detect entities in Wikidata mentioned in each cell, extract the existing triples (binary relations) between pairs of entities on the same row, and extract statistics about the entities and triples' predicates in order to build features which we will use for classifying correct triples.

We considered three ways of querying the knowledge base for extracting information. Wikidata has a SPARQL query service for extracting up-to-date information, however the query time is too slow, considering an average delay of 5 seconds; the second option is to use a local endpoint to avoid the delay time, but due to the large size of Wikidata, the use of on-disk indexes by most SPARQL engines, it also takes too much time for the amount of random-access style queries that are required. The last option is to load the required information for tables in memory. We adopt the last option of loading and querying Wikidata locally in memory. To make the data fit in memory, the dump file needs to be pre-processed, extracting only entity, relation and predicate information relevant to the table corpus; this attempts to avoid overloading the in-memory index and reduce query time.

There are many methods for mapping text to entities, as we have shown in previous chapter, and various Entity Linking techniques allow extracting Wikipedia links corresponding to identified entities. Considering this insight we map the links in table cells to the corresponding entities in the knowledge base. This task uses the links provided by Wikidata between its entities and Wikipedia articles, but it requires some preprocessing for correctly identifying links and looking for redirect pages, since Wikipedia manages multiple URLs for one resource. Figure 4.5 shows an example of mapping the links from a table to entities. Where a table cell does not offer a link to a Wikipedia article we currently do not consider it in the extraction process.

Though not the focus of the current work, this limitation could be overcome by using the aforementioned Entity Linking techniques [26, 40, 4]. We create an index (for querying the knowledge base) only with the Wikidata entities mapped from the tables in our corpus, as well as the relations found between the pairs of entities. Figure 4.6 shows the corresponding entities from Wikidata to the Wikipedia links in the table (previously normalized); also we extract the existing relations between the mapped entities including the article entity, for example the relation `cast member:P161` between `Stranger Things:Q19798734` and `Sadie Sink:Q27452406`. The index that we keep in memory consists on a pair of *key* and *value* (k, v), where the key corresponds to the Wikipedia link and the value to the Wikidata ID, for example (`https://en.wikipedia.org/wiki/Stranger_Things`, `Q19798734`) and also the existing relations between the entities (`Q19798734, P161, Q23452406`); thus we will look for the Wikidata ID corresponding to a hyperlink and get the existing relations between two entities. This initial process will allow us to create a "filtered knowledge base" that can be stored in memory for later look-ups.

Year	Award	Category	Nominated work	Result
2018	Screen Actors Guild Awards	Outstanding Performance by an Ensemble in a Drama Series	Stranger Things	Nominated
2018	MTV Movie & TV Awards	Best On-Screen Team (with G. Matarazzo, F. Wolfhard, C. McLaughlin, N. Schnapp)	Stranger Things	Nominated

Figure 4.6: Mapping entities from a normalized table

4.3 Grouping tables

Considering a small table like Figure 4.4b, if it does not have any relation between entities across columns, we could propose the relations extracted for entities of table in Figure 4.4a by merging (taking the union of) the rows of both tables; considering that they have the same headers, as per the example in Table 4.2. To achieve this we elaborate on the criteria for grouping and merging tables that have the same structure.

The task of grouping tables requires a preliminary analysis to establish characteristics of the data, an appropriate similarity measure, and the method for grouping tables, particularly considering that our corpus contains tables from multiple topics; therefore the main challenge for us is grouping and merging tables without selecting a specific knowledge area within a large corpus while trying to support a variety of different types of table structures.

Table 4.2: Example of merging two tables with same headers (Colors pink and green identify the rows that come from both tables from example in Figure 4.4

Year	Awards	Category	Nominated work	Result
1995	Independent Spirit Awards	Best Supporting Male	Fresh	Nominated
1995	National Board of Review	Best Cast	The Usual Suspects	Won
...
2018	Screen Actors Guild Awards	Outstanding Performance by an Ensemble in Drama Series	Stranger Things	Nominated
2018	MTV Movie & TV Awards	Best On-Screen Team (with G. Matarazzo, F. Wolfhard, C. McLaughlin, N. Schanapp)	Stranger Things	Nominated

4.4 Relation Extraction

There are many tasks in table information extraction, and many of them have been implemented in different works, mentioned in the related work chapter; some of these works are focused on extracting entities with their properties and corresponding classes, while others are focused on extracting relations between entities. We will focus on extracting relations across table columns, extracting the existing relations for entities in a pair of columns and using these reference triples for generating new candidate triples.

Associating properties to pairs of table columns, we extract binary relations that do not already exist in Wikidata, which are proposed as *candidate relations*. Consider, for example, in the merged Table 4.2 the relation (Q132351,P1411,Q697007) in Wikidata indicating The Usual Suspects *nominated for* Best Cast, then we will propose this relation to generate a candidate triple such as (Q19798734,P1411, Q2530270) indicating Stranger Things *nominated for* Outstanding Performance by an Ensemble in a Drama Series.

Within the set of candidate triples we also can find some incorrect relations, such as (Q19798734,P166,Q268200) corresponding to Stranger Things *was awarded* Outstanding Performance by an Ensemble in a Drama Series, because the exiting relation between The Usual Suspects and Best Cast; hence we have to identify which triples from the set of candidate triples are correct; the next step is to perform a binary classification to attempt to identify extracted relations as correct or incorrect.

4.5 Classification and validation

We perform supervised classification of triples, by training Machine Learning algorithms with triples previously classified as correct or incorrect. Different algorithms are validated to determine which performs better for our data, using validation and test sets of triples (labeled by two annotators). The input for these algorithms consists of features defined in Muñoz work [30] and some new features proposed by us. The features are extracted from: the table, like the length of the content cell from which entities were extracted; the occurrences of the predicate in the table; and some statistics related to the proposed predicate such as the number of objects per subject existing in the knowledge base.

The best algorithm is selected and trained with a set of labeled triples and later evaluated over the set of candidate triples; finally a sample of triples classified as correct are randomly selected to manually verify if they were indeed correctly classified. The final results consist of the precision obtained from this verification.

In the next chapters we discuss each step with more detail.

Chapter 5

Table Corpus

In this chapter we describe the corpus of tables used in this thesis, how it was acquired, the data preprocessing applied over it, as well as data exploration results to better understand the contents of the corpus.

5.1 Tables extraction

A total of 5,582,225 articles were downloaded from English Wikipedia. For getting these articles we used an XML dump to extract article's titles, since Wikipedia does not have dump files in HTML format using these titles; we then downloaded HTML pages using the Rest API of Wikipedia ¹. (See Figure 5.1). From these pages we extract a corpus of 17 million tables.

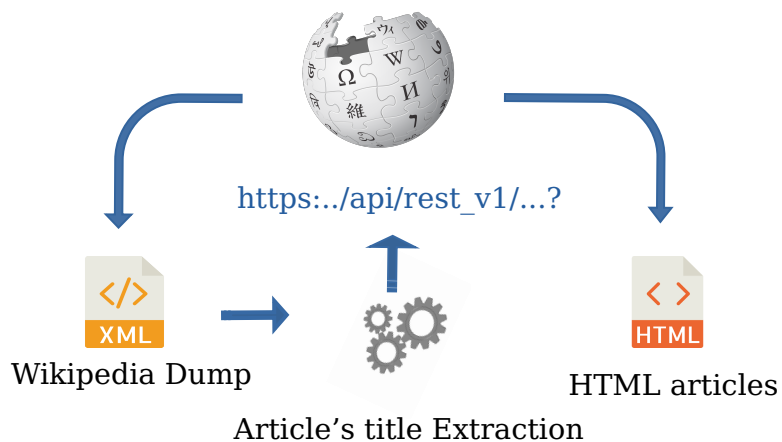


Figure 5.1: HTML files extraction

¹https://en.wikipedia.org/api/rest_v1/

5.2 Tables classification

We perform an initial classification of the tables in our corpus (see Table 5.1) to detect tables with useful content, based on Muñoz et al.’s [29] approach, where the style classes of the tables are employed for detecting *infobox* tables used for defining attributes and values of the article’s main entity; layout tables, such as *toc tables* used for designing tables of content; and *format tables* that contain tables with navigation controls and message boxes. In this classification we found that 52.5% of tables in the dataset are format tables and 21.7% are *infoboxes*; for our purposes we do not consider infoboxes since information extraction for this type of table is already well-covered by works such as DBpedia [5] and Yago [42].

Table 5.1: Tables classification

Tables	Total	%
Infobox	3,816,975	21.7%
Small (dim<2x2)	566,883	3.22%
Toc	496	0.002%
Layout tables	9,211,566	52.5%
Tables with usefull content	3,957,549	22.5%
Total	17,553,469	100%

5.3 Tables normalization

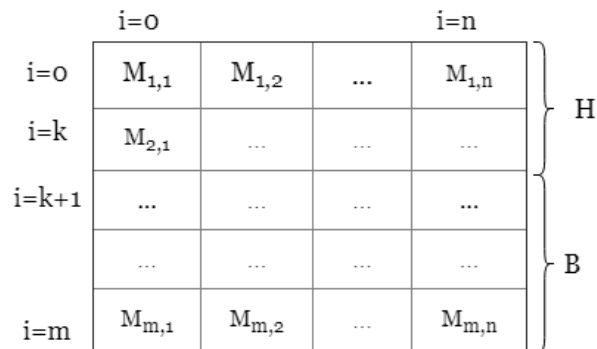


Figure 5.2: Table normalization model

After the initial classification we perform normalization, following the next steps:

- Split merged cells (colspan and rowspan) duplicating their content.
- Extract and separate tables arranged inside other tables per Figure 5.3.

Table1

Table2					Table3				
Medal	Name	Sport	Event	Date	Medals by sport				
Gold	Dalma Ružičić-Benedek Milica Starović	Canoe sprint	Women's K-2 500m	16 June	Sport	1	2	3	Total
Gold	Marko Novaković Nebojša Grujić	Canoe sprint	Men's K-2 200m	16 June	Shooting	4	0	0	4
Gold	Andrea Arsović	Shooting	Women's 10 metre air rifle	16 June	Canoeing	2	1	0	3
Gold	Zorana Arunović	Shooting	Women's 10 m air pistol	17 June	Sambo	1	0	0	1
Gold	Damir Mikec	Shooting	Men's 10 m air pistol	17 June	Water polo	1	0	0	1
Gold	Damir Mikec	Shooting	Men's 50 metre pistol	20 June	Taekwondo	0	2	0	2
Gold	Table4 Men's junior national water polo team		Water polo	Men's tournament	Wrestling	0	1	0	1
	Marko Janković	Stefan Todorovski			Basketball	0	0	1	1
	Petar Kasum	Marko Tubić			Swimming	0	0	1	1
	Jasmin Kolašinac	Petar Velkić			Volleyball	0	0	1	1
	Nikola Lukić	Matija Vlahović			Total	8	4	3	15
	Vladan Mitrović	Đorđe Vučinić							
	Dragoljub Rogać	Uroš Vuković							
Kristian Šulc									

Figure 5.3: Table with inner tables

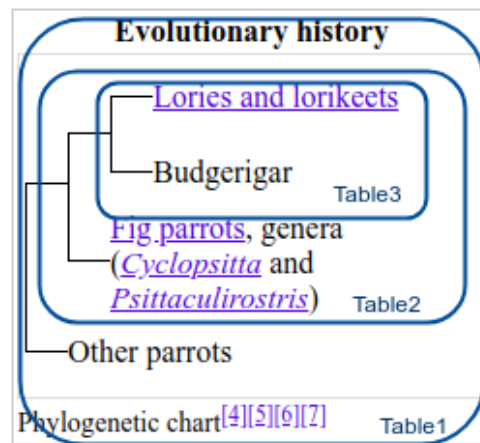


Figure 5.4: Table with inner tables used as an organizer chart

The result of table normalization is an $m \times n$ matrix M_T for each table T in the set of tables \mathcal{T} , from which the set of headers H_T of length n is extracted. The table header may be composed of multiple rows (the table caption was extracted separately and is not part of the table matrix), hence we identify the table body as B_T for $(i \leq m, j \leq n)$, (see Figure 5.2).

A total of 3,967,412 (98.68%) tables were successfully normalized (the number of total tables increased because of the extraction of inner tables).

As we require headers for grouping tables, we reject tables without headers. While some tables without headers are part of the original set, others are the result of unnesting tables (per Figure 5.4) where all are discarded. As a result a total of 3,631,229 (90.31%) tables with headers are considered. In the next section we describe the

Table 5.2: Results after table normalization

Tables normalized	3,967,412	98.68%
- Wikitable	2,479,245	61.66%
- Tracklist	128,194	3.19%
- Multicol	146,950	3.67%
- Others (collapsible, no style class, etc.)	1,213,023	30.17%
Tables not normalized	52,999	1.31%
- Ill-formed	4,488	0.11%
- With inner tables	48,511	1.20%
Total	4,020,411	100%

Table 5.3: Useful tables

Useful tables	3,631,229	90.31%
- More than one column header	3,523,890	87.64%
- One column header	107,339	2.66%
No-useful tables	389,189	9.68%
Total	4,020,411	100%

preprocessing tasks for detecting table headers.

Table title	Wettest tropical cyclones and their remnants in Antigua and Barbuda						
	Highest-known totals						
Table headers	Precipitation			Storm	Location	Ref.	
	Rank	mm	in				
Table body	1	465.3	18.32	Lenny 1999	V. C. Bird International Airport	[2]	
	2	252.5	9.94	Luis 1995	V. C. Bird International Airport	[3]	
	3	245.8	9.68	Frederic 1979	V. C. Bird International Airport	[3]	
	4	232.6	9.16	Omar 2008	V. C. Bird International Airport	[3]	

Figure 5.5: Table structure recognition

5.4 Table Interpretation

The normalized tables are saved in a JSON format keeping their HTML content, which allows to interpret the table structure and extract specific information such as the table title and headers. The content of JSON files is described following:

```
{
  "tableId": "942176.1",
  "title": "Wettest tropical cyclones and their remnants in Antigua and Barbuda Highest-known totals",
  "htmlMatrix": [[..],[..],[..]],
  "startRows": 2,
  "colHeaders": ["precipitation**rank@1", "precipitation**mm@1", "precipitation**in@1", "storm**storm@3", "location**location@3", "ref**ref@1"],
  "rowHeaders": [],
  "nrows": 4,
  "ncols": 6,
  "tableType": "WELL_FORMED",
  "articleTitle": "List of wettest tropical cyclones by country",
  "articleId": 942176,
  "attrs": {class: ["wikitable"],...},
  "articlePath": '../'
}
```

Listing 5.1: Example of JSON file structure for a table

- **tableId**: unique ID to identify the table, (e.g. 10.1 indicating table 1 of article 10).
- **title**: the table's title that was extracted from the *caption* element; if a table does not have a *caption*, the first row with colspan that covers all columns is considered to be the table title.
- **htmlMatrix**: matrix generated after table normalization, splitting span cells where each cell contains an HTML tag from the original table.
- **startRows**: index of row where the table body starts (k).
- **colHeaders**: ordered list of the column headers.
- **rowHeaders**: ordered list of the row headers extracted from the first column of a table.
- **nrows**: number of table body rows.
- **ncols**: number of table columns.
- **tableType**: identifies if a table is `WELL_FORMED`, `ILL_FORMED` or is a table `WITH_INNER_TABLES`. For the remainder of this work, we use only `WELL_FORMED` tables. `ILL_FORMED` tables do not have a well defined table matrix, as the original table could not be normalized. Tables marked as `WITH_INNER_TABLES`,

have multiple tables inside that could not be extracted.

- **articleTitle**: article's title.
- **articleId**: unique number for the article in the corpus.
- **attrs**: HTML attributes of the table.
- **articlePath**: path of the original article file.

In what follows we describe in more detail the process of extracting these metadata from the raw tables.

5.4.1 Header identification

HTML tables have the element `<th>` to represent table headers; nevertheless in some tables these tags may not be present, where we also consider the following criteria to recognize headers that do not use this tag:

- Cells with element `` for text with bold style.
- Cells with element `<abbr>` used to make abbreviations of text.

For extracting table headers, we look over table rows and and get the last row with headers without colspan. We then join headers in all levels to avoid ambiguous information. For example in Figure 5.5 the two row headers are joined getting the structure: `[precipitation**rank@1, precipitation**mm@1, precipitation**in@1, storm**storm@3, location**location@3, ref**ref@1]`, where `**` indicates a sub-header and `@` indicate the type of values in the column (discussed later). We also consider columns without names from tables like Figure 5.6 where the second column has no name; such columns are named `spancol`, and the resulting header structure is `[no@1, spancol@4, position@3, player@3]`.

spancol



No.		Position	Player
1		GK	Andreas Isaksson
2		DF	Stanislav Manolev
3		DF	Wilfred Bouma
4		DF	Marcelo

Figure 5.6: Table with empty header

5.4.2 Text processing

Because we wish to merge as many similar tables as possible, the text header was stemmed to represent it in its basic linguistic form. This process was necessary to

eliminate minor differences between headers, such as those differentiated by singular and plural; for example, taking *writer* and *writers*, after stemming the header for both columns will be *writer*. Common special characters were also eliminated, such as periods and parentheses. For example in Figure 5.7 we extract the column names [*name_of_award*, *award*, *award_as*, *award*] as a result of applying text processing over the table headers. Given that stemming words produces similar headers as per the example, it may also lead to false matches between table headers; for this reason it is necessary to consider other properties to identify better the table columns; in the next section we will describe this process.

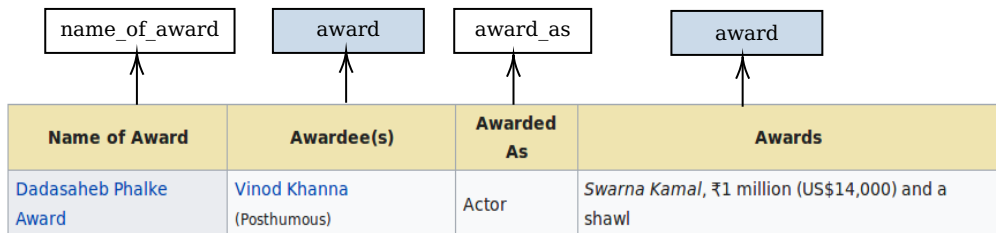


Figure 5.7: Text processing over table headers

To eliminate noise in the dataset we replace number headers with word *numberheader* (e.g., columns named 1,2,3, etc.) and years by *yearheader*. In Table 5.4 we provide a list of the top twenty words in table headers.

Table 5.4: Top 20 stemmed header names in tables

year	376405	name	187896
no	336579	player	187014
titl	293632	posit	168954
parti	274531	result	162588
vote	249535	rank	160743
candid	241385	length	135635
percentag	241086	<i>numberheader</i>	133440
note	227520	pts	120074
team	212292	pos	116233
date	199161	time	104324

5.4.3 Data types

To differentiate tables by the type of content in each column the types: *numeric[1]*, *date[2]* and *string[3]* and *other[4]* (*empty columns*) were used. For example in the table of Figure 5.6 the types corresponding to each header are [1,4,3,3], so the complete header structure is represented as [*no@1, spancol@4, posit@3, player@3*]. This assignment allows to differentiate columns with the same name but different types of content per the example in Figure 5.8, where column headers *teams* and *players*

(being transformed to singular words) have the data types *numeric* instead of *string*, unlike the column *player* in the previous example (Figure 5.6).

Overview of European Universities Bridge Championships





	Location	Countries	Teams	Players	Team			Pairs		
					Winner	Finalist	Bronze Medalist	Winner	Finalist	Bronze Medalist
2009	Opatia  Croatia	11	22	108	Technical University of Wroclaw  Poland	Paris University of France  France	Hogskolen i Sor-Trondelag University  Norway	*	*	*

Figure 5.8: Table with numeric type of columns

Some tables have repeated column names as per the example in Figure 5.9 (after colspan normalization). To avoid incorrect assignments, we numbered the columns with the same name resulting in the header *[event@3, 1_gold@3, 2_gold@1, 1_silver@3, 2_silver@1, 1_bronze@3, 2_bronze@1]*, where the columns can be distinguished by their number and data type; we only assign numbers to repeated columns to allow for matching tables that reorder columns with unique names.


Event	Gold		Silver		Bronze	
Men's singles <i>details</i>	Georg Hackl  Germany	3:21.571	Markus Prock  Austria	3:21.584	Armin Zoggeler  Italy	3:21.833
Women's singles <i>details</i>	Gerda Weissensteiner  Italy	3:15.517	Susi Erdmann  Germany	3:16.276	Andrea Tagwerker  Austria	3:16.652

Figure 5.9: Table with repeated name headers

For extracting data types various patterns were used. For the *Date* data type we used the library *dateutil*², which consider a list of common date patterns³ used on the Web, such as month and day abbreviations, and time zone; additionally we add a pattern to recognize periods of year like *1990-2000*, because there are many tables with this type of content. For *Numeric* data, we consider a numeric pattern extracting special characters such as parenthesis, grades and currency symbols; the remainder of string data was recognized as *String* while the empty text columns were marked as *Other*. Figure 5.10 shows the data type distribution in our tables, extracting the ratio of each column type by table, from which we make the following high-level observations:

- 60% of the tables have numeric columns while 20% of tables have more than 50% numeric columns.
- About 20% of the tables have date columns.

²<https://dateutil.readthedocs.io/en/2.5.0>

³<https://www.w3.org/TR/NOTE-datetime>

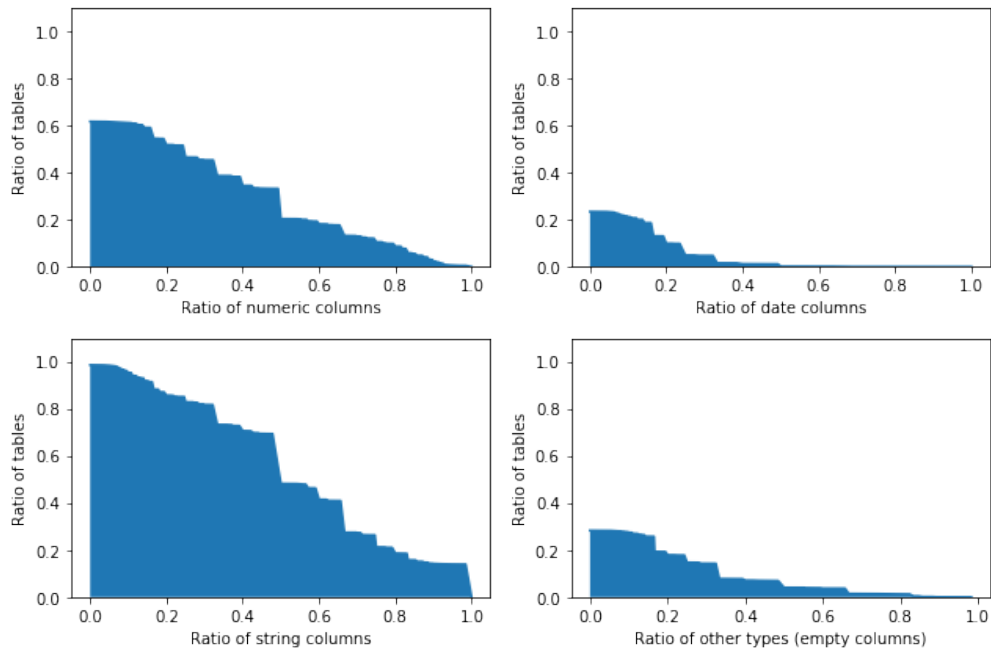


Figure 5.10: Distribution of column data types

- 75% of the tables have at least two string columns.
- About 25% of tables have empty columns.

There is a large percentage of tables with numeric columns, from which we currently do not extract entities; however almost all tables have at least one string column from which we can potentially extract entities, while 75% of tables have at least two string columns, which makes it possible to extract entity relations between them. In fact, we are able to also extract relations from tables with a single string column, as we add a "virtual" column to each table containing the entity of the article. We describe this in the next section.

Table 5.5: Example of a normalized table with article entity

<i>Protagonist article</i>	<i>City</i>	<i>Country</i>	<i>Pop. (2015)</i>
South America	Sao Paulo	Brazil	11,967,825
South America	Rio de Janeiro	Brazil	6,476,631
South America	Mexico City	Mexico	8,918,653
South America	Santiago	Chile	5,507,282

5.5 Article's table entity

The article's name is included in each row of the tables as a new "virtual" column for extracting relations between entities within tables and the entity corresponding to the article's title, as Muñoz et al. [29] propose in their work. This method aims

to find relations, for example, in *attribute-value* tables where the main entity is not within table. In Table 5.5 is an example of a table with the article entity included as a new column. We use a representative index of -1 for the new column and the name "protagonist article" for the column.

Chapter 6

Knowledge Base Querying

In this chapter we describe the process of extracting reference knowledge from Wikidata, which we use to enrich data from tables and later to propose relations between table columns, extracting new candidate triples. We also present some statistics of the data extracted.

6.1 Wikidata Access

Wikidata has an endpoint to query its knowledge base; however since we have a large amount of data extracted from tables for querying, we used a dump file downloaded from the official site ¹ to extract information rather than overloading the endpoint. Wikidata has multiple versions of dump files in different formats. For our purposes we used two versions:

- A Turtle file that contains the entire knowledge graph of Wikidata, from which we extracted the entities.
- An N-triples dump file that contains direct relationships without qualifiers.

A total of 51,488,027 entities and 4,936 properties were extracted from these files. To achieve an efficient querying process, we load the information in memory, building a virtual index to store all the necessary information over which we can perform efficient lookups to find candidate relations and features.

¹<https://dumps.wikimedia.org/wikidatawiki/> - July 30, 2018

6.2 Entity extraction

Most tables in Wikipedia with *String* content have hyperlinks to Wikipedia resources; we use these hyperlinks to identify the Wikidata entities in each table cell. In Section 5.3 we describe table normalization where each table is represented as an $m \times n$ matrix with the table body $B_T(i, j)$ (for $i \leq m - k, j \leq n$). A cell of table body can contain multiple links, which we represent as $W_T(i, j)$ (using the notation applied in Muñoz et al. [29]). For each $w \in W_T(i, j)$ we map the Wikipedia URL (pre-processed replacing special characters and adding the namespace `http://en.wikipedia.org/wiki/` when links are incomplete) as well as the article title to the corresponding entity in Wikidata.

For mapping the URLs, we use the Turtle file that contains triples about the Wikidata ID of each Wikipedia Article, as is shown in Listing 6.1. The entity ID can also be extracted using SPARQL by querying for the Wikipedia Article as per the example in Listing 6.2.

```
<https://en.wikipedia.org/wiki/Santiago> a schema:Article ;
  schema:about wd:Q2887 ;
  schema:inLanguage "en" ;d
  schema:isPartOf <https://en.wikipedia.org/> ;
  schema:name "Santiago"@en .
```

Listing 6.1: Example of triple of Wikidata that has ID and Wikipedia Article

```
PREFIX schema: <http://schema.org/>
SELECT ?id WHERE {
<https://en.wikipedia.org/wiki/Santiago> schema:about ?id.
}
```

Listing 6.2: Example query ID from Wikipedia Article

Additionally it was necessary to query for redirect pages in Wikipedia using the Wikipedia service available for this task (see Listing 6.3). A total of 929,336 new hyperlinks were found. Finally 3,175,548 entities were identified from hyperlinks in table cells. The number of tables with hyperlinks is shown in Table 6.1.

```
https://en.wikipedia.org/w/api.php?action=query
&titles=page1&redirects&format=jsonfm&formatversion=2
```

Listing 6.3: Example redirect Wikipedia Querying

Table 6.1: Tables with hyperlinks

Total Tables	3,631,230	100%
Tables with hyperlinks	3,100,794	85%
Tables with Wikidata entities	3,050,328	84%

There is an average of 5 Wikidata entities per column in the tables, and an average of 5 columns and 13 rows per table with such entities.

Finally we denote by $E_T(i, j)$ the set of Wikidata entities extracted from the corresponding cells of table T .

6.3 Triple extraction

With Wikidata entities now identified we extracted candidate relations by finding existing relations in Wikidata for pairs of entities on the same row of a table.

We used the N-triples file dump for extracting relations between entities; this file contains triples of the form $(subject, predicate, object)$, which was converted to a simple CSV file using the entity and predicate identifiers.

For extracting relations in both directions we add inverse triples $(object, predicate-inv, subject)$, for example $(Chile, capital, Santiago)$ and $(Santiago, capital-1, Chile)$. After sorting the file we can get all (inverses) relations where *Santiago* is involved such as: $(Santiago, capital-1, Chile)$, $(Santiago, region, Region Metropolitana)$ (see an example of extracted triples in Figure 6.1), making it possible to create an index in memory and look up relations by a given subject entity; we discarded triples where either the subject or object entity do not appear in any table, to save space.

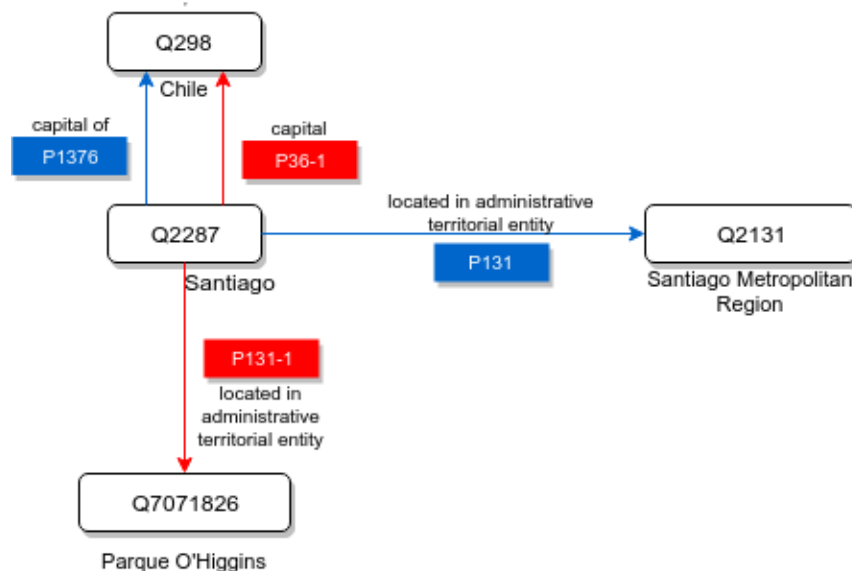


Figure 6.1: Example of inverse relations extracted

6.4 Statistics for features modelling

In addition to triples, we extracted statistics for each predicate. We believe that this information is relevant to determine features that help to describe relations (explained in Chapter 9); for example if there exist multiple objects for a subject and predicate we could add a new object, but we often should not add new objects for a predicate that usually has only one object like *capital*, and where an object already exists. Figure 6.2 shows the multiplicity of predicates *capital* and *located in administrative territorial entity* of Wikidata.

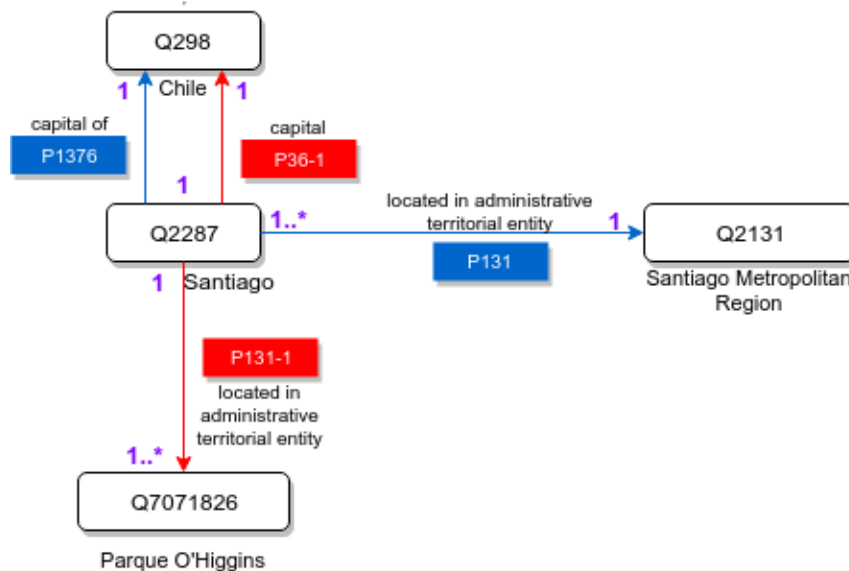


Figure 6.2: Example of predicate multiplicity

On the other hand we also add information about the range and domain of predicates to identify triples that do not comply with this condition. For example in Figure 6.3, is an example of table where a triple proposed with predicate *country* for *sport* is incorrect given the observed class of the object entity.

For extracting such information about entities and predicates we read Wikidata dump files, extract necessary information and build local indexes containing the following data, which later will be used for extracting candidate triples from tables, that will be classified as correct or incorrect by associating features to them.

- **WikidataRedirects:** list of Wikipedia links extracted from tables with their redirect links.
- **WikidataLinks:** list of links extracted from tables and article names with their corresponding Wikidata ID.
- **WikidataRels:** list of entity pairs (Wikidata ID) related with one predicate by line.

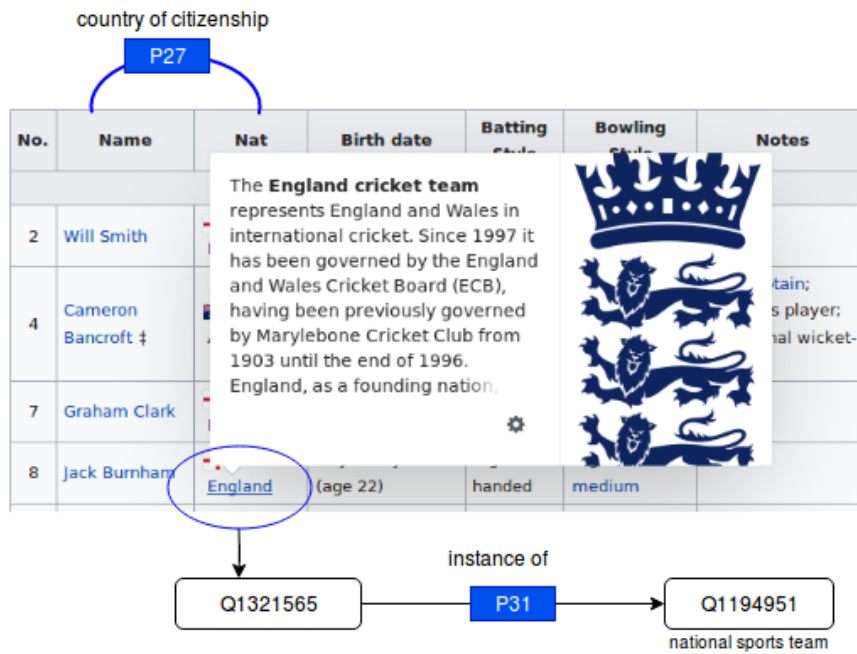


Figure 6.3: Example of object entity not in range of predicate

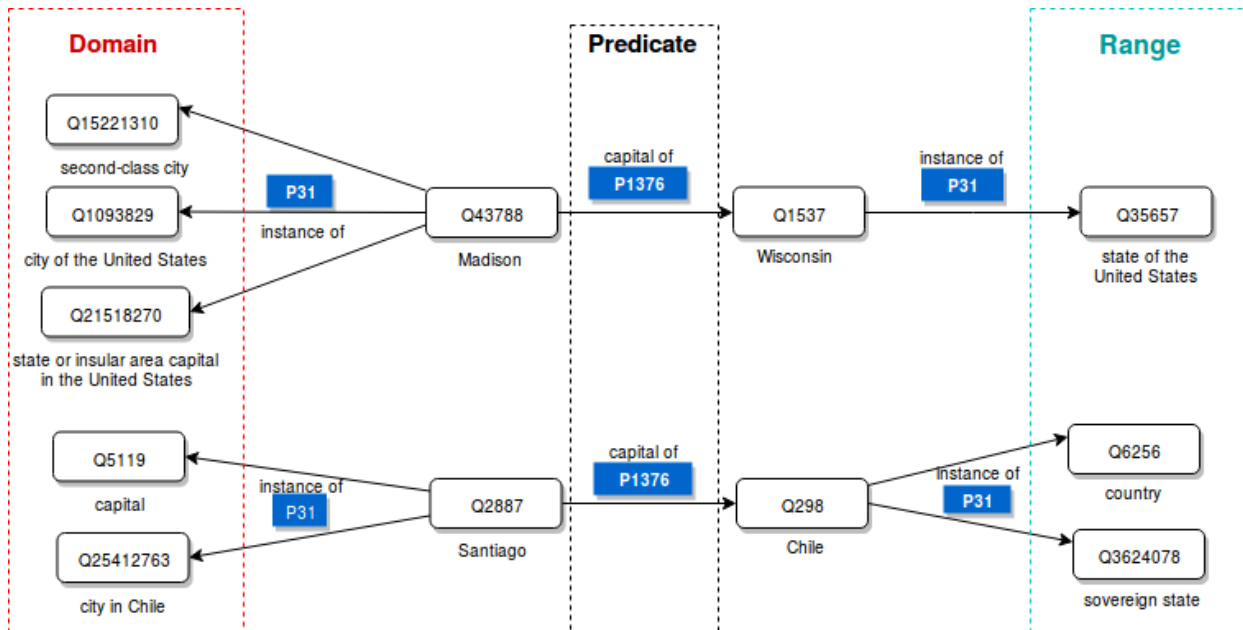


Figure 6.4: Range and domain predicate extraction

- **WikidataEntityClass**: list of entities with their corresponding list of classes. Wikidata uses the predicate P31 that indicates the class of an entity. One entity can belong to multiple classes.
- **WikidataDomain**: list of domain classes by predicate. Wikidata does not offer explicit domain or range definitions so we rather extract this information from the data. Figure 6.4 shows the classes in the domain and range of a predicate; since an entity is related to many classes we consider all classes for subjects

(domain) and objects (range) existing in the knowledge base.

- **WikidataRange**: list of range classes by predicate.
- **WikidataPropStats**: statistics by predicate with the number of unique objects, number of unique subjects, number of triples, maximum number of objects by subject and maximum number of subjects by object.
- **WikidataSubjPredCount**: number of objects by subject and predicate.
- **WikidataObjPredCount**: number of subjects by object and predicate.

In Appendix A there is a summary of predicates and entity classes that were extracted from tables.

Chapter 7

Grouping tables

This chapter contains an overview of the tables extracted from Wikipedia, using some clustering algorithms to explore the distribution of the data and gain initial insights about existing groups of tables.

7.1 Clustering overview

Related works [7, 3, 48, 10, 21] present different ways of grouping and joining tables. The method to be used depend on the general objective; for example for extracting relevant information about a certain topic, which involves searching and retrieving similar tables based on a specific query, the search may be implemented based on a similarity measure for recovering the most similar tables based on matching the specific query to table content. On the other hand, for extracting information at large scale, as is our goal, the grouping task is independent of a particular query, and rather requires scalable methods for partitioning the entire corpus of tables into groups; a natural place to start is to consider clustering methods, where we present exploratory results for applying standard clustering methods over our tables.

We used table headers to generate clusters over 10,000 tables extracted randomly, considering the hypothesis that tables with same column names will contain similar entities, for which we vectorize the table headers using the term frequency in the dataset.

We consider that column table headers are a set of words (without a specific order e.g: [country, city, name, elevation] and [name, city, country, elevation]), and use the Jaccard distance which measures the similarity between two sets: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

In the following we present some results of applying clustering algorithms over the sample data.

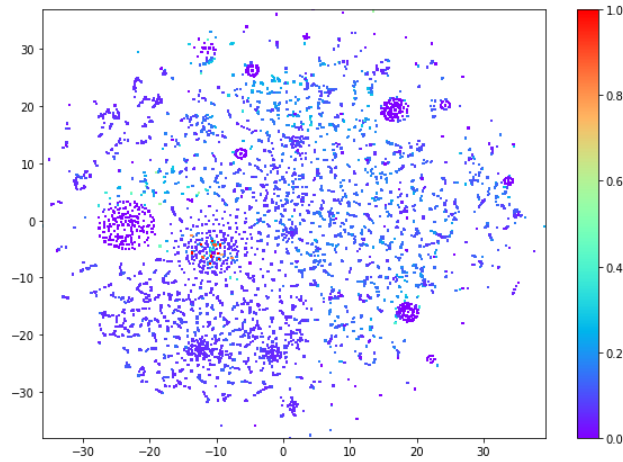


Figure 7.1: Cluster visualization of sample test using T-SNE algorithm (the color represents the normalized distance from each point to the closest points)

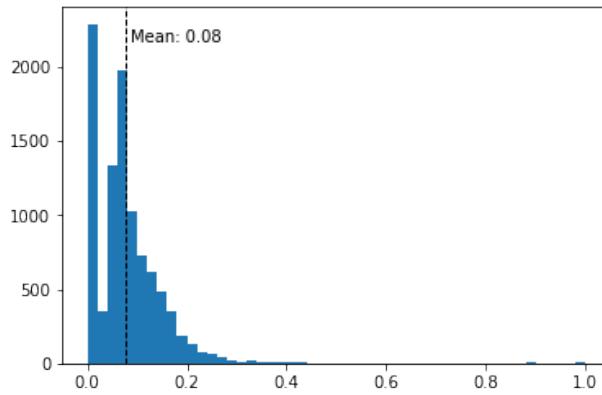
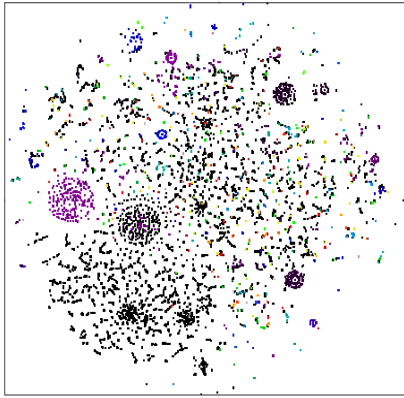
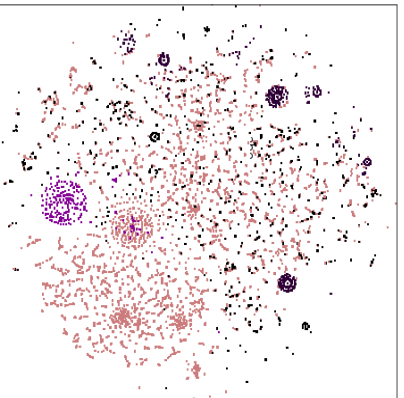
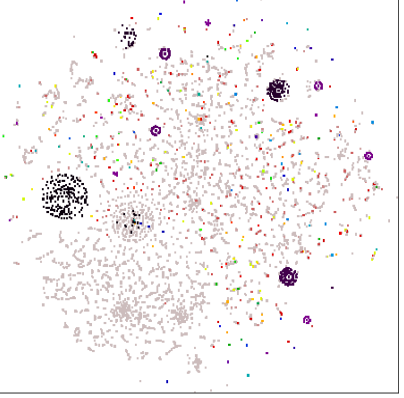
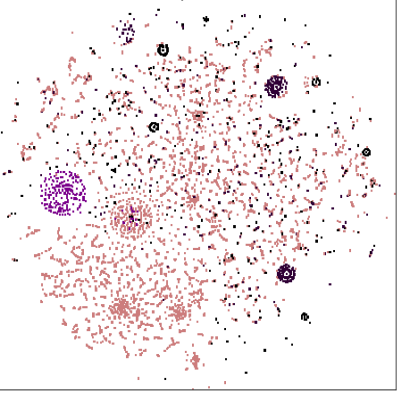
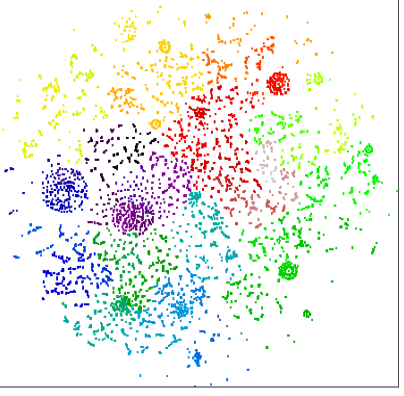
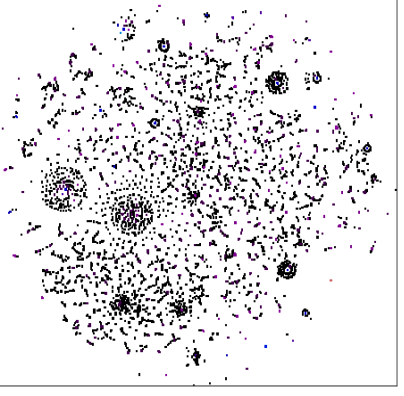


Figure 7.2: Mean of jaccard distances sample set

Figure 7.1 shows a cluster representation using the T-SNE algorithm (T-Distributed Stochastic Neighbour Embedding), which makes a non-linear dimensionality reduction preserving the local distances from each point to their neighbours. Its goal is to minimize the difference between the normal distribution of points in high-dimensional space to the distribution when projected to 2-dimensional space. The algorithm uses a parameter called perplexity that is roughly equivalent to the number of the nearest neighbours considered for matching the two distributions, and the learning rate for the gradient descent process applied to find the best fitting. After some iterations, we consider a perplexity and learning rate of 10; using a larger number for the model improves learning speed but the visualization does not change significantly. Figure 7.1 shows small dense clusters with Euclidean distance of 0, which reflects the presence of identical sets of headers and a large quantity of headers with about 0.5 distance. The mean distance to the 10 nearest neighbours is 0.08, as is shown in Figure 7.2. More than 50% of tables in the sample set have at least one table with the same header; for this reason the mean distance of all points is small. If we apply DBSCAN anecdotally over the sample data set considering a maximum distance of 0.2, and 2 minimum

Table 7.1: Cluster visualizations of sample set of 10000 tables

Algorithm	2D visualization of clusters	2D visualization of clusters size
DBSCAN		
HDBSCAN		
HIERARCHICAL COMPLETE LINKAGE		

points by cluster, we get 582 clusters. Despite the fact that DBSCAN groups tables that share similar headers (e.g. country, capital, population and country, city, population), the algorithm clusters data points of less dense areas, due to the *min_points* parameter selected; if we increase this parameter, we can avoid some noise, but skip small clusters. The density variation algorithm HDBSCAN resulted in 167 clusters; the main difference between both algorithms is that the second one considers points in less dense space as clusters while DBSCAN considers them as noise and groups those points with similar density. We also used hierarchical clustering, building a tree

with maximum distance of 0.2; the result was 2,842 clusters.

Table 7.1 shows the visualization of clustering using these three algorithms, considering Jaccard distance. Our exploratory analysis here raises three problems using standard clustering methods. First we note that different clustering methods give different results depending on the parameters configuration required for each one, where it is not clear how we should configure the clustering algorithm, since we are not focused on performing the algorithms for specific groups. Second, these clustering methods tend to group by overlapping terms in table headers, but this does not necessarily correspond to what we want, which is to group table for which the same knowledge-base relations hold between pairs of columns; for example two tables with [*country, city, elevation*] and [*artist, city, country*] have a Jaccard distance of 0.5, but we do not want to merge these tables. Third, there is the issue of scale, where we perform clustering experiments on a small fraction of our overall corpus, but scaling these methods for the full corpus has a high cost.

7.2 Grouping tables proposal

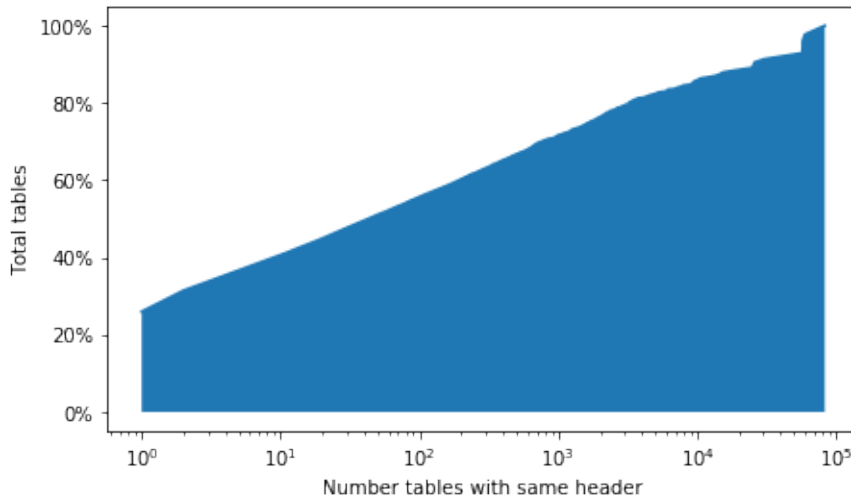


Figure 7.3: Distribution of tables with same headers

In the previous section we show different clustering methods for grouping tables using the headers; this overview suggests that the tables in our Wikipedia corpus do indeed form natural clusters which may be due to the use of standard templates that automatically generate tables with a particular structure for a particular purpose (e.g. the climate of a city) or due to users manually copying and pasting similar tables from related articles and changing the content. Such results are encouraging for this work as we expect clusters to capture large groups of tables for which the same relations exist between pairs of columns. Also we observe that it is possible to group similar tables considering a distance measure; however, such methods are not directly applicable

for our use-case; for this reason we propose to group tables based on having the same (normalized) headers. We choose this conservative form of grouping tables to avoid introducing noise to the relation extraction process, for which not merging relevant tables will have less (negative) effect than merging two unrelated tables for which different relations are applicable, allowing us validate our method over the whole corpus of tables without filtering those that do not fit in any cluster if we apply other clustering methods. We thus consider to sort the headers of the table to group tables with headers in different order but with similar information.

Grouping tables by their header, we achieve a total of 1,135,977 groups; this large number is due to tables with unique headers. Figure 7.3 shows the number of tables by group; we can see that 30% of the tables have unique headers, and conversely about 30% of the tables are in groups formed with more than 1000 tables. Within these groups, there may still be tables that happen to have the same header but that are unrelated, meaning that different relations hold between their columns. In the next chapter, we will propose a heuristic method to detect and repair such cases based on the relations extracted for individual tables, but first we present a initial evaluation of groups of tables with the proposed method.

7.3 Evaluation of table groups

We extracted the top 10 groups of tables extracted from grouping tables with the same header, to evaluate how similar in structure and context are the tables in each group. A total of 250 pairs of tables were randomly extracted from each group. These pairs of tables were then evaluated manually to see if they should be merged (yes), should not be merged (no), or were partly similar, meaning that the structure is similar but not exactly the same, as per the example in Figure 7.4. The results of this evaluation are shown in Figure 7.5. From the 10 groups evaluated, one of them contains tables that are partly similar, since we excluded the name of empty table columns in the dataset; the difference in this group is because it contains tables with an empty column and others without it (see Figure 7.4), nevertheless this variation does not change the meaning and context of the tables inside the evaluated group.

Heath Town				
Party	Candidate	Votes	%	±
Labour	Caroline Siarzewicz	2122		
Conservative	John Lee	1242		
Liberal Democrat	Roger Gray	675		
Majority		880		
Turnout		4037	54.17	
Labour hold		Swing		

California's 1st congressional district election, 1990			
Party	Candidate	Votes	%
Republican	Frank Riggs	99,782	43.33
Democratic	Douglas H. Bosco (incumbent)	96,468	41.90
Peace and Freedom	Darlene G. Comingore	34,011	14.77
Total votes		230,261	100.00
Turnout			
Republican gain from Democratic			

Figure 7.4: Example tables from group 1

With this manual evaluation we have more certainty that the proposed relations (at least) for the larger groups of tables are in the same context, however there is still

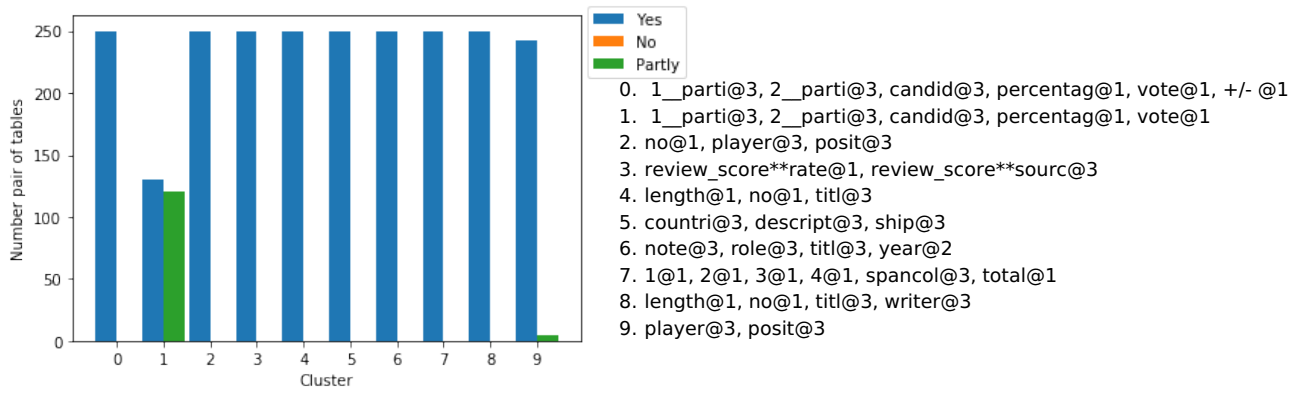


Figure 7.5: Evaluation of pair of tables from top 10 groups

the possibility of finding tables with same headers but with different information, for example in tables with not very specific headers like *[name, country]*, where name can contain different types of entities. Rather than use the classes of entities from each table, we explore another alternative using the candidate triples. In next chapter we present the candidate triples extracted from individual tables and also from the group of tables that are merged using the vertical merging method.

Chapter 8

Relation extraction

As we propose to compare two methods of extracting relations with respect to Wiki-data: from individual tables based on Muñoz et al.' [29] approach and from merged tables, in the present chapter we present the candidate triples extracted using both methods.

8.1 Candidate triples from individual tables

For extracting the candidate triples from individual tables we iterate over the $m \times n$ body $B_T(i, j)$ of each table matrix (for $i \leq m, j \leq n$), reading the entity pairs $\{(e_{i,j}, e_{i,k}) \mid e_{i,j} \in E_T(i, j), e_{i,k} \in E_T(i, k), i \leq m, j < k \leq n\}$, where any relations found in $(e_{i,j}, e_{i,k})$, are added to the pair of corresponding columns $(h_j, h_k) = (H_T(i), H_T(k))$; after extracting the candidate relations from each entity pair, we iterate over the table generating candidate triples with the suggested relations for each (h_j, h_k) . The article's entity is added as a virtual column in the table (e.g. ha in Figure 8.1); in this way candidate triples between entities within tables and the article's entity can be also extracted.

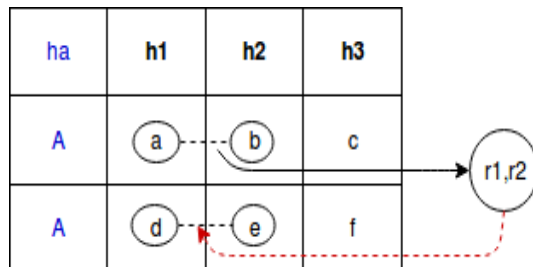


Figure 8.1: Example of candidate triples extraction from individual tables

In Figure 8.1 we observe that relations **r1** and **r2** extracted from the triples between entities **(a,b)** can be proposed for the pair of entities **(d,e)** generating two novel candidate triples: **(d,r1,e)** and **(d,r2,e)**.

About 14 million existing triples were extracted from Wikidata and a further 62 million candidate triples can be proposed considering these existing relations. Next we describe the method used for extracting candidate triples by merging tables.

8.2 Candidate triples by merging tables

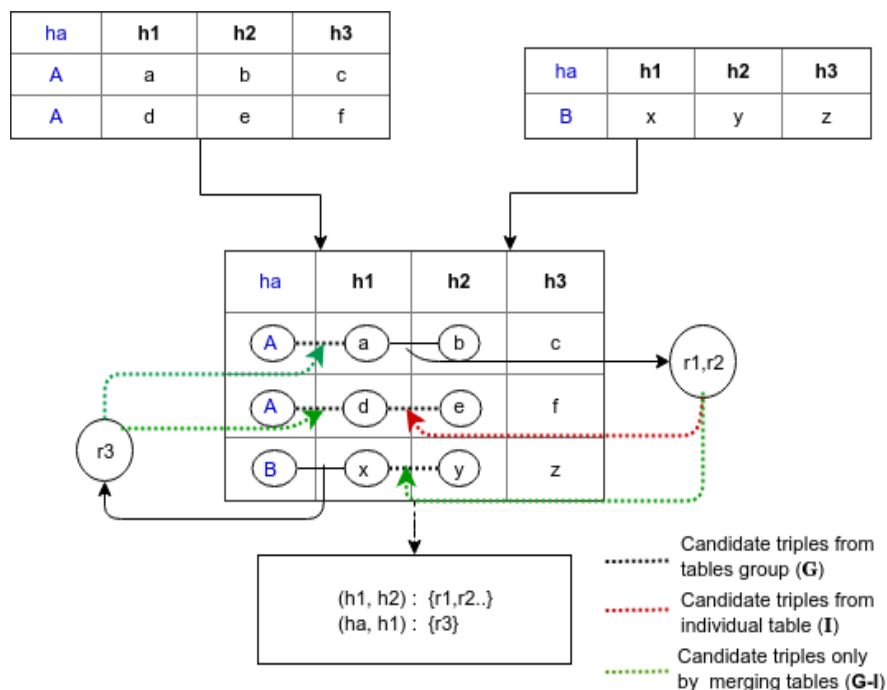


Figure 8.2: Example candidate triples extraction by merging tables

We apply vertical merging to join table rows generating a virtual merged table (keeping the merged table in memory) from each group of tables with same header. We illustrate the relation extraction process with an example in Figure 8.2; here we identify two types of candidate triples: the candidate triples from the same table (red line), for example triples with relations $r1$ and $r2$ for the entity pair (d, e) ; and the candidate triples by merging both tables (green line), resulting in the relations proposed for (x, y) and the relation $r3$ for the entity pairs (A, a) and (A, d) ; we denote the set of candidate triples that can be extracted without merging tables as I and the new triples that only would be achieved by grouping and merging tables, which we denote as $G - I$, since G includes all triples.

The relation extraction process allows us to propose relations for entities in tables that do not have any one, or add new relations for entity pairs. For instance, considering the two tables in Figure 8.3, we add the relations `residence` for the entity pairs of rows 2, 3, 4 and also add this relation for the entity pairs of the second table; likewise the relation `placeOfBirth` is added to entity pairs of the first table. Since both relations are used to create triples from tables where no such relation previously existed,

	Country	Position	Name
①	Mexico	FW	Abraham Carreño
②	Mexico	MF	Jesús Manuel Corona
③	Argentina	DF	José María Basanta
④	Mexico	DF	Hiram Mier

	Country	Position	Name
⑤	Portugal	DF	Paulo Ferreira
⑥	Germany	MF	Marko Marin
⑦	England	DF	Ashley Cole
⑧	Spain	DF	César Azpilicueta

①	residence (Abraham Carreño, Mexico)	⑤	placeOfBirth(Paulo Ferreira, Portugal)
		⑥	placeOfBirth(Marko Marin, Germany)
		⑦	placeOfBirth (Ashley Cole, England)
		⑧	placeOfBirth (César Azpilicueta, Spain)

Figure 8.3: Example of existing triples from two tables indicating the source rows

Table 8.1: Example of candidate triples extracted based on Figure 8.3

	Relation	Source
②	residence (Jesús Manuel Corona, Mexico)	<i>I</i>
③	residence (José María Basanta, Argentina)	<i>I</i>
④	residence (Hiram Mier, Mexico)	<i>I</i>
①	placeOfBirth (Abraham Carreño, Mexico)	<i>G – I</i>
②	placeOfBirth (Jesús Manuel Corona, Mexico)	<i>G – I</i>
③	placeOfBirth (José María Basanta, Argentina)	<i>G – I</i>
④	placeOfBirth (Hiram Mier, Mexico)	<i>G – I</i>
⑤	residence (Paulo Ferreira, Portugal)	<i>G – I</i>
⑥	residence (Marko Marin, Germany)	<i>G – I</i>
⑦	residence (Ashley Cole, England)	<i>G – I</i>
⑧	residence (César Azpilicueta, Spain)	<i>G – I</i>

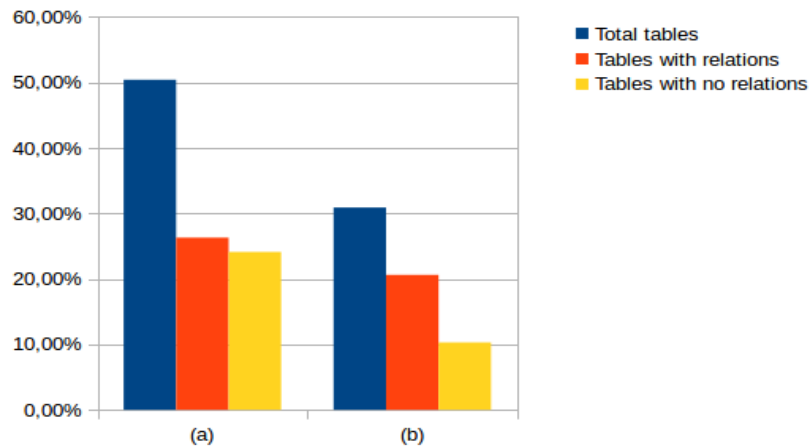


Figure 8.4: Number of tables by group (a) including article's entity relations and (b) with no article's entity relations

we marked the triples as "triples from group". The results of triples extraction are shown in Table 8.1.

With this approach we reached 398 million candidate triples that can be extracted only by merging tables. These triples result from proposing relations extracted from about 26.3% of the 3,631,230 individual tables, generating new triples for about 24.1% of the tables that contain no relation, including triples involving the article's entity and 10% of tables with no article entity relations. Finally it is possible to propose relations for 1,830,189 tables (50%), as shown in Figure 8.4.

The next step is to identify which of the extracted relations are correct, since as we see in Table 8.1 the relation residence may not be correct for entities in the second table; thus we propose to evaluate triples from both groups: those extracted from the same table (I) and those resulting only by merging tables ($G - I$).

8.3 Evaluation of candidate triples

The precision of candidate triples was evaluated initially considering the number of correct triples that we can achieve with this method.

From the 62 million candidate triples extracted from individual tables and 398 million candidate triples extracted by merging tables, we randomly selected 100 triples from each group and classified them as correct or incorrect; we also consider to add two classes: *contextual* and *unknown*. We describe these labels following:

- **Correct:** indicates that a triple is semantically correct following the use of the predicate in the Wikidata knowledge base and being applicable for the given subject and object. An example of this class is the triple extracted from a table with the columns [*artist@3*, *song@3*]: (Just_Like_Fire :Q23902348, P76:lyrics by, Pink_(singer):Q160009).
- **Incorrect:** indicates that the triple is not correct per the previous definition. For example from a table with columns [*citi@3*, *country@3*] the triple (United_States:Q30, P36:capital, Burlington,_Vermont:Q31058) is incorrect.
- **Contextual:** the triple was correct at some point but not currently. For example given a table with headers [*country@3*, *represent@3*] the triple (United_States:Q30, head_of_state:P35, Barack_Obama:Q76) is contextual.
- **Unknown:** a triple for which the judge lacks the required knowledge and information for classifying it as correct or incorrect; this may include highly subjective cases. For example a table with columns [*actor@3*, *film@3*] the triple (Mos_Def:Q38875, P800:notable work, The_Italian_Job:Q1051032).

The triples were classified manually through a web application that provides links to

Wikipedia articles and Wikidata entities to facilitate the annotations (see Figure 8.5). For example, from the table in Figure 8.6, we extracted the triples shown in Table 8.2, one of which is classified as correct and the other one as incorrect.

Triples Evaluation Logout

Choose list of triples.

Correct

Number: 108

Id	Subject	Predicate	Object	Option
32204	Melani_Costa Q983177 name@3	P19:place of birth	Spain Q29 nation@3	<input checked="" type="radio"/> Correct <input type="radio"/> Incorrect <input type="radio"/> Contextual <input type="radio"/> Unknown

826469.3 [Swimming at the 2014 European Aquatics Championships – Women's 400 metre freestyle](#)

Rank	Heat	Lane	Name	Nationality	Time	Notes
1	2	5	Federica Pellegrini	Italy	4:07.09	Q
2	2	4	Mireia Belmonte	Spain	4:07.12	Q
3	3	7	Sharon van Rouwendaal	Netherlands	4:07.59	Q
4	3	5	Jazmin Carlin	Great Britain	4:08.17	Q
5	3	6	Boglárka Kapás	Hungary	4:09.17	Q
6	3	4	Melani Costa	Spain	4:09.59	Q
7	3	3	Lotte Frils	Denmark	4:09.75	Q

Figure 8.5: Application used for triples annotation

Year	Song	Artist	Album	Label	Performed As
1995	Graveyard	Project Born	Born Dead EP		
1997	\$85 Bucks An Hour	Twiztid	Mostasteless	Psychopathic Records	
	Spin The Bottle				
	Meat Cleaver (w/Myzery)				
	Hound Dogs (w/Blaze Ya Dead Homie)				Dark Lotus

Figure 8.6: Example table from which triples in table 8.2 where extracted

Table 8.2: Example of triple classification

Table	Subject	Predicate	Object	Label
500871.10	Mostasteless :Q778240	record label :P264	Psychopathic_Records :Q2116093	Correct
500871.10	Myzery :Q6949375	lyrics by :P676	Twiztid :Q1849210	Incorrect

Table 8.3: Initial labeling of sample triples from I and $G - I$

Triples	Annotators	Correct	Incorrect	Contextual	Unknown
I	A1	11	73	3	3
	A2	18	68	0	4
Agreement	73%	10	63	0	0
$G - I$	A1	5	93	0	2
	A2	10	86	1	3
Agreement	88%	4	84	0	1

Table 8.3 shows the results of labeling the sample set of triples, where we initially achieved about 80% of agreement in both sets of triples. According to this exercise we realize that some triples, where we do not achieve a total agreement, are mostly because of the use of predicates and also of lack of knowledge. For example the triple (Copernicus:Q619, P17:country, Poland:Q36) may be correct, however the predicate P17:country should be used only for objects and places or events; another example is the triple (Q19444:Birmingham_City_FC, P5138:season of club or team, Q3996006:2009-10_Tottenham_F.C._season) here the correct subject should be Tottenham_Football_Club or the correct predicate P1923:participating team. After reviewing the cases with different labels finally we agree on 23 correct triples from I and 10 correct triples from $G - I$, leaving 73 and 86 incorrect triples from each group respectively. The difference (2%) was considered contextual and unknown.

Best Feature Film	Best Director
<ul style="list-style-type: none"> • Lovely Man <ul style="list-style-type: none"> • Arisan! 2 • Cita-Citaku Setinggi Tanah • Mata Tertutup • Postcards from the Zoo • The Raid • Rayya, Cahaya di Atas Cahaya • Tanah Surga... Katanya 	<ul style="list-style-type: none"> • Teddy Soeriaatmadja – Lovely Man <ul style="list-style-type: none"> • Eugene Panji – Cita-Citaku Setinggi Tanah • Garin Nugroho – Mata Tertutup • Gareth Evans – The Raid • Viva Westi – Rayya, Cahaya di Atas Cahaya

Figure 8.7: Table with multiple entity cells

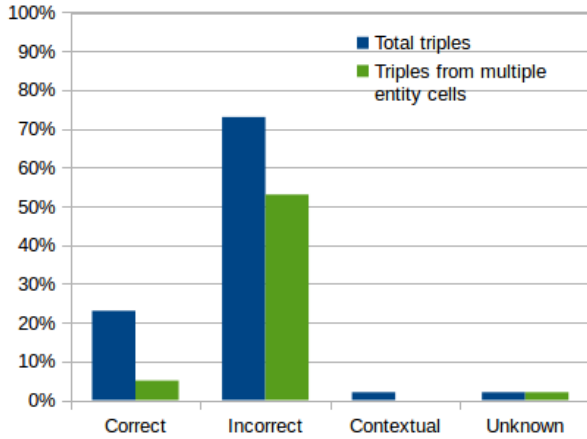


Figure 8.8: Triples generated from tables (I) with multiple entity cells

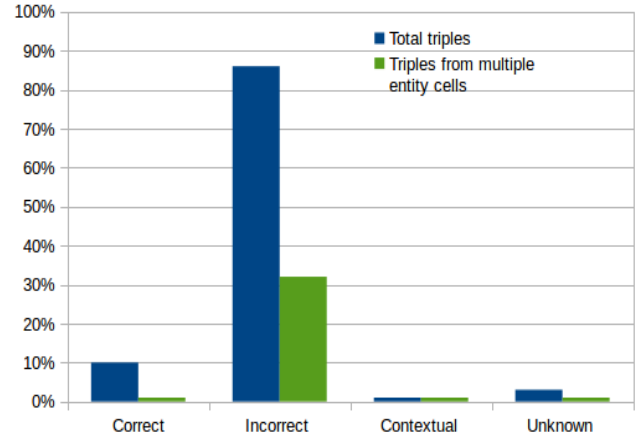


Figure 8.9: Triples generated by merging tables ($G - I$) with multiple entity cells

The results show that the number of incorrect candidate triples increases in the case of merged tables, which is to be expected as we extract candidate triples for all rows if any row witnesses a given relation; larger tables thus lead to more candidate triples and more noise. In absolute terms, however, merging tables does lead to more correct triples extracted as candidates. Whether or not the information extraction process benefits from merging tables then depends on the next phase, which attempts to automatically classify the large amount of candidate triples. Nevertheless the low number of correct triples was deemed a risk for training a good model, where we thus looked for a way to filter (mostly) incorrect triples.

In the initial annotation process we observed that most of the triples that come from cells with more than one entity are incorrect as per Figure 8.7 where the column *Best Director* contains cells with more than one entity and we can not propose correct relations with entities from the first column. From the total of sample triples, 59 triples from I are from cells with more than one entity, of which 53 are incorrect; while 35 triples from $G - I$ come from this type of cells, of which 32 are incorrect. Based on this observation we decide to delete triples that come from multiple entity cells.

After applying the filter, 21 million candidate triples from I and 145 million candidate triples from $G - I$ were extracted respectively; and considering that cells with multiple entities also produce multiple candidate triples for rows with one subject and object entities, we do not consider these candidate triples. Finally 18 and 134 million of triples were extracted from each set.

8.4 Dividing table groups

Additionally to solve the issue where unrelated tables can be erroneously grouped producing incorrect triples we explore the option of dividing groups that may have un-

related tables, splitting those whose tables contain columns with conflicting relations; for example two tables with headers $[name, country]$ may be conflicting as per Figure 8.10, where the first one has the relation P276:location between rivers and country and the second one P1532:country for sport. Given that pairs of columns may have more than one relation, we rank the relations by frequency and consider only the most frequent.

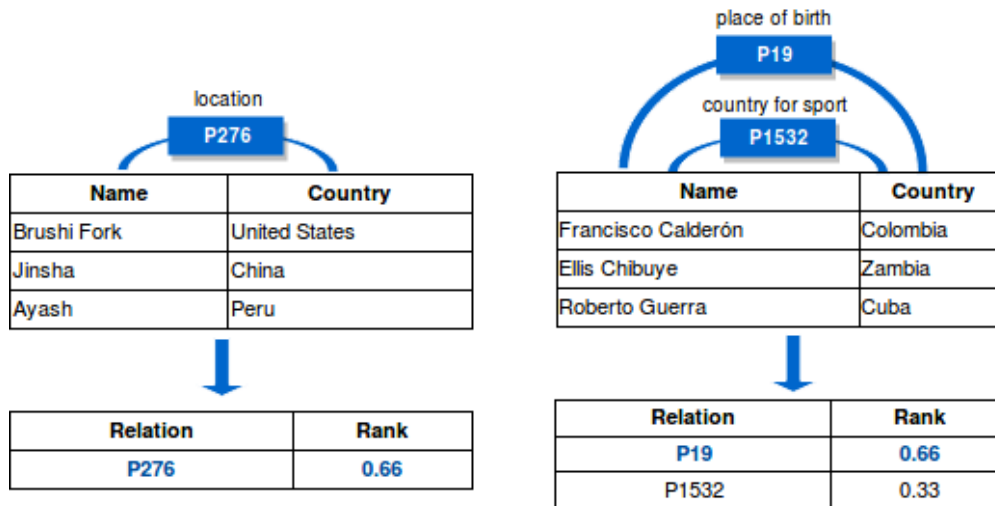


Figure 8.10: Tables with conflicting relations

We define the conflict of two relations in the knowledge base by the number of entity pairs (a, b) in the corresponding columns that have two different relations r_1 and r_2 . If the two relations have multiple common entity pairs the conflict is lower than in the other case. For example if the predicate P19:place of Birth and P1532:country for sport have multiple entity pairs in common (e.g. $\{(Lionel_Messi, Argentina), (Diego_Maradona, Argentina)\}$) then these predicates have a high correlation and there is no conflict between them. We consider a threshold of 0.5 as the ratio of the number entity pairs (a, b) with both predicates (e.g. $r_1:P276$ and $r_2:P1532$) and the number of existing triples with the predicate in the knowledge base, as following.

$$correlation = max\left(\frac{|r_1, r_2(a, b)|}{|r_1(a, b)|}, \frac{|r_1, r_2(a, b)|}{|r_2(a, b)|}\right)$$

Where $|r_1, r_2(a, b)|$ is the number of entity pairs that have both relations r_1 and r_2 , while $|r_1(a, b)|$ is the number of entity pairs that have the relation r_1 and $|r_2(a, b)|$ the number of entity pairs that have the relation r_2 ; since one of the relations may be more popular in the knowledge base we take the maximum of both ratios. If the correlation obtained is less than the threshold it means that there are few pairs of entities that have both relations; in this case they will be considered in conflict and tables with those relations between corresponding columns should not be merged.

With this conflict estimation we split groups of tables achieving 33,705 more groups than the previous division; in Figure 8.11 we see that larger groups (more than 10000 tables) are reduced.

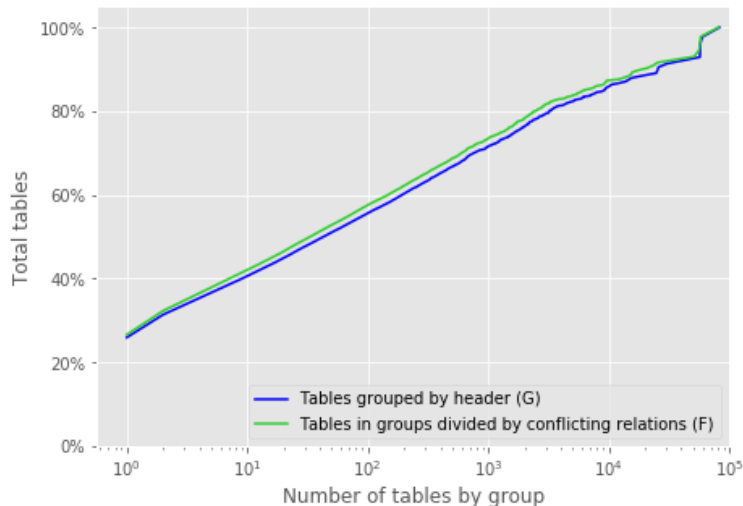


Figure 8.11: Number of tables by group, conflicts analysis

By splitting these groups of tables with incompatible relations, we attempt to reduce the number of incorrect triples, produced by merging tables. We believe that by increasing the ratio of correct triples the classification methods could achieve better precision considering the unbalanced condition of the dataset, thus we denote the new set of triples from split groups as F . To evaluate the number of correct triples that we can achieve from each set we annotated anew 100 triples randomly selected from each group, this time without considering relations from multiple entity cells. In Table 8.4 are shown the results of annotation made by two annotators with the observed agreement.

Table 8.4: Annotation agreement

	Triples	Correct	Incorrect	Contextual	Unknown
I	A1	37	57	4	2
	A2	38	59	2	1
	Agreement 84%	31	51	2	0
$G - I$	A1	5	91	-	4
	A2	5	91	-	4
	Agreement 91%	4	87		0
$F - I$	A1	11	81	4	4
	A2	9	85	2	4
	Agreement 84%	6	77	1	0

In Appendix B we present the inter-rater agreement achieved for each group of triples. We calculate the *Kappa Cohen* metric achieving 69% for triples from I , 47% for triples from $F - I$ and 46% for triples from $G - I$ and interpret these values according to Landis et al. [24] where they define an agreement in range 41%–60% as moderate

agreement and a value in range 61%–80% as substantial agreement. Although we did not achieve a substantial agreement in the triples extracted by merging tables we consider that moderate agreement is acceptable to continue evaluating the triples using classification algorithms, since the total set of triples generated by merging tables includes I .

By splitting the initial groups of tables we obtained about 10% fewer incorrect triples in the evaluated test set increasing the ratio of correct triples; although this is not a significant variation, since we want to reduce the quantity of incorrect triples in our dataset for the classification process, we will use the dataset splitting the initial groups of tables, even when this task implies that we will omit some correct triples.

Grouping and merging tables should allow us to increase the number of novel triples that may be obtained from individual tables; however an issue we have to deal now is the ratio of correct triples for training a good algorithm and classifying the extracted triples. In next chapter we will present how we address this issue and the results achieved.

Chapter 9

Triple classification

In the present chapter we elaborate on the use of machine learning models to classify candidate triples as correct and incorrect using the features proposed in the baseline work [29] and compare the results with our approach with table merging. Based on experiments with various models and configurations, we choose one model, it is applied over the entire dataset to get the total number of correct triples that we can achieve by applying the baseline and our approach. We describe the dataset, features and algorithms that will be used for classification.

9.1 Features description

We extracted the features associated with candidate triples proposed by Muñoz et al. [29], categorized in 5 groups, which are shown in Table 9.1. Given that many features are self explanatory, rather than discuss each feature individually, we now discuss features by group, providing details only for selected features whose definition might not otherwise be clear. We aim to keep the features precisely as defined by Muñoz et al. [29].

- **Table features:** number of rows and columns and total number of relations extracted from the table. Triples extracted from the same table will have the same value for this features; however the remaining features will be different for each triple.
- **Column features:** features with the count of entities in subject and object columns and the relations extracted by row for each entity pair.
 - *Feature 15/16:* (Unique) Potential Relations counts the number of entity pairs in the table for which a relation could hold (i.e., entity pairs (a,b) from two columns in the same row). Given that the same pair could appear in multiple rows, feature 16 provides a unique count.
- **Predicate features:** features related to the existing information in the knowl-

Table 9.1: Features for classification in Muñoz [30] approach

#	Table Features
1	Num. of tables in article
2	Table id in article
3	Num. of rows
4	Num. of columns
5	Ratio (3)/(4)
6	Total relations extracted
Column Features	
7	Subject column index
8	Object column index
9 & 10	Num. entities in subject and object column
11	Ratio (9)/(10)
12 & 13	Num. unique entities in subject and object column
14	Ratio (12)/(13)
15	Potential relations
16	Unique potential relations
Predicate Features	
17	Normalized unique subject count
18	Normalized unique objects count
19	Normalized triples count
20	Ratio (18)/(19)
Cell Features	
21 & 22	Num. entities in subject and object cell
23	Ratio (21)/(22)
24 & 25	String length in subject and object cell
26 & 27	Formatting present in subject and object cell
28	String similarity for predicate and subject header
29	String similarity for predicate and object header
30	Maximum between 28 and 29
Predicate/Column Features	
31	Num. of rows where predicate holds
32	Ratio (31)/(3)
33	Num. of potential relations where predicate holds
34	Ratio (33)/(15)
35	Num. of unique potential relations where predicate holds
36	Ratio (35)/(16)
Triple Features	
37	From article or body relation (if the relation extracted corresponds to article's entity)
38	Exists in Wikidata

edge base for the predicate of the candidate triple; we describe these features in

more detail in the following.

- *Feature 17: Normalized unique subject*, number of unique subjects for the predicate divided by the maximum number of unique subjects for a predicate in the knowledge base.
- *Feature 18: Normalized unique object*, number of unique objects for the predicate divided by the maximum number of unique objects for a predicate in the knowledge base.
- *Feature 19: Normalized number of triples*, number of triples for the predicate divided by the maximum number of triples for a predicate in the knowledge base.
- **Cell features**: features related to the content of the subject and object cells.
 - *Features 21/22*: The number of entities by cell. These features have the value of 1 after the filter where triples with more than one entity by cell were removed, due to the analysis made in Chapter 7.
 - *Features 24/25, 26/27*: The length of content in the pair of cells from which entities were extracted, and a Boolean value to evaluate the presence of formatted content such as bullets or bold text.
 - *Features 28/29/30*: String similarity between the predicate and subject/object column header, computed using two metrics: one based on characters (Jaro Winkler) and the other based on terms (Dice Score). For this feature the stemmed text from column names and predicates from Wikidata were used (e.g. *locat* for the predicate *location*).
- **Predicate / Column features**: are the features related to the frequency of a predicate by entity pairs (a, b) in the rows and in the total number of relations by a column pair, named as the number of potential relations where the predicate holds.
- **Triple features**: feature 37 is a Boolean feature that describe if the triple is extracted using the article entity (1) or not (0), while feature 38 indicates if the triple already exists in Wikidata; we filter triples that already exist in Wikidata; then this feature will have value of 0, for training and test data.

9.1.1 Adding new features

In this work we propose to add new features related to the table cells, predicates and features resulting from merging tables, which are presented in Table 9.2. We attempt to improve the results of classifying triples with these new features; the results will be presented in the next sections.

We discuss the new features added by category according to Table 9.2:

- **Cell features**: we include features related to the length of additional text in subject and object cell; there are two features (24,25) about the content cell length

Table 9.2: New proposed features

Cell features	
41 & 42	Length of additional text in subject and object cell (without links)
43 & 44	Number of links in subject and object cell
45 & 46	Exist colspan or rowspan in subject and object cell (Values: 1,0)
Predicate features	
39	Object entity is in range of predicate (Values: 1,0,-1)
40	Subject entity is in domain of predicate (Values: 1,0,-1)
47	Maximum number of subjects by predicate
48	Maximum number of objects by predicate
49	Num. of objects by the subject and predicate
50	Num. of subjects by the object and predicate
64	Inverse predicate
Tables group features	
51	Num. of entities in subject column by tables group
52	Num. of entities in object column by tables group
53	Num. of unique entities in subject column by tables group
54	Num. of unique entities in object column by tables group
55	Num. of potential relations in tables group
56	Num. of unique potential relations in tables group
57	Num. of potential relations where predicate holds in tables group
58	Num. of unique potential relations where predicate holds in tables group
59	Num. of rows where predicate holds in tables group
60	Num. of rows in tables group
61	Ratio (58)/(56)
62	Ratio (59)/(60)
63	Ratio (57)/(55)
65	Num. tables by group

in the baseline work, however it includes the size of links, hence we consider to use the length of additional text that are not links; furthermore we add Boolean features that indicate if a triple was extracted from cells with colspan or rowspan.

- **Predicate features:** we add features capturing information about of the predicates in the knowledge base; as we discuss previously we decided to add features that indicate if the subject and object entities of the triple are in the domain and range of predicate respectively (value -1 if the respective domain/range classes do not appear in the knowledge base) and also the maximum number of existing subjects and objects by predicate in the knowledge base. In addition to feature 20 (the ratio of objects by predicate), we include the number of existing objects per subject for predicate, and the existing subjects per the object, thus indicating the observed multiplicity in the knowledge base (in both directions).

- **Table group features:** we replicate the features extracted from individual tables, such as number of entities in subject and object columns, the number of potential relations, etc., but this time from merged tables; thus we have, for instance the ratio of potential relations where a predicate holds in the individual table, and also in the corresponding group of tables to which the individual table belongs.

A total of 65 features were described, however we considered only 60 features for classifying triples since the features related to the number of links and entities by cell (features: 21,22,23, 43 and 44) have a unique value as a result of applying the filters explained previously (Chapter 7); furthermore we do not consider existing triples in the knowledge base, where feature 38 was also removed, along with feature 1 indicating the number of tables from the corresponding Wikipedia article (feature 1 was not considered due to fact the tables were extracted and saved individually to process them in parallel, improving the time of processing). We will analyse the features correlation and feature importance later.

9.2 Classification algorithms

We use the same models that achieved better results in the baseline work [29]: Naive Bayes, Bagging Decision Tree (BDT), Random Forest (RF) and Logistic Regression (LR). The best results presented in the previous work correspond to Bagging Decision Tree, which achieves 81% precision and 79.4% recall, we additionally add two new algorithms: K-Nearest Neighbours (KNN) and Extreme Gradient Boost (XGBoost).

9.3 Dataset description

In the previous chapter we presented an initial manual evaluation of candidate triples from individual and merged tables; now we describe the sets of triples to be used to validate and test different classification algorithms.

For building the training dataset we randomly selected 700 triples from two sets I and F , and we classified them considering the four classes explained previously (see Section 8.3) (*correct*, *incorrect*, *contextual* and *unknown*); however finally we reject those triples classified as *contextual* and *unknown* and keep those triples where two raters agreed, collecting 600 triples classified as *correct* and *incorrect* from each set. We are not interested in *contextual* triples, which can be considered correct, but only if additional information is provided (e.g., start date, end date); we filter them from the training and validation data and although they can appear in the final set of triples, we will manually label these later in a separate process.

From each set of extracted triples we keep (out-of-bag) 100 triples for testing the models and use 500 triples for validating them in the different steps proposed in our methodology (explained later). In Table 9.3 we present the number of corresponding triples of each class and set; as we observe in the previous chapter the number of incorrect triples in each set is higher, but mostly in triples extracted from merged tables where we have 14% of incorrect triples; to deal with this issue we propose to use some known techniques to balance the sets of triples, we will present the results of applying these techniques in the next section.

Table 9.3: Validation and test set

	Training set		Test set	
	Correct	Incorrect	Correct	Incorrect
Triples from individual tables (I)	191 (38%)	309 (62%)	38 (38%)	62 (62%)
Triples from merged tables (F)	74 (15%)	426 (85%)	13 (13%)	87 (87%)

9.4 Classification models

In order to obtain comparable results applying both methods of relations extraction from tables (with and without grouping tables), we generate and validate models using different sets of features for the labeled sets of triples from I and F described previously.

- **Model A:** we validate the models for triples from set I with the baseline features.
- **Model B:** we add the *predicate and table features* (e.g. the range and domain of the predicate) proposed in Table 9.2, to evaluate how these features contribute to the classification over set I . In other words, Model B includes the baseline features and new features that we propose, defined for individual tables.
- **Model C:** we use the baseline features and all new features proposed including those for groups of tables (e.g. the ratio of rows where the predicate holds in the group of tables), considering that each individual table belongs to a specific group. With this experiment we will test how the features of groups of tables impacts the performance of classification over set I .
- **Model D:** we use all features for the set of triples F where candidate triples for single tables without any relation between a pair of columns are also included. We attempt to validate how many novel triples we can reach by applying this approach.

For validating and selecting the final model for each group of features and set of triples, we develop four experiments in order to identify the best models. These experiments are validated using the test set (of 100 triples) that we left out of the validation and training process for this purpose.

1. **Initial validation:** We validate the proposed classification algorithms using their parameters by default.
2. **Features selection:** We apply features selection removing highly correlated features and also features not correlated with the positive class. We do not consider only pair-wise correlation, for each pair of correlated features we looked for those with the maximum mean correlation with the remaining features. This experiment is designed to address the imbalance in data available for both classes (38% correct for I , 15% correct for F).
3. **Balancing training set:** We balance the training set adding triples extracted from tables that already exist in Wikidata.
4. **Hyper-parameters tuning:** We look for the best parameters for the algorithms by using Grid Search method where a set of different values for each parameter is given. The algorithm is validated with a combination of all parameters using cross-validation. Finally the models are configured with the best parameters and validated with the test set.

We next describe the results achieved for each of these four experiments; the details of the Feature selection task and Hyper-parameter tuning are described in Appendix D.

9.4.1 Model A

Considering the features proposed in the baseline method (see Table 9.1) we validate the four configurations previously described using initial validation (denoted A.1), feature selection (denoted A.2), balancing training set (denoted A.3) and hyper-parameter tuning (denoted A.4). All configurations are tested over a held-out training set with the original distribution of correct/incorrect triples.

For the first configuration we validate the classification models with default parameters (described in Appendix C); the results of this exercise are presented in Table 9.4 (A.1) where we observe that ensemble models achieve the higher F1-scores, while Naive Bayes achieves a high recall but with lower precision.

We apply features selection over the baseline features removing those with high correlation with each other and with remaining features; some of the features removed are the number of rows and columns in the table that are mostly correlated with the number of entities and relations in the table. After removing these features the models did not generally achieve better results as we see in Table 9.4 (A.2).

The training set of triples extracted from I consists of 38% of correct triples. We balance the classes using triples extracted from tables but that already exist in Wikidata; then we randomly selected 59 triples extracted from this set of triples and removed the same number of incorrect triples resulting in a new training set with 50% (250) of correct and 50% incorrect triples. The results of this experiment are presented in Ta-

Table 9.4: Results A1: Initial validation, A2: Features selection, A3:Balancing training set, A4: Hyper-parameters tuning

	A.1			A.2			A.3			A.4		
	P	R	F	P	R	F	P	R	F	P	R	F
KNN	54%	32%	40%	40%	18%	25%	54%	31%	40%	77%	18%	30%
NB	44%	87%	57%	49%	68%	57%	44%	87%	58%	44%	87%	58%
LR	78%	18%	30%	50%	18%	27%	78%	18%	30%	78%	18%	30%
RF	78%	66%	71%	76%	50%	60%	69%	53%	60%	72%	55%	62%
BDT	68%	71%	70%	73%	63%	68%	66%	61%	63%	71%	66%	68%
XGB	72%	61%	66%	73%	57%	65%	72%	61%	66%	73%	71%	72%

ble 9.4 (A.3). We do not validate the set of existing triples but rather assume they are correct. The results of balancing the training set with these triples does not present an improvement, while XGBoost reached the same results as for the initial configuration without balancing the training set.

The last experiment was conducted by setting the parameters of models based on the AUC metric and using the GridSearch method; the best combination of the parameters is used for validating the models. The details of the configuration of parameters for the ensemble models are presented in Appendix D.1.2; for KNN model we only changed the number of neighbors considering a value of 2, since the default value is 5. We do not change parameters for Logistic Regression and Naive Bayes algorithms, and so these models keep their results. We observe that KNN improved in precision, as XGBoost, which reached a better precision with similar recall (see Table 9.4 (A.4).

The results of validating the models over the test set are similar to the results achieved by applying cross-validation with 5 folds over the 500 triples. We also test the models generated in each fold with the test set for evaluating the models variation. XGBoost present less variation with higher precision and recall than other models.

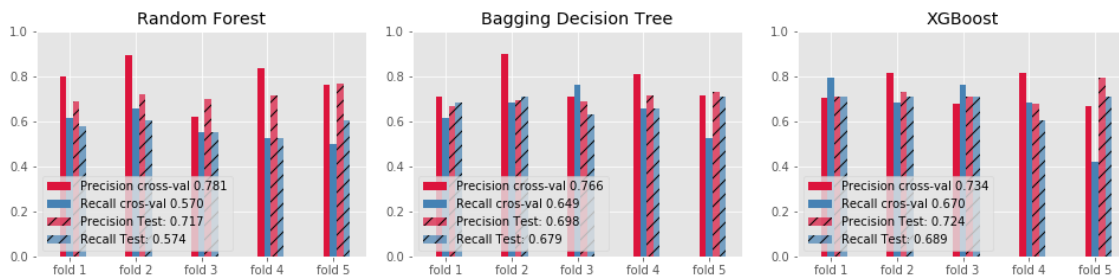


Figure 9.1: Cross-Validation Model A.4 (including results for 100 triples from held-out test set for each fold)

9.4.2 Model B

Following the same methodology as for the set of features in Model A, we present the results of validating models for the same set of triples from I adding the new *predicate* and *table features* defined for individual tables.

Including the new features we observe that models improve their precision. XGBoost is the best model with 74% precision and 75% recall as we observe in Table 9.5 (B.1). In the analysis of features importance presented in Appendix D.2.1, the features about the maximum number of objects and subjects for a predicate achieve a higher Gini score (0.08) with the ensemble models, then we attribute the performance of these models to applying the new proposed features.

After applying features selection for this new set of features, we observed that the models achieve better precision and recall (see Table 9.5 (B.2)) and also by setting model parameters, where Random Forest achieves a good precision of 84%.

Table 9.5: Results B1: Initial validation, B2: Features selection, B3:Balancing training set, B4: Hyper-parameters tuning

	B.1			B.2			B.3			B.4		
	P	R	F	P	R	F	P	R	F	P	R	F
KNN	75%	32%	44%	80%	32%	45%	76%	42%	54%	75%	32%	44%
NB	42%	92%	57%	41%	92%	57%	41%	92%	56%	41%	92%	59%
LR	100%	2%	5%	0%	0%	0%	56%	24%	33%	100%	2%	5%
RF	71%	66%	68%	95%	55%	70%	75%	71%	73%	84%	68%	75%
BDT	72%	68%	70%	76%	68%	72%	70%	74%	71%	74%	68%	71%
XGB	74%	76%	75%	78%	74%	76%	71%	76%	73%	74%	76%	75%

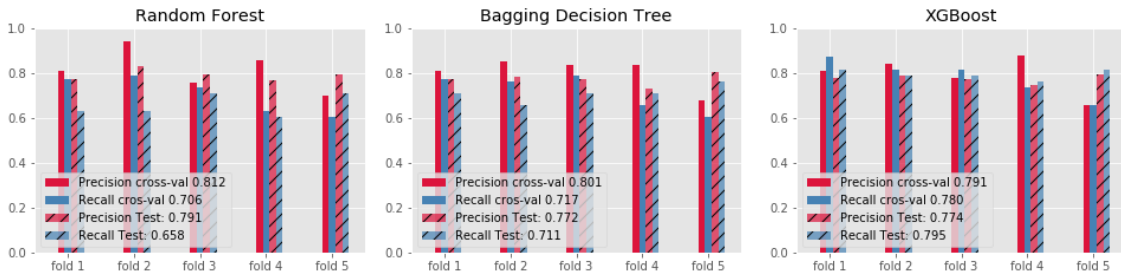


Figure 9.2: Cross-Validation Model B.4 (including results for 100 triples from held-out test set for each fold)

Finally in Figure 9.2, we present the results of 5-fold cross validation, including the results for the held-out set of 100 test triples. Though there is variance between the folds, less variance is presence for the test set. In general, we conclude that the additional features we propose for individual tables help to improve classification performance.

9.4.3 Model C

We also validate the models for classifying triples from single tables but adding features about the corresponding groups to which the single tables belong; the results of this validation considering the same configurations are presented in Table 9.6.

Considering all features proposed we see a notable increase in the precision achieved by XGBoost which reached the best results by setting its parameters (C.4), while Bagging Decision Tree only achieve a good precision by applying features selection (C.2). We consider that given the large number of features (60) the parameters configuration and features selection are more important for the models achieve their best performance. In Appendix D.3.1 we show the features importance based on Information Gain and we observe that features related to the ratio of rows and relations where predicate holds in the group of tables take great importance for tree based models, and also that these features are most correlated with the positive class.

Table 9.6: Results C1: Initial validation, C2: Features selection, C3:Balancing training set, C4: Hyper-parameters tuning

	C.1			C.2			C.3			C.4		
	P	R	F	P	R	F	P	R	F	P	R	F
KNN	65%	45%	53%	68%	50%	58%	53%	63%	57%	50%	16%	24%
NB	66%	11%	18%	67%	11%	18%	63%	13%	21%	67%	11%	18%
LR	60%	7%	14%	67%	11%	18%	36%	18%	24%	60%	8%	14%
RF	70%	50%	58%	72%	61%	66%	68%	71%	69%	76%	34%	47%
BDT	69%	63%	66%	82%	61%	66%	63%	68%	66%	70%	55%	62%
XGB	82%	71%	76%	81%	68%	74%	71%	76%	73%	85%	76%	81%

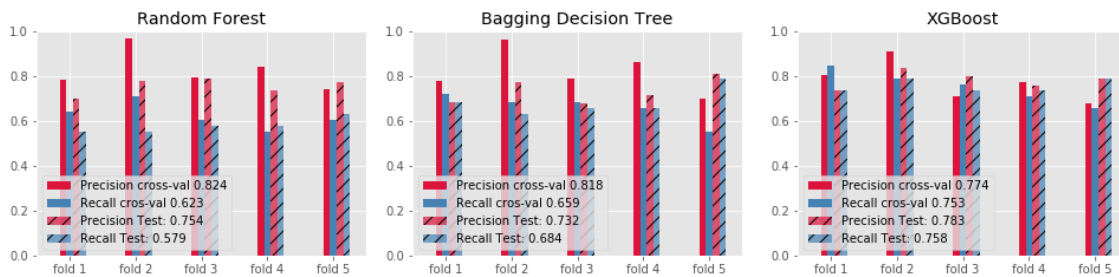


Figure 9.3: Cross-Validation Model C.4 (including results for 100 triples from held-out test set for each fold)

Of this set of features, Random Forest and Bagging Decision Tree reached a precision of over 80% with cross-validation; however when we validate the models over the test set the results decrease except for XGBoost. We can conclude that for the triples that can already be extracted for individual tables, having group-level features helps to improve the classification performance, perhaps because such features smooth out variances that are present in smaller individual tables.

9.4.4 Model D

The last models validation consist of use all features with the triples extracted from set F . In comparison with the set of triples from I the training set from F is much more unbalanced with 14% correct triples, which motivates the use of oversampling and undersampling techniques to balance the training set; however by applying oversampling, we found that the precision and recall decrease, while with undersampling we only achieved better recall but with lower precision, hence we do not present these additional results, but rather present the same configurations as seen for previous experiments. The results are shown in Table 9.7.

Table 9.7: Results D1: Initial validation, D2: Features selection, D3:Balancing training set, D4: Hyper-parameters tuning

	D.1			D.2			D.3			D.4		
	P	R	F	P	R	R	P	R	F	P	R	F
KNN	25%	7%	11%	42%	23%	30%	28%	30%	30%	25%	7%	11%
NB	50%	23%	31%	60%	23%	33%	60%	23%	33%	50%	23%	31%
LR	50%	7%	13%	25%	15%	19%	27%	46%	34%	50%	7%	13%
RF	40%	15%	22%	36%	31%	33%	45%	38%	42%	50%	15%	23%
BDT	50%	15%	23%	60%	46%	52%	44%	53%	48%	50%	15%	23%
XGB	66%	31%	42%	62%	38%	48%	43%	53%	46%	85%	46%	60%

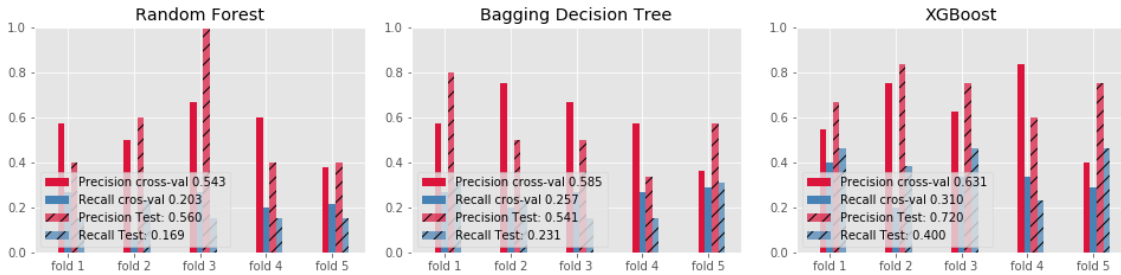


Figure 9.4: Cross-Validation Model D.4 (including results for 100 triples from held-out test set for each fold)

Removing highly correlated features we observed an increment in recall and also by balancing the training set; however precision stays low. On the other hand, the best result we obtained was by applying XGBoost with tuned parameters. In reference to Figure 9.4 which shows the results obtained with cross-validation, we observe high variance caused by the low number of correct triples per fold; XGBoost reached the best performance for this unbalanced training set achieving 72% precision and 40% recall in the mean results. We conclude that classification of correct/incorrect triples in the F set of candidate triples is greatly complicated by the significantly higher ratio of incorrect triples. However, XGBoost with parameter tuning is capable of achieving precision competitive with that for models on the I set, though with much lower recall. Given that the F set contains many more correct triples in absolute terms, we require

further results to see if this trade-off of more triples with poorer recall is a positive one.

9.5 Model Selection

Having compared different methods of validating the models for the various set of features and both set of triples we finally select the best model for classifying the whole dataset of extracted triples. We are interested in getting a good quantity of triples for enriching Wikidata with good precision.

In the previous section we presented the results of precision, recall and F1-score, but we also obtained the AUC metric that is shown in Appendix D for each analysed model. XGBoost achieves 0.89, 0.90, 0.88 and 0.92 for each model; while these results are not different for the AUC obtained for Random Forest and Bagging Decision Tree, however considering that the sets of triples are unbalanced we consider the precision and recall metrics where XGBoost achieved the best results.

Finally, we considered the option of increasing the number of triples in the training sets to obtain better results; however the learning curves (Appendix D) indicate that after 300 triples there is not a significant improvement. Hence the final models we train will use the 600 triples for I and F labeled previously.

9.6 Evaluation

Our primary goal is to compare the results for relation extraction with and without grouping tables. Our hypothesis is that grouping tables can help in two ways. First, more robust statistics for features can be extracted from groups of tables versus individual tables, which should improve results even for candidate triples extracted already from individual tables. Second, we should be able to extract additional candidate triples from groups of tables that would not be extracted from individual tables. In our final experiments, we thus compare how many triples can be extracted of Wikipedia and with what level of precision for these different settings.

As we mentioned before the results obtained with new features (Model B) over the set of triples from individual tables I are better than those achieved only with baseline features; thus we train a model with these. The second setting takes triples from I adding features from groups of tables (Model C). The results of both models will be compared with the classification of triples extracted by merging tables from set F (Model D).

We train the models with 600 labeled triples from sets I and F (Model B and Model

C using set I and Model D using set F) and apply the models achieving the number of correct triples presented in Table 9.8. From triples classified as correct by each model, we randomly selected and validated 200 triples. The results of the validation were 71% precision for triples from set I without including features about the related groups of tables, 75% precision for triples from set I including features of related groups of tables and 70% precision for triples from set F .

The final results show that there is an improvement of the precision of triples extracted from tables adding the features of the corresponding groups of tables (Model B). Some of these features that contributed to the classification improvement were the ratio of rows and relations between a pair of entities where a predicate holds in the corresponding group, as well as the maximum number of objects and subjects for a predicate. However, the increased precision of Model C comes at the cost of a reduced number of triples versus Model B; about 265,000 fewer triples are extracted. Even though we did not get a higher precision for triples from merged tables (Model D) where a large number of incorrect triples are present, we achieve a precision of 70%, which is competitive with the Model B baseline without grouping; furthermore, we achieve over 800,000 more triples.

With these results we conclude that it is possible to achieve better results on relations extraction by grouping related tables, both in terms of being able to define more robust features (Model B vs. Model C), and also in terms of being able to extract more triples with similar levels of precision (Model B vs. Model D). These results are illustrated in Figure 9.5 and Figure 9.6, which, by varying the models' threshold, show the precision possible versus the number of triples extracted in the labelled sample and projected for the full sets of candidate triples respectively, without considering an specific knowledge area or type of tables and also extracting triples from small tables without any relation.

Table 9.8: Results of classifying triples extracted from individual I tables (Model B and C) and triples extracted by merging tables (Model D)

Triples and features	Total Correct triples	Validation	Total	Precision
Model B	6,773,694	Correct	141	71%
		Incorrect	53	
		Contextual	4	
		Unknown	2	
Model C	6,508,755	Correct	150	75%
		Incorrect	49	
		Contextual	0	
		Unknown	1	
Model D	7,591,258	Correct	140	70%
		Incorrect	57	
		Contextual	1	
		Unknown	2	

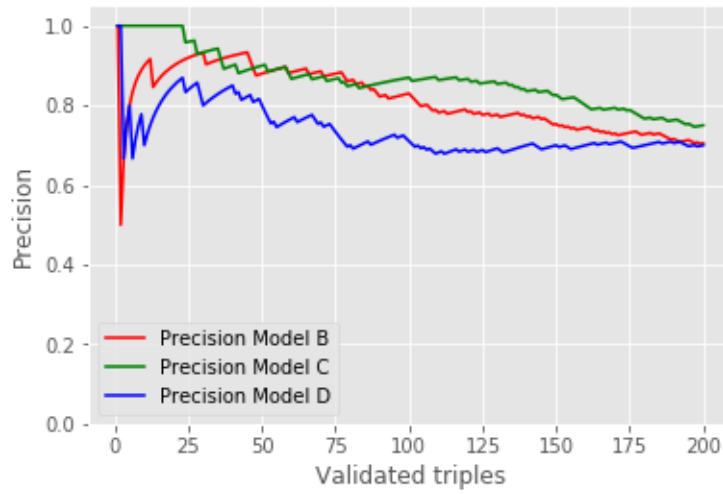


Figure 9.5: Precision vs number of validated triples

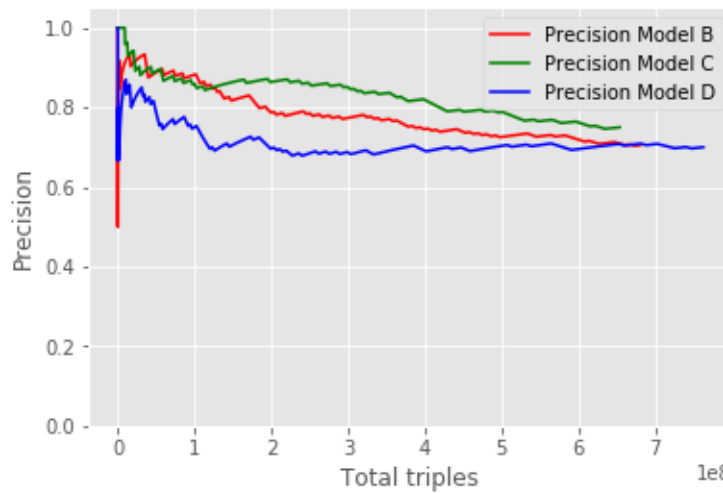


Figure 9.6: Precision vs total triples classified as correct (estimated from labeled sample)

Conclusions

In the present work we propose grouping HTML tables for extracting relations with better precision considering the small proportion of contextual and correlated information that we can find in an individual table.

The research questions of this work have been addressed for extracting triples from Wikipedia tables using Wikidata as a reference knowledge base.

We first conducted a literature review to find related works that group and merge tables for information extraction; however we did not find a similar proposal that attempts to merge tables for extracting relations without considering a specific knowledge area or type of tables.

We then evaluate clustering methods using the headers of tables and found that in Wikidata there exist at least 500,000 pairs of tables with a common header and groups with more than 10,000 tables. For grouping a larger number of tables, we apply some text processing tasks for stemming the headers text. To ensure that we merge tables with similar types of content, we identify the types of content of each column. We also apply a method for refining the initial grouping of tables, splitting groups having conflicting relations. With these methods we achieve about 1.1 million groups of tables over which we apply the relation extraction process to extract candidate triples.

Many of the candidate triples extracted from the previous phase are incorrect. Hence following the same relation extraction process proposed by Muñoz et al. [29], we use classification methods to distinguish correct and incorrect candidate triples. To facilitate this process, we developed a web application that facilitates the labeling of triples extracted from tables. With this application we achieve a considerable number of labeled triples that were used for validating the classification methods over two sets of candidate triples, extracted based on the method proposed by Muñoz et al. [29] and our method based on grouping tables. We also evaluate different classification models, looking for the best model by applying techniques of feature selection and model parameters tuning, trying to select the best model for extracting relations with both methods.

The results show that by adding features about the corresponding group of tables we can increase the precision of relations extracted from an individual table. With our

grouping method we can also extract new triples that cannot otherwise be extracted from individual tables; this allows us to obtain more triples from groups of tables with similar precision to individual tables.

We obtained 7,591,258 novel triples with a precision of 70%, (see examples of extracted triples in Appendix D.5.1) by merging tables, considering existing entities and properties in the Wikidata knowledge base. It is hard to compare the precision achieved by our proposal with the precision of the available data in the Wikidata knowledge base: currently there is no clear evaluation of the precision of Wikidata’s content [18, 34]. Our dataset can be uploaded to Wikidata, and validated previously to guarantee the data quality before to be published¹.

The proposed method for grouping and merging tables considers tables with the same headers; however we demonstrate that clustering methods can be used for grouping tables based on more specific knowledge areas, where expert people can evaluate the clusters; furthermore a full merging technique can be used to increase the candidate triples extracted from a group of merged tables.

Since the method was validated with Wikipedia tables, where Wikipedia links can be mapped directly to a knowledge base; it cannot be used with tables from other sources without mapping the table content to the knowledge base entities previously; such a mapping can be carried out with the aforementioned Entity Linking techniques.

As future work, we are curious about extracting n-ary relations from tables using the same method, since we could identify different types of content in tables that can be used for proposing n-ary relations. Additionally by increasing the number of linked entities within tables to a knowledge base (i.e. using Entity Linking techniques rather than relying on explicit links) it should be possible to increase the confidence of extracted relations and even extract more novel triples.

¹Wikidata provides tools to import datasets that are validated by Wikidata Community users https://www.wikidata.org/wiki/Wikidata:Dataset_Imports.

Bibliography

- [1] D Abián, F Guerra, J Martínez-Romanos, and Raquel Trillo-Lado. Wikidata and dbpedia: a comparative study. In *International KEYSTONE Conference on Semantic Keyword-Based Search on Structured Data Sources*, pages 142–154. Springer, 2017.
- [2] Sean Bechhofer, Yeliz Yesilada, Robert Stevens, Simon Jupp, and Bernard Horan. Using ontologies and vocabularies for dynamic linking. *IEEE Internet Computing*, 12(3):32–39, 2008.
- [3] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, IDEA '13*, pages 18–26, New York, NY, USA, 2013. ACM.
- [4] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Tabel: Entity linking in web tables. In Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d’Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015*, pages 425–441, Cham, 2015. Springer International Publishing.
- [5] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. volume 7, pages 154 – 165, 2009. *The Web of Data*.
- [6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
- [7] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. In *Proceedings of the VLDB Endowment*, 1(1):538–549, 2008.
- [8] Michael J Cafarella, Alon Y Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. Uncovering the relational web. In *WebDB*, 2008.
- [9] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowl-*

edge discovery and data mining, pages 160–172. Springer, 2013.

- [10] Matteo Cannavicchio, Lorenzo Ariemma, Denilson Barbosa, and Paolo Merialdo. Leveraging wikipedia table schemas for knowledge graph augmentation. In *Proceedings of the 21st International Workshop on the Web and Databases*, page 5. ACM, 2018.
- [11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.
- [12] Marco Costantino, Richard G. Morgan, Russell James Collingham, and Roberto Garigliano. Natural language processing and information extraction: qualitative analysis of financial news articles. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering, CIFER 1997, New York City, USA, March 24-25, 1997*, pages 116–122, 1997.
- [13] Eric Crestan and Patrick Pantel. Web-scale table census and classification. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11*, pages 545–554, New York, NY, USA, 2011. ACM.
- [14] Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.
- [15] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. Building the dresden web table corpus: A classification approach. *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, pages 41–50, 2015.
- [16] B Efron and RJ Tibshirani. An introduction to the bootstrap. chapman and hall, new york, ny. *Farrell, J., Johnston, M. and Twynam, D.(1998), "Volunteer motivation, satisfaction, and management at an elite sporting competition", Journal of Sport Management*, 12:288–300, 1993.
- [17] David W. Embley, Cui Tao, and Stephen W. Liddle. Automatically extracting ontologically specified data from html tables with unknown structure. In *Proceedings of the 21st International Conference on Conceptual Modeling (ER'02)*, pages 322–327, 2002.
- [18] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, 9(1):77–129, 2018.
- [19] Michael Färber, Basil Ell, Carsten Menne, and Achim Rettinger. A comparative

- survey of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web Journal*, 1:1–5, 2015.
- [20] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
- [21] Anna Lisa Gentile, Petar Ristoski, Steffen Eckel, Dominique Ritze, and Heiko Paulheim. Entity matching on web tables: a table embeddings approach for blocking. In *EDBT*, pages 510–513, 2017.
- [22] Anabel Gómez-Ríos, Julián Luengo, and Francisco Herrera. A study on the noise label influence in boosting algorithms: Adaboost, gbm and xgboost. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 268–280. Springer, 2017.
- [23] Matthew Hurst. Layout and language: Challenges for table understanding on the web. In *In proceedings of the International Workshop on Web Document Analysis*, pages 27–30, 2001.
- [24] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.
- [25] Larissa R. Lautert, Marcelo M. Scheidt, and Carina F. Dorneles. Web table taxonomy and formalization. *SIGMOD Rec.*, 42(3):28–33, October 2013.
- [26] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3(1-2):1338–1347, 2010.
- [27] Andreas Mayr, Harald Binder, Olaf Gefeller, and Matthias Schmid. The evolution of boosting algorithms. *Methods of information in medicine*, 53(06):419–427, 2014.
- [28] David Milne and Ian H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *In Proceedings of AAAI 2008*, 2008.
- [29] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Triplifying i’s tables. In *Proceedings of the First International Conference on Linked Data for Information Extraction - Volume 1057, LD4IE’13*, pages 26–37, Aachen, Germany, Germany, 2013. CEUR-WS.org.
- [30] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Using linked data to mine rdf from wikipedia’s tables. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 533–542. ACM, 2014.
- [31] Stéphane Mussard, Françoise Seyte, and Michel Terraza. Decomposition of gini

and the generalized entropy inequality measures. *Economics Bulletin*, 4(7):1–6, 2003.

- [32] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web, WWW '16*, pages 1419–1428, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.
- [33] David Pinto, Michael Branstein, Ryan Coleman, W Bruce Croft, Matthew King, Wei Li, and Xing Wei. Quasm: a system for question answering using semi-structured data. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 46–55. ACM, 2002.
- [34] Alessandro Piscopo and Elena Simperl. What we talk about when we talk about wikidata quality: a literature survey. In *Proceedings of the 15th International Symposium on Open Collaboration*, page 17. ACM, 2019.
- [35] Aleksander Pivk, Philipp Cimiano, York Sure, Matjaz Gams, Vladislav Rajkovic, and Rudi Studer. Transforming arbitrary tables into logical form with TARTAR. *Data Knowl. Eng.*, 60(3):567–595, 2007.
- [36] J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.
- [37] Abhishek Singh Rathore and Devshri Roy. Ontology based web page topic identification. *International Journal of Computer Applications*, 85(6), 2014.
- [38] Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 366–372. Springer, 2017.
- [39] Deepika Sharma and Me Cse. Stemming algorithms: a comparative study and their analysis. *International Journal of Applied Information Systems*, 4(3):7–12, 2012.
- [40] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. Liege:: Link entities in web lists with knowledge base. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12*, pages 1424–1432, New York, NY, USA, 2012. ACM.
- [41] Sonit Singh. Natural language processing for information extraction. *arXiv preprint arXiv:1807.02383*, 2018.
- [42] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents*

on the World Wide Web, 6(3):203–217, 2008.

- [43] Zareen Syed Varish Mulwad, Tim Finin and Anupam Joshi. T2LD: Interpreting and Representing Tables as Linked Data. In *Proceedings of the Poster and Demonstration Session at the 9th International Semantic Web Conference, CEUR Workshop Proceedings*, November 2010.
- [44] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 4(9):528–538, 2011.
- [45] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57:78–85, 2014.
- [46] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q Zhu. Understanding tables on the web. In *International Conference on Conceptual Modeling*, pages 141–155. Springer, 2012.
- [47] Yalin Wang and Jianying Hu. A machine learning based approach for table detection on the web. In *Proceedings of the 11th International Conference on World Wide Web, WWW '02*, pages 242–250, New York, NY, USA, 2002. ACM.
- [48] Minoru Yoshida, Kentaro Torisawa, and Junichi Tsujii. A method to integrate tables of the world wide web. In *Pacific Association for Computational Linguistics*, pages 332–341, 2001.

Appendix A

Information Extracted from Wikidata

To know the topics which the tables are related with, we extracted the class of entities corresponding to article's titles from Wikidata. We present a word cloud with the top 30 classes by number of tables in Figure A.1. There are many articles related to sport events of different types, hence many tables can be related to players, teams and sport events having common headers.



Figure A.1: Word cloud of article classes from which tables were extracted

The most common entities that appear as subject in the triples are countries: Q30:United States appears in 43,452 triples while the entity Q193592:Midfielder appears in 30,859; these values are expected given the most common articles' topics.

Figure A.2 shows the number of entity classes identified for individual columns in the tables; about 1.2 million tables have one class per column, while 451 tables have more than 100 entity classes; the columns identified with a large number of different entity classes are those with names *note* and *description* that usually describe many entities by cell. The values were obtained selecting the class with more instances by entity,

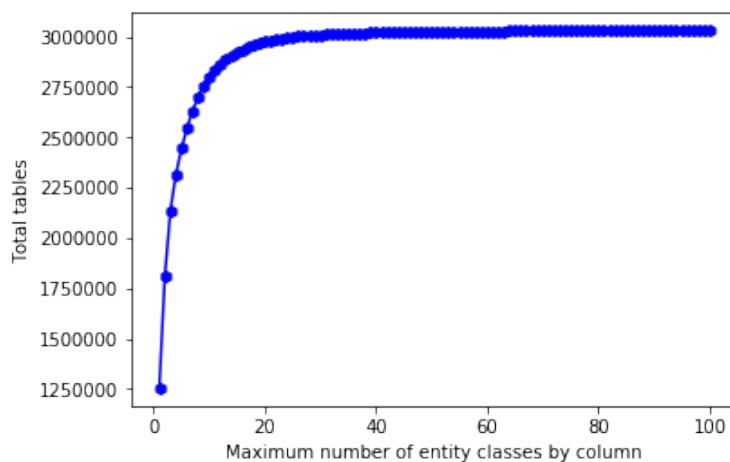


Figure A.2: Number of different classes by column in tables

given that an entity belongs to multiple classes.

Triples with 620 different predicates were extracted from tables, considering the entities in a pair of columns and also between the article’s entity and entities inside tables. In Table A.1 we list the top 20 most common predicates. In this table we can observe specific predicates such as those related to sport content and locations, but we observe also more general predicates such as P361:part of. Despite the fact that Wikidata recommends the use of more specific predicates, we found many triples with generic types, for example instead of the predicate P54:member of sport team between a player and a sport team we can find the relation P361:part of which though not incorrect, since P54 is a *subproperty* of P361, is not as informative nor as desirable as the P54 relation (where appropriate). Still, we do not reject triples with more general relations, therefore these will be proposed for candidate triples since they appear in the knowledge base for similar pair of entities; in other words, were Wikidata’s recommendations to use more specific properties strictly followed by Wikidata editors, our method would not propose any general relations.

On the other hand from the total number of predicates, there are 210 that appear in fewer than 10 triples; Table A.2 shows a random selection of these predicates.

Table A.1: Top 20 predicates from existing triples

<i>Predicate</i>	<i>Triples</i>
P54 :member of sports team	299,115
P131 :located in the administrative territorial entity	284,375
P27 :country of citizenship	251,741
P161 :cast member	243,586
P17 :country	146,239
P175 :performer	130,034
P413 :position played on team / speciality	125,401
P102 :member of political party	104,402
P57 :director	90,140
P150 :contains administrative territorial entity	66,737
P361 :part of	65,952
P31 :instance of	62,386
P1344 :participant of	42,404
P58 :screenwriter	38,908
P166 :award received	37,032
P19 :place of birth	36,282
P197 :adjacent station	36,036
P3450 :sports season of league or competition	34,347
P527 :has part	33,618
P495 :country of origin	33,488

Table A.2: Predicates with fewer than 10 triples

<i>Predicate</i>	<i>Triples</i>
P912 :has facility	2
P795 :located on linear feature	2
P3730 :next higher rank	3
P2647 :source of material	2
P5136 :less than	6
P2888 :exact match	3
P489 :currency symbol description	4
P3161 :has grammatical mood	4
P4100 :parliamentary group	2
P3263 :base	5
P1478 :has immediate cause	5
P769 :significant drug interaction	2
P2289 :venous drainage	4
P532 :port of registry	8
P1420 :taxon synonym	2
P3828 :wears	1
P817 :decay mode	4
P3259 :intangible cultural heritage status	5
P924 :medical treatment	2
P1531 :parent(s) of this hybrid	5

Appendix B

Inter-rate agreement

In Table B.1, Table B.2 and Table B.3 we show statistics regarding the inter-rater agreement for the triples annotated for sets I , $F - I$ and $G - I$ respectively. We show the number of correct and incorrect triples annotated for both annotators, the *Kappa Cohen k* statistic, the relative observed agreement $Pr(a)$ among raters (accuracy), and $Pr(e)$ is the hypothetical probability of chance agreement; using the observed data to calculate the probabilities of each observer randomly seeing each class.

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}$$
$$Pr(e) = \frac{1}{N^2} \sum_k n_{k1}n_{k2}$$

For k classes, number of items N and n_{ki} the number of times rater i predicted class k .

Table B.1: Inter-rate agreement for triples from I

	Correct	Incorrect	Contextual	Unknown	Annotator 1
Correct	31	6	0	0	37
Incorrect	5	51	0	1	57
Contextual	1	1	2	0	4
Unknown	1	1	0	0	2
Annotator 2	38	59	2	1	$Pr(a) = 84\%$
$Pr(e) = 48\%$					$\kappa = 69\%$

Table B.2: Inter-rate agreement for triples from F-I

	Correct	Incorrect	Contextual	Unknown	Annotator 1
Correct	6	2	0	3	11
Incorrect	2	77	1	1	81
Contextual	0	3	1	0	4
Unknown	1	3	0	0	4
Annotator 2	9	85	2	4	$Pr(a) = 84\%$
$Pr(e) = 70\%$					$\kappa = 47\%$

Table B.3: Inter-rate agreement for triples from G-I

	Correct	Incorrect	Contextual	Unknown	Annotator 1
Correct	4	0	0	1	5
Incorrect	1	87	0	3	91
Contextual	0	0	0	0	0
Unknown	0	4	0	0	4
Annotator 2	5	91	0	4	$Pr(a) = 91\%$
$Pr(e) = 83\%$					$\kappa = 46\%$

Appendix C

Model parameters description

We describe some relevant parameters that were configured for the classification algorithms.

K-Nearest Neighbors

The main parameter of the KNN classifier algorithm is the number of neighbors **n_neighbors** to evaluate the class. Other parameters such as the distance measure (default=Euclidean) and searching algorithm (default=auto) can also be selected; however we keep those parameters at their default settings.

Naive Bayes

The Gaussian Naive Bayes algorithm only allows the configuration of two parameters: **priors** (default=n_classes) which is adjusted according to the weights of classes, and **var_smoothing** (default=1e-9) which according to the Sci-Kit Learn library documentation is the portion of the largest variance of all features that is added to variances for calculation stability. We keep the default values of these parameters.

Logistic Regression

This algorithm provides more parameters that can be configured like the regularization method **penalty** (default=l2), the **class_weight** (default=None) where we passed the value *balanced*; and the maximum number **max_iter** (default=100) of iterations taken for the solvers to converge.

Random Forest

Random Forest like the other ensemble models requires the configuration of certain parameters for avoiding overfitting in the construction of the model. We describe the parameters that we consider relevant for our data.

1. **n_estimators** : (default=10) number of base estimators in the ensemble model. The value of this parameter is selected by applying cross-validation with different values, we consider five values to be evaluated [100,200,300,400,500].
2. **min_samples_leaf** : (default=1) the minimum number of samples required to be at a leaf node, the algorithm will split the nodes until the leaves have this number of samples. The default value for this parameter is 1, we configure this parameter with values: [1,2,3,4,6]
3. **max_features** : (default= \sqrt{n}), the square root of the number of features (n) is considered. The algorithm evaluates this number of features in each split.
4. **max_depth** : the nodes are expanded until all leaves are pure or until all leaves contain less than **min_samples_split**; we validate the model with values: [2,4,6,8,10,20] to avoid overfitting.
5. **class_weight** : we configured this parameter with value *balanced*.

Bagging Decision Tree

We use a Decision Tree as the base estimator for the Bagging model, and configure the same features as Random Forest, namely **n_estimators**, the **max_depth** and **min_samples_leaf** with the same list of proposed values.

XGBoost

The nature of XGBoost is the Gradient descent method for adjusting the weights in each step of classification process, in this manner the method uses more parameters that are selected according to the problem of classification. Next we describe the main parameters according to its documentation ¹, and the values used for selecting the best parameters.

1. **n_estimators**: (default=100) as in Bagging Decision Tree the evaluated numbers of estimators were [100,200,300,400].
2. **learning_rate**: (default=0.1) is used for shrinking the weights for non-classified samples in each step. We tested the values [0.0001,0.001,0.01,0.1,0.2]

¹<https://xgboost.readthedocs.io/en/latest/>

3. **min_child_weight**: (default=1) is used to control overfitting, avoiding that the model split a node with the min child weight, which is the sum of the weights of all observations required in a child. We used the values [0,1,2,3]
4. **max_depth**: (default=3) is the maximum depth of a tree. We test the model with values [2,4,6,8,10,20].
5. **reg_alpha**: (default=0) denotes L1 regularization term on weights, used for shrinking less important features. We tested the values [0.0001, 0.001, 0.01, 0.1],
6. **scale_pos_weight**: (default=1) is used for balancing the weights for classes. We conserve the default value of 1 for balancing the weights as the number of samples in each class.

Appendix D

Classification Models

D.1 Model A

D.1.1 Features selection

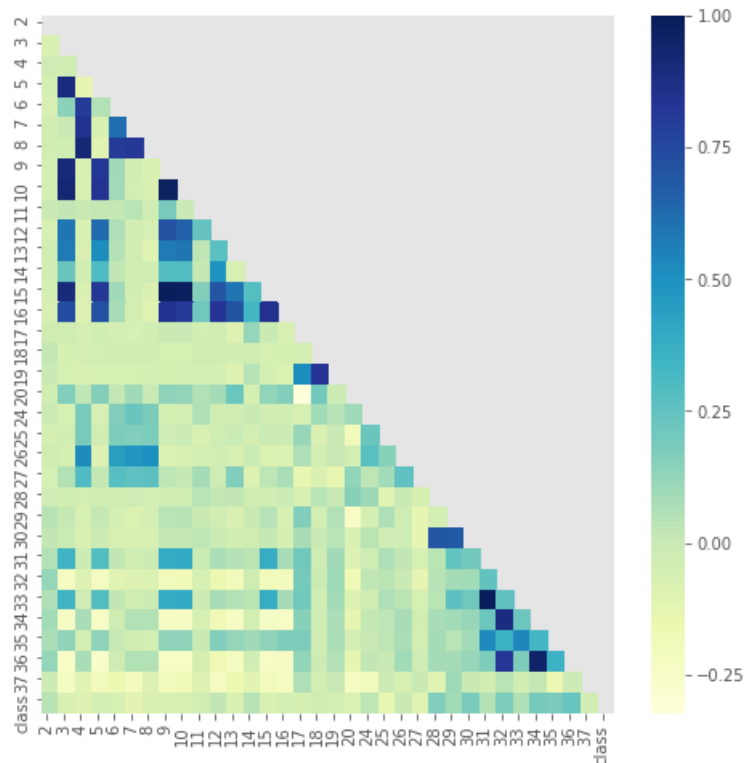


Figure D.1: Correlation Matrix (Model A)

To obtain the features' importance we explore the correlation between features and also with the labeled class by using the *Pearson* coefficient which gives a value in the range -1 to 1 , where a positive value indicates a positive correlation between

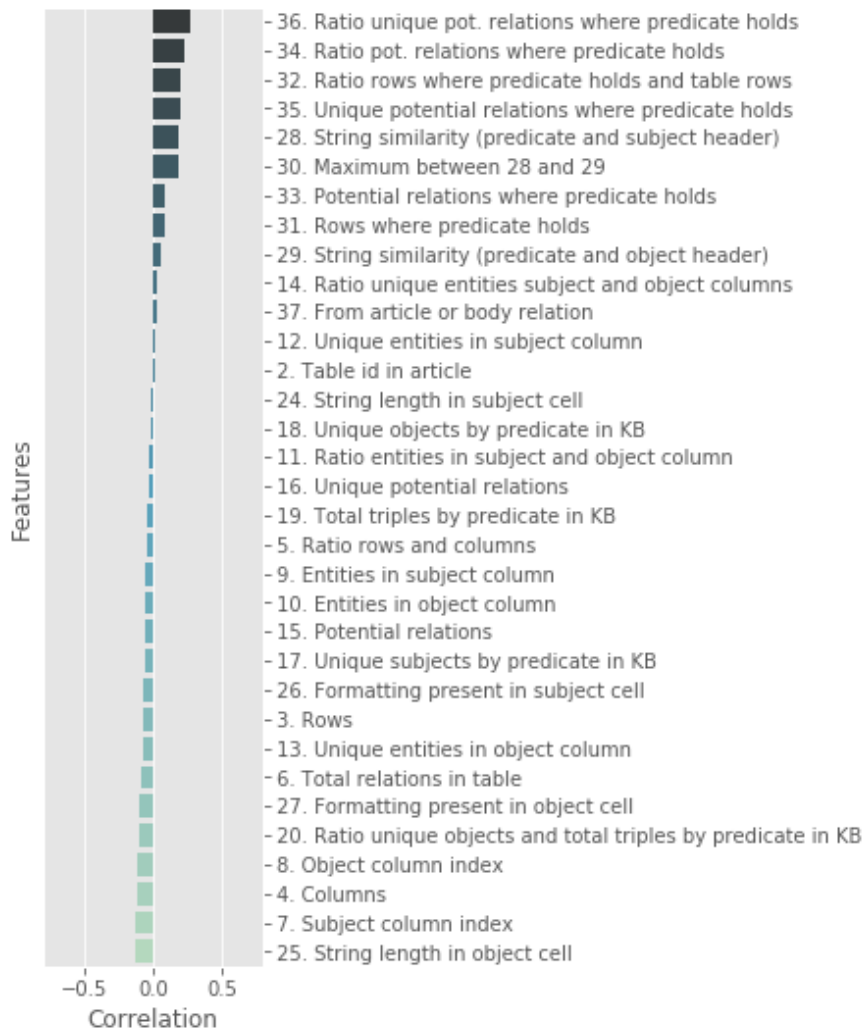


Figure D.2: Correlation between features and class (Model A)

two variables, 0 when there is no correlation and a value lower than 0 when there is a negative correlation; furthermore we obtain the features importance using an Information Gain metric, which will be explained later.

According to the correlation matrix in Figure D.1, we observe some highly correlated features, where we selected a cut of 0.80 and present them in Table D.1; in this table are presented the number of table rows (Feature 3) which is correlated with the number of entities in table columns (Feature 9 and Feature 10); as well as the number of columns (Feature 4) is also highly with the indexes of subject and object column (Features 7, 8), which is an expected behaviour. Along the same lines in the same way the features that assess the total number of entities and relations (Feature 15) versus the unique number of them (Feature 16) are correlated, including for example the number of potential relations where the predicate holds in the table and the number of unique potential relations (Features 34 and 36).

On the other hand we observe the features correlated with the positive class, which

Table D.1: Features with high correlation for Model A

Feature_1	Feature_2	Correlation
3	5	0.905
3	9	0.901
3	10	0.905
3	15	0.894
4	7	0.857
4	8	0.916
5	9	0.816
7	8	0.808
9	10	0.970
9	15	0.995
9	16	0.873
10	15	0.974
10	16	0.835
12	16	0.879
15	16	0.826
31	33	0.997
32	36	0.821
32	34	0.888
34	36	0.944

are those regarding the predicate frequency in a table. In Figures D.3, D.4 and D.5 we see that correct triples (points in red) have higher ratio in three features; conversely these features have a negative correlation with the number of rows in table, since the ratio of relations with the same predicate may be small as the number of table rows increases. Likewise features about the similarity of the predicate with column names (Features 28,29,30) correlate well with the class labeled correct.

Figures D.6 and D.7 show the number of correct and incorrect triples for different string similarity values, where we observe that 90% of triples have a value lower than 0.1 but those with higher value correspond mostly to correct triples. In Table D.2, we present some examples of triples that have a value of 1 for these features.

We observe that there are few columns of tables with similar names to the predicate labels (from Wikidata), which makes it difficult to assign relations based on these names; however being part of the features, the similarity between columns and predicate labels contribute to the classification process.

After analysing the features correlation we obtained the Information Gain score based on the Gini Index, which gives a measure of how pure are the leaves after splitting the tree by a given feature.

To analyse this measure we present the Gini score obtained for features with the tree

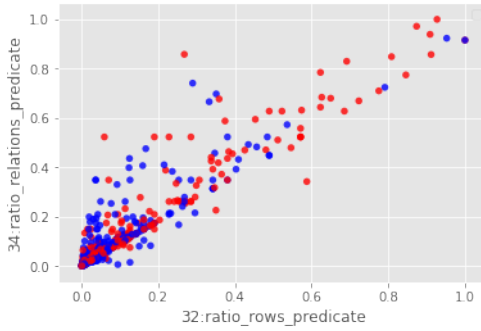


Figure D.3: Features 32 - 34

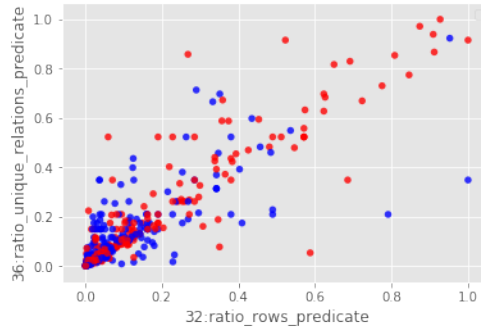


Figure D.4: Features 32 - 36

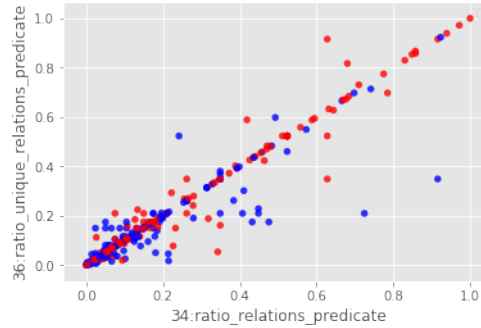


Figure D.5: Features 34 - 36

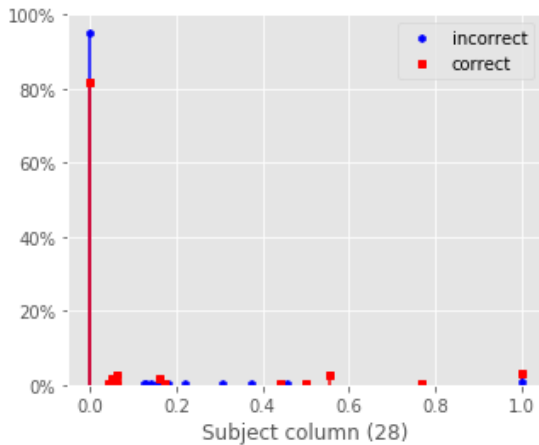


Figure D.6: String similarity between predicate and subject column name

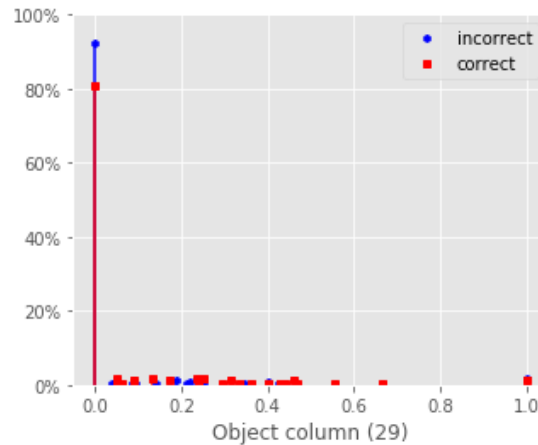


Figure D.7: String similarity between predicate and object column name

ensemble models proposed for the classification.

We observe that features related to the number of unique subjects and objects for the predicate in Wikidata (Features 17,18,19,20) have a higher score and also the ratio of unique relations where the predicate holds in the table (Feature 36). On the other hand the features about the presence of formatting content in table cells (Features 26, 27) and the feature that indicates if the triple correspond to the article's entity (Feature 37) present a small score.

Table D.2: Examples of triples with maximum similarity between column names and predicate

Column 1	Column 2	Predicate
director@3	titl@3	P57-1 :director
album@3	record_label@3	P264 :record label
countri@4	studio@3	P17-1 :country
album@3	record_label@3	P264 :record label
locat@3	school@3	P276-1 :location
2_team_place_in_nation_championship@3	countri@3	P17 :country
cathedr@3	present_archdioc_or_dioc@3	P1885-1 :cathedral
publisher@3	titl@3	P123-1 :publisher
leagu@3	recruit_from@3	P118-1 :league
genr@3	titl@3	P136-1 :genre
locat@3	univ@3	P276-1 :location
locat@3	main_articl@3	P276-1 :location

With this analysis we consider the following criteria to remove features for this set of triples:

- From each pair of correlated features we removed the feature with higher mean correlation with all features, thus we removed features: 3,4,5,9,10,15,31.
- Looking at features contribution we removed features 26, 27 and 29 which achieved a Gini score lower than 0.01 in all tree based models.

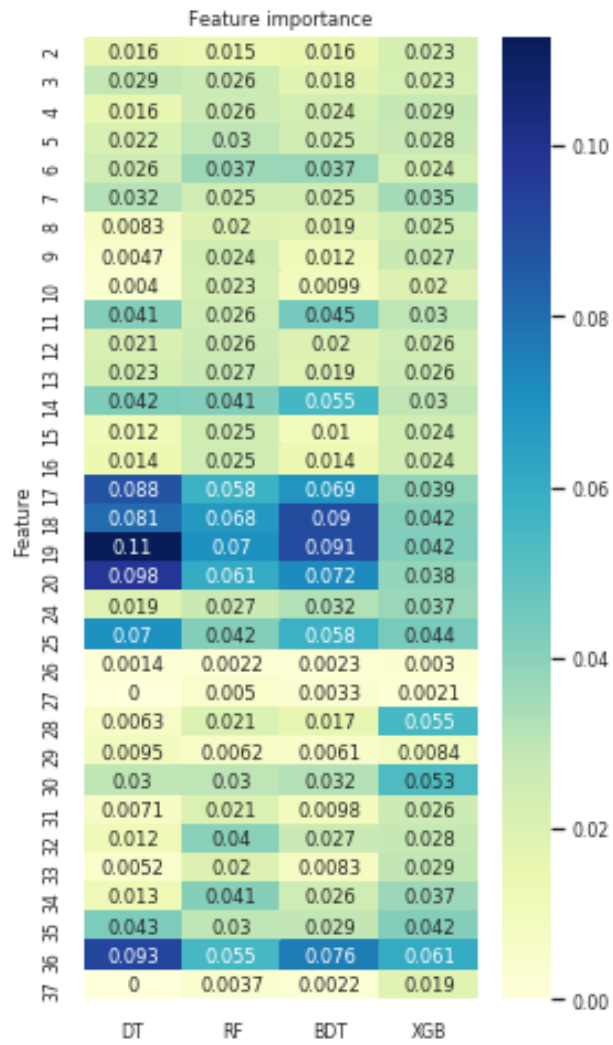


Figure D.8: Gini score of features for Model A

D.1.2 Hyper-parameters tuning

The selected parameters correspond to the first row of each table.

Table D.3: Top AUC values with Random Forest parameters Model A.1

AUC	n_estimators	max_depth	min_samples_leaf
0.839	100	20	2
0.838	200	20	2
0.838	200	20	5
0.838	500	20	2
0.837	500	20	5
0.837	400	20	2
0.837	500	10	2
0.836	500	20	2
0.836	400	20	4
0.836	500	10	5

Table D.4: Top AUC values with Bagging Decision Tree parameters Model A.1

AUC	n_estimators	max_depth	min_samples_leaf
0.800	500	2	2
0.798	200	2	6
0.794	400	2	1
0.791	500	2	6
0.790	300	2	2
0.787	300	2	1
0.787	400	2	6
0.785	200	2	2
0.785	200	6	1
0.785	200	2	1

Table D.5: Top AUC values with XGBoost parameters Model A.1

AUC	n_estimators	max_depth	learning_rate	reg_alpha	min_child_weight
0.847	100	6	0.01	0.001	1
0.847	100	6	0.001	0.001	1
0.847	100	6	0.0001	0.001	1
0.846	200	4	0.0001	0.1	1
0.846	200	4	0.001	0.1	1
0.846	200	4	0.01	0.1	1
0.846	200	4	0.1	0.1	1
0.846	200	10	0.0001	0.001	2
0.846	200	10	0.001	0.001	2
0.846	200	10	0.1	0.001	2

D.1.3 Model selection

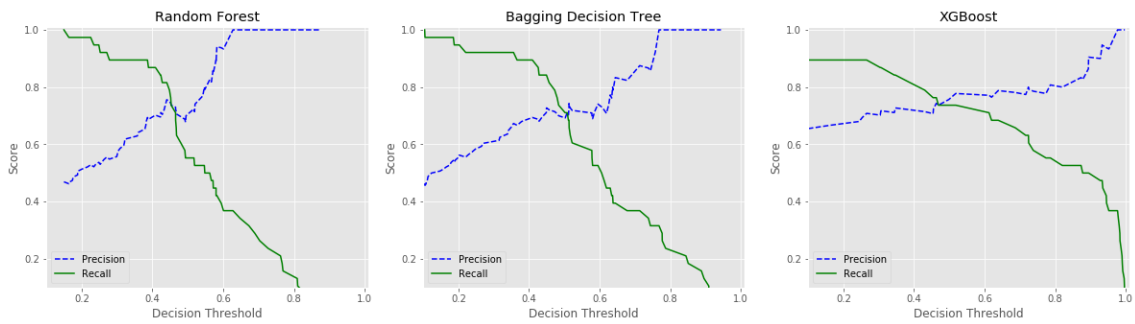


Figure D.9: Precision-Recall curves Model A.4

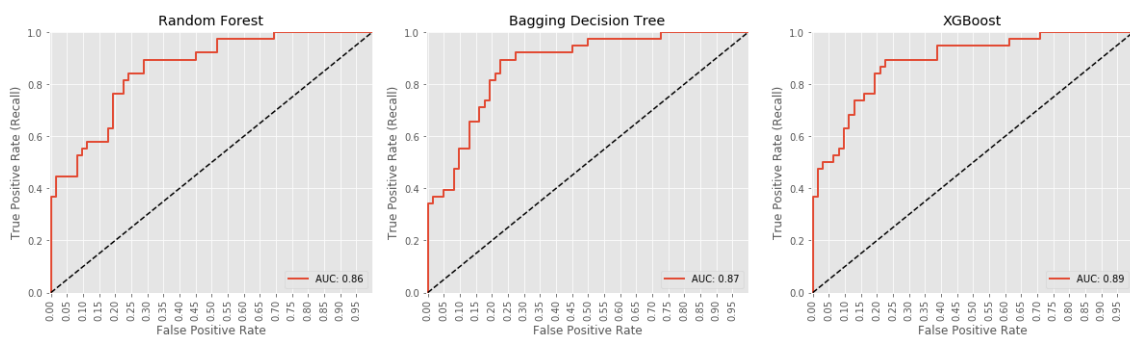


Figure D.10: ROC-curves Model A.4

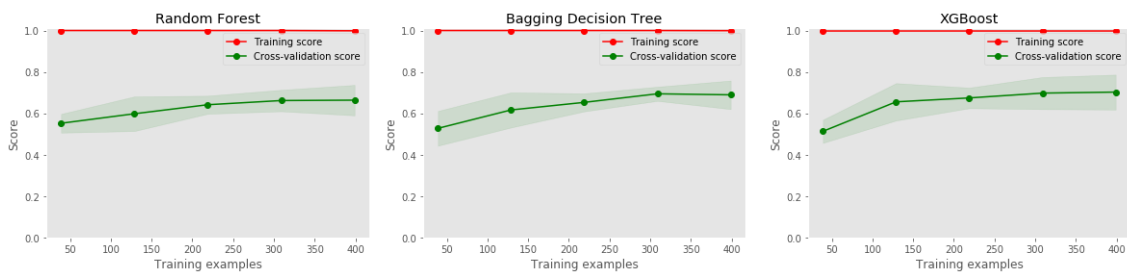


Figure D.11: Learning curves Model A.4 (Using F1-score)

D.2 Model B

D.2.1 Features selection

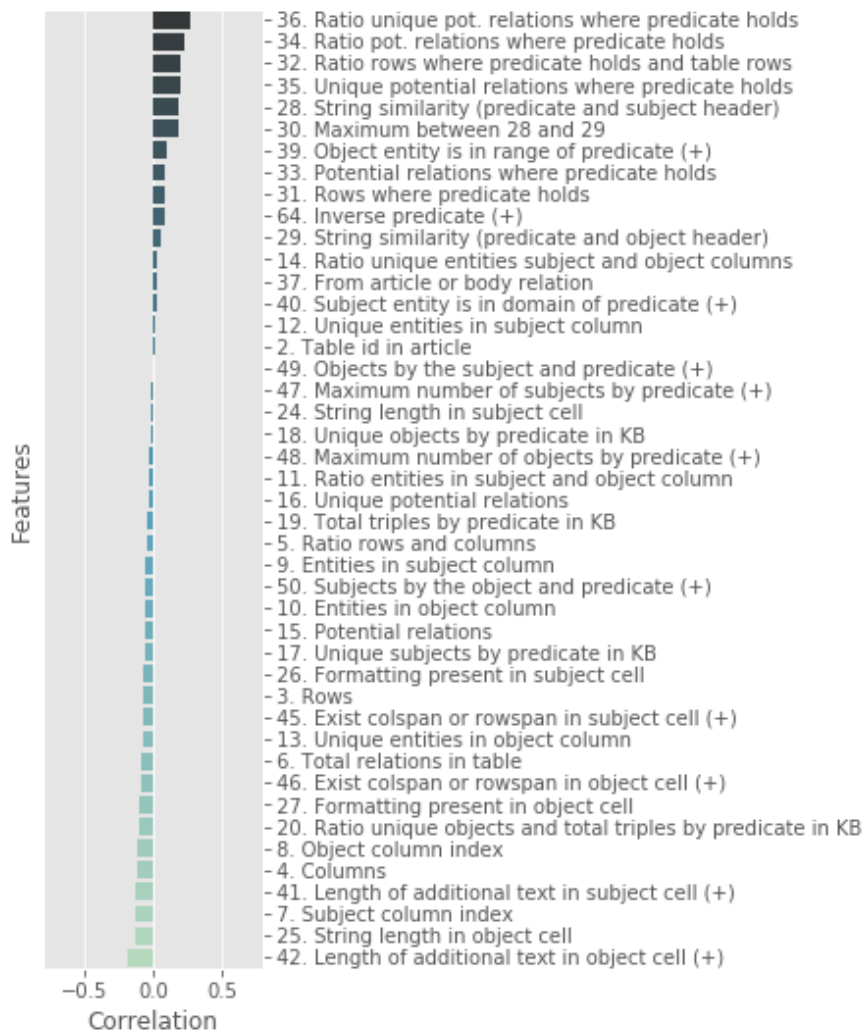


Figure D.12: Correlation between features and class (Model B, (+) indicates new feature)

Following the same process for selecting features for Model A, we obtained the top correlated features considering the new features proposed for this model and a threshold of 0.80. From the new features added we obtained two pairs of features with a high correlation; the values are presented in Table D.6. Features 17 and 18 represent the number of unique subjects and objects for a predicate respectively while features 47 and 48 indicate the maximum number of subjects by object and the maximum number of objects by a subject for a predicate. Both pairs of features are highly correlated; however they do not present the same values since the new features attempt to give information about how many objects and subjects are "allowed" for a predicate, then we also analyse the features contribution in Figure D.13.

Since we observed that the results of classification algorithms with the new features are better, and these features present a higher Gini score, we decide to remove, additionally to correlated features found analysed for the Model A, features 17 and 18 for this model.

Table D.6: Features with high correlation for Model B

Feature_1	Feature_2	Correlation
3	5	0.905
3	9	0.901
3	10	0.905
3	15	0.894
4	7	0.857
4	8	0.916
5	9	0.816
7	8	0.808
9	10	0.970
9	15	0.995
9	16	0.873
10	15	0.974
10	16	0.835
12	16	0.879
15	16	0.826
31	33	0.997
32	36	0.821
32	34	0.888
34	36	0.944
17	47	0.894
18	48	0.952

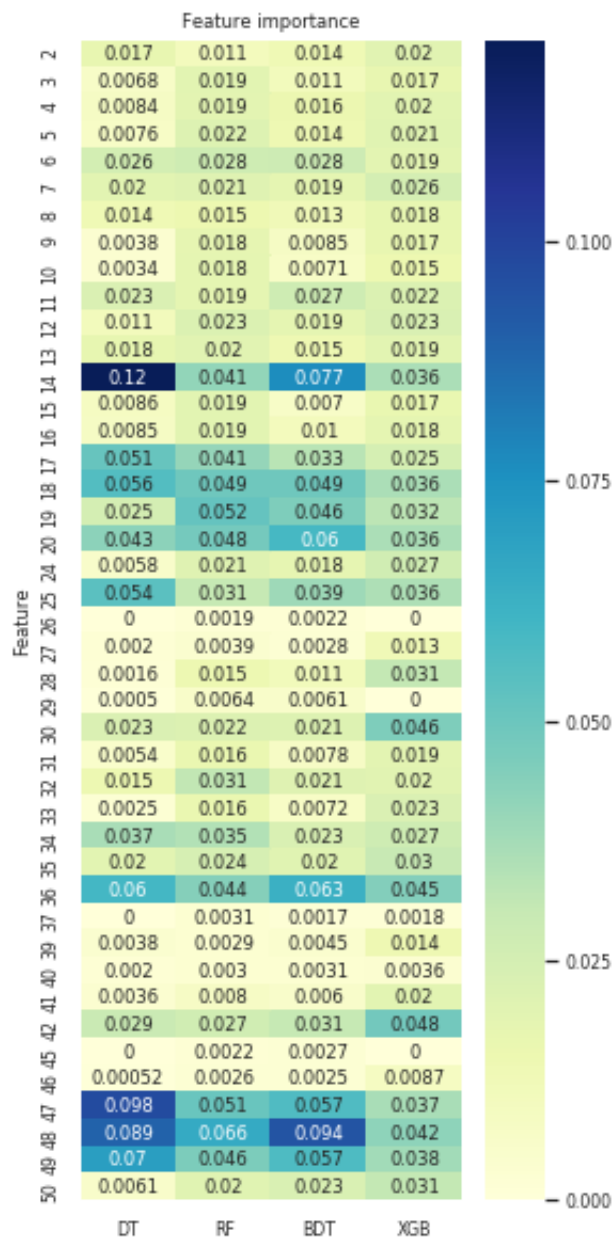


Figure D.13: Gini score of features for Model B

D.2.2 Hyper-parameters tuning

Table D.7: Top AUC values with Random Forest parameters Model B.1

AUC	n_estimators	max_depth	min_samples_leaf
0.865	200	10	2
0.862	500	10	2
0.862	300	10	1
0.861	400	10	1
0.861	500	10	1
0.860	200	10	1
0.860	300	10	1
0.860	200	10	4
0.859	300	10	2
0.859	100	10	2

Table D.8: Top AUC values with Bagging Decision Tree parameters Model B.1

AUC	n_estimators	max_depth	min_samples_leaf
0.800	300	6	2
0.795	200	20	1
0.795	400	8	1
0.795	100	6	4
0.790	400	20	4
0.790	500	8	2
0.788	200	8	6
0.788	300	10	2
0.788	500	6	1
0.788	400	6	2

Table D.9: Top AUC values with XGBoost parameters Model B.1

AUC	n_estimators	max_depth	learning_rate	reg_alpha	min_child_weight
0.878	100	10	0.0001	0.001	2
0.878	100	10	0.001	0.001	2
0.878	100	10	0.01	0.001	2
0.878	500	6	0.0001	0.0001	1
0.878	500	6	0.001	0.0001	1
0.878	500	6	0.01	0.0001	1
0.878	400	6	0.0001	0.0001	1
0.878	400	6	0.001	0.0001	1
0.877	400	6	0.01	0.0001	1
0.877	100	20	0.0001	0.001	2

D.2.3 Model selection

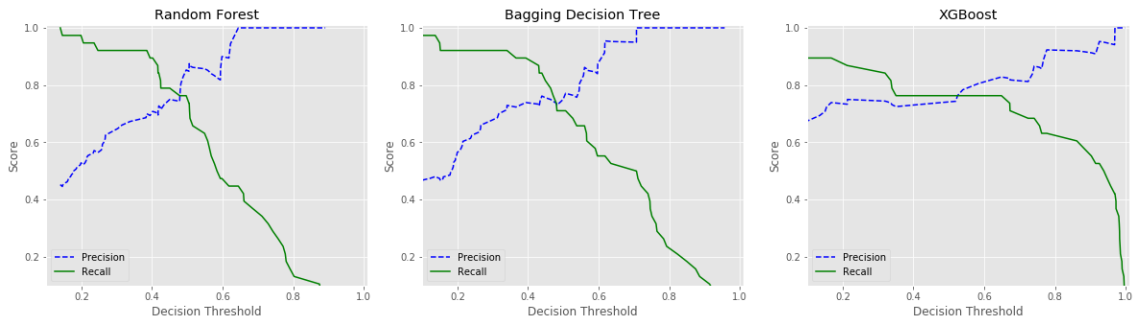


Figure D.14: Precision-Recall curves Model B.4

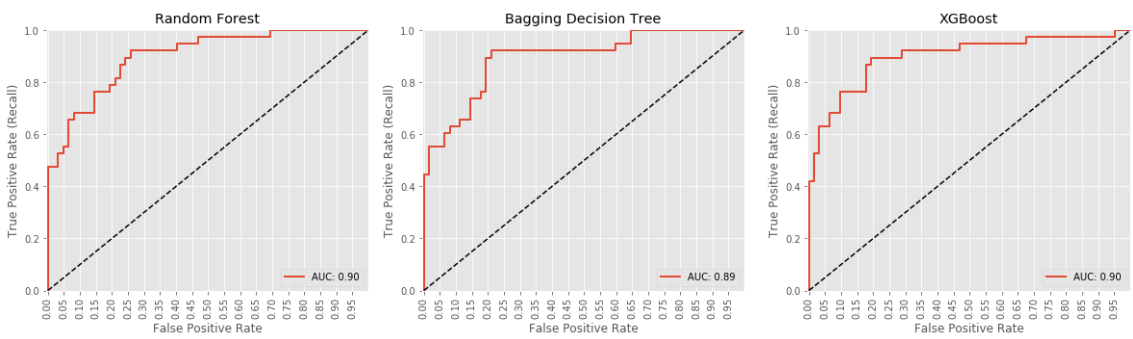


Figure D.15: ROC-curves Model B.4

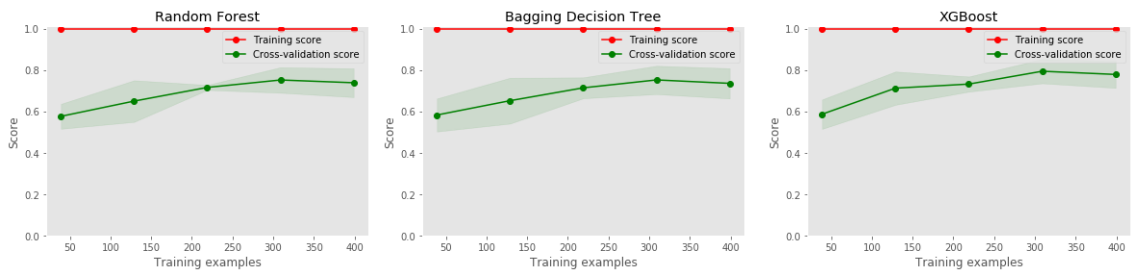


Figure D.16: Learning curves Model B.4 (Using F1-score)

D.3 Model C

D.3.1 Features selection

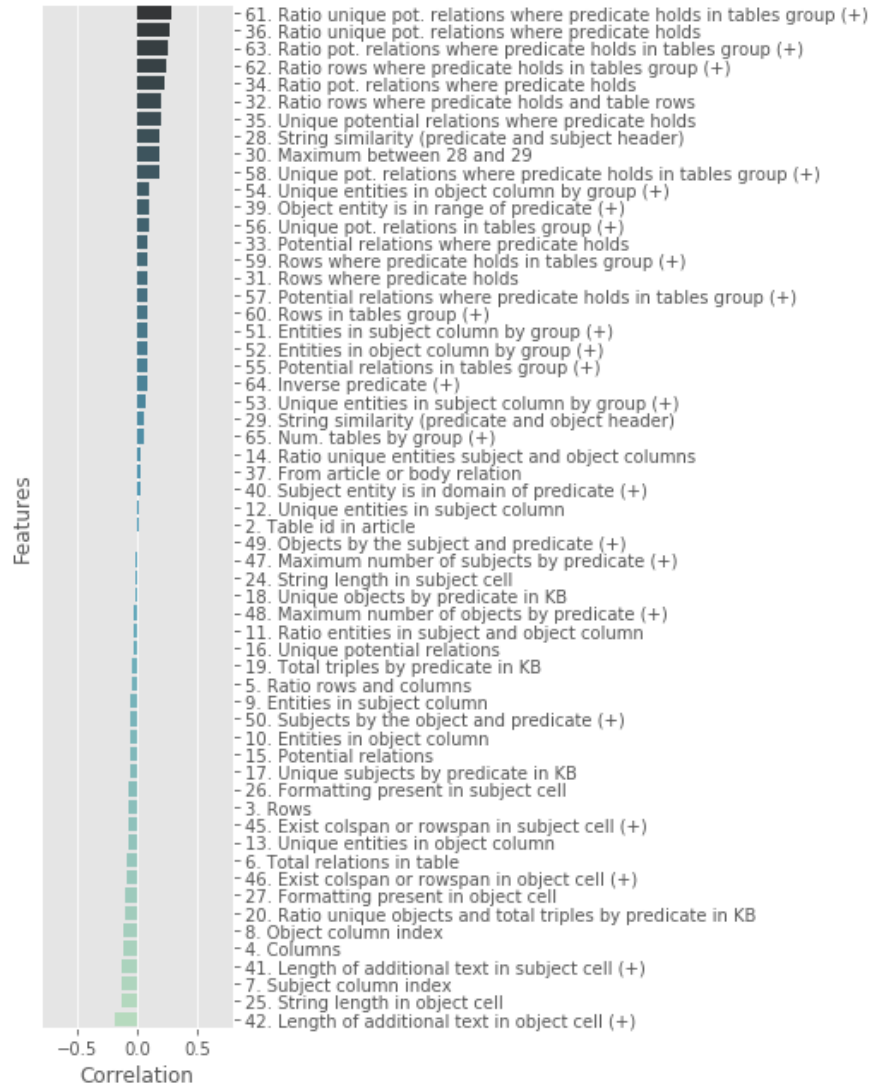


Figure D.17: Correlation between features and class (Model C, (+) indicates new feature)

The models were also validated with triples from I . We presented the top correlated features for these models in Table D.10.

We observe that many features extracted from groups of tables are highly correlated with each other (see Table D.10); for example the number of entities in subject and objects columns in the group of tables (Features 51, 52), the number of rows in the group of tables (Feature 60), as well as the number of relations in the rows of the group (Features 55, 56).

Table D.10: Features with high correlation for Model C

Feature_1	Feature_2	Correlation
32	62	0.806
34	63	0.885
34	61	0.827
36	61	0.890
36	63	0.865
51	55	0.998
51	52	0.956
51	60	0.933
52	60	0.985
52	55	0.968
52	65	0.911
52	56	0.860
53	55	0.889
53	65	0.813
54	56	0.957
55	60	0.946
55	65	0.933
56	60	0.872
56	58	0.799
60	65	0.927
61	63	0.939
62	63	0.828

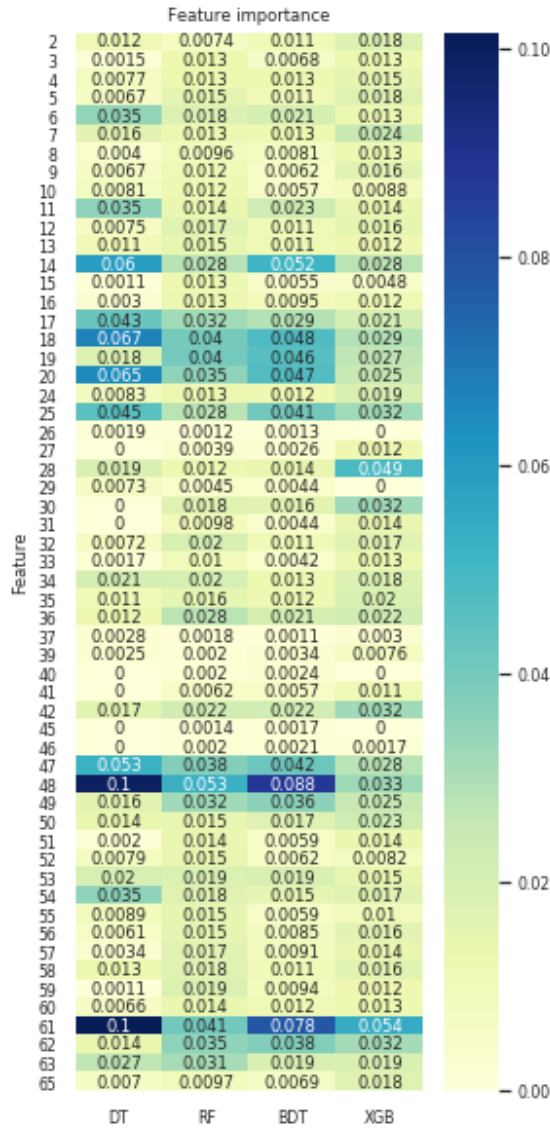


Figure D.18: Gini score of features for Model C

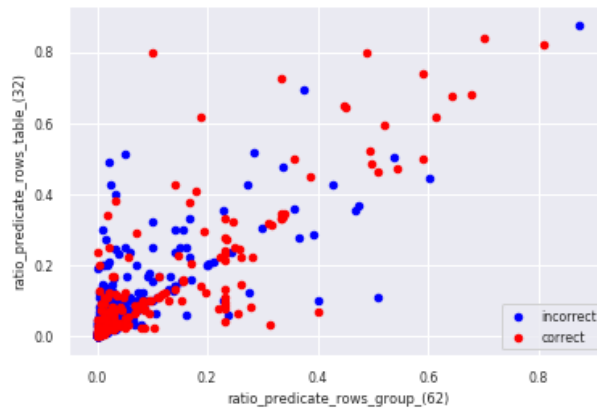


Figure D.19: Ratio of rows where predicate holds in table (32) and group (62)

We remove the features with higher mean correlation with all the features and also considering the minimum Gini score obtained applying the tree classification models (see Figure D.18).

Considering this analysis and the features analysed in previous models we removed features: 3,4,5,9,10,15,17,18,26,27,29,31,33,51,52,55,56 and 60.

We observed the correlation between the ratio of rows and relations where the predicate holds in the single table and the ratio in the corresponding group; these features are highly correlated with each other but they present a different tendency with respect to the classes, as we see in Figure D.19; therefore we did not remove these features.

D.3.2 Hyper-parameters tuning

Table D.11: Top AUC values with Random Forest parameters Model C.1

AUC	n_estimators	max_depth	min_samples_leaf
0.857	400	10	1
0.856	300	20	1
0.855	200	10	1
0.855	400	10	2
0.854	300	10	2
0.854	200	20	2
0.854	500	20	1
0.854	500	20	2
0.853	500	10	1
0.852	400	20	1

Table D.12: Top AUC values with Bagging Decision Tree parameters Model C.1

AUC	n_estimators	max_depth	min_samples_leaf
0.822	500	10	1
0.815	300	4	1
0.815	400	8	2
0.808	200	6	1
0.806	200	20	2
0.805	200	8	1
0.803	300	4	1
0.802	300	10	4
0.800	300	10	2
0.799	500	8	2

Table D.13: Top AUC values with XGBoost parameters Model C.1

AUC	n_estimators	max_depth	learning_rate	reg_alpha	min_child_weight
0.876	100	4	0.001	0.1	2
0.876	100	4	0.01	0.1	2
0.876	100	4	0.1	0.1	2
0.875	100	4	0.001	0.1	2
0.875	100	4	0.01	0.001	2
0.875	100	4	0.1	0.001	2
0.875	100	4	0.001	0.001	1
0.875	100	4	0.01	0.01	1
0.875	100	4	0.1	0.01	1
0.875	200	8	0.001	0.01	1

D.3.3 Model Selection

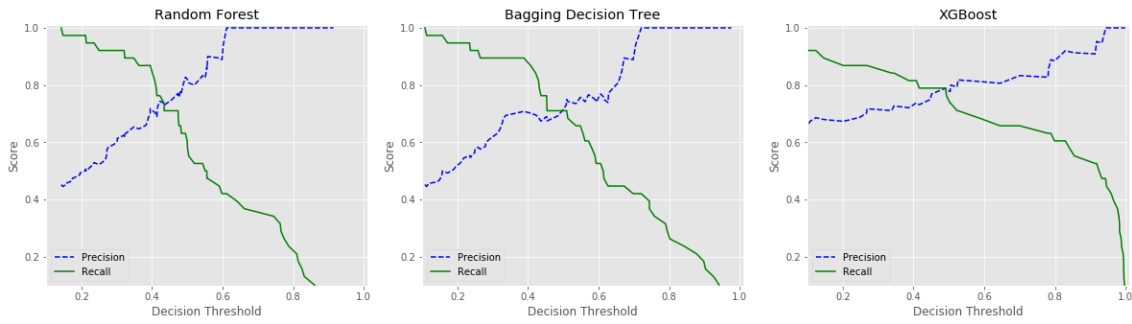


Figure D.20: Precision-Recall curves Model C.4

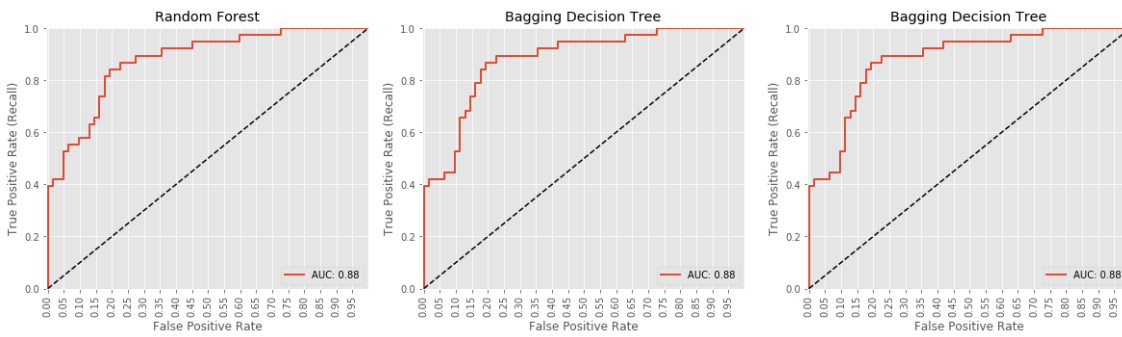


Figure D.21: ROC-curves Model C.4

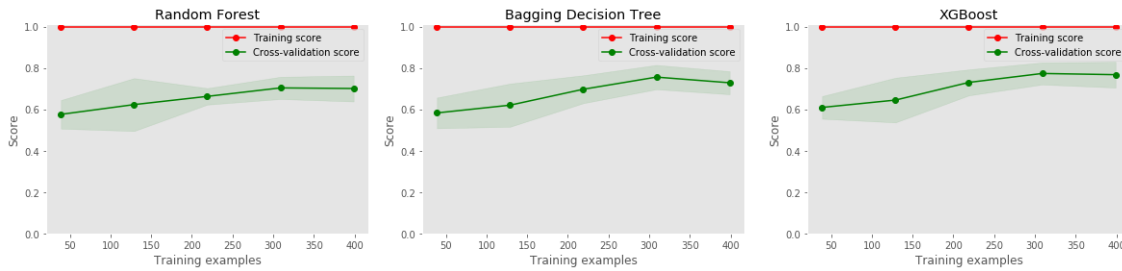


Figure D.22: Learning curves Model C.4 (Using F1-score)

D.4 Model D

D.4.1 Features selection

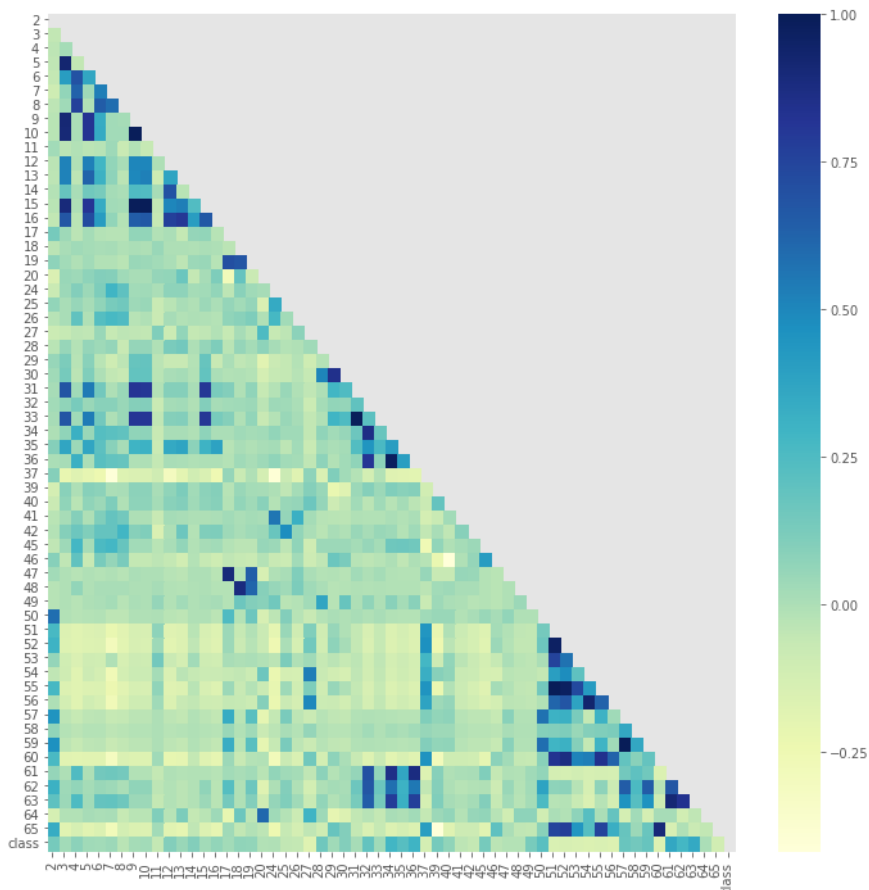


Figure D.23: Correlation Matrix with new proposed features

Following the same methodology used for selecting features for validating the models of set I , we apply features selection to the set of labeled triples from F considering all features.

The correlation matrix in Figure D.23 for this set of triples shows the same correlated features as for previous models I ; thus then we consider to remove the same features: 3, 4, 5, 9, 10, 15, 26, 27, 31, 33, 51, 52, 55, 56, 60, 33, 17, 18 and also remove those features with lower Gini score (see Figure D.25) in all models 37,41,42,45,46.

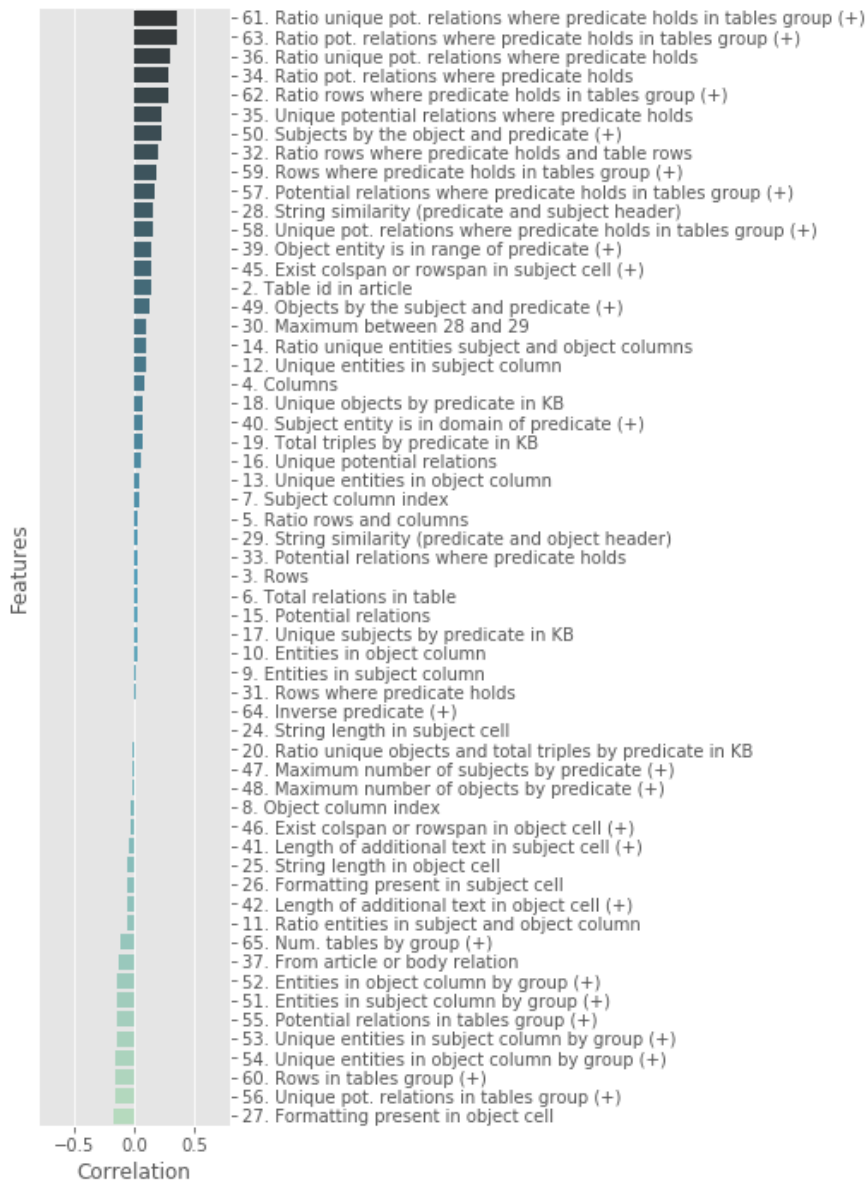


Figure D.24: Correlation between features and class (Model D, (+) indicates new feature)

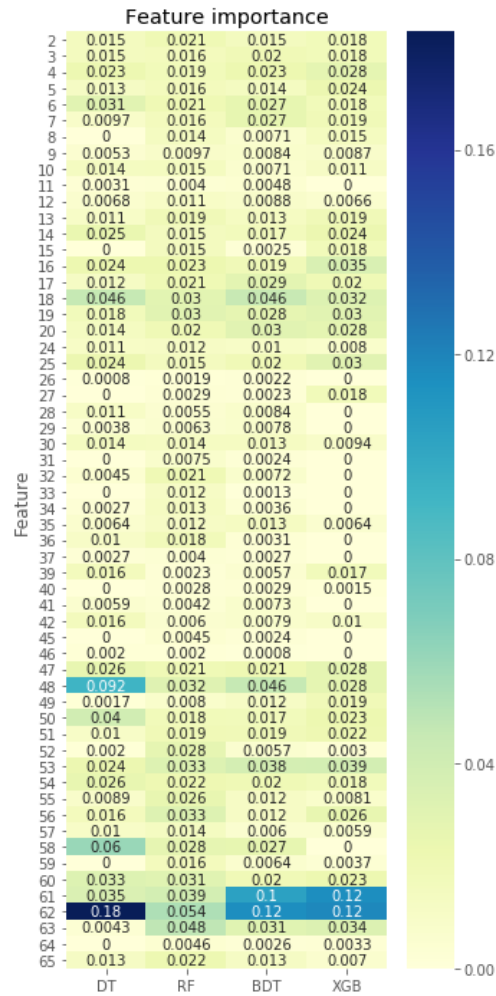


Figure D.25: Features importance for model D

D.4.2 Hyper-parameter Tuning

Table D.14: Top AUC values with Random Forest parameters Model D.1

AUC	n_estimators	max_depth	min_samples_leaf
0.858	500	10	1
0.858	300	20	1
0.857	400	20	1
0.857	500	20	1
0.857	100	20	1
0.854	200	20	1
0.854	300	20	2
0.853	500	8	1
0.853	200	8	2
0.853	300	10	1

Table D.15: Top AUC values with Bagging Decision Tree parameters Model D.1

AUC	n_estimators	max_depth	min_samples_leaf
0.705	400	20	4
0.699	200	20	6
0.699	200	6	6
0.697	100	4	1
0.697	200	40	6
0.695	400	6	6
0.694	100	4	4
0.692	100	6	2
0.690	300	4	4
0.69	500	6	4

Table D.16: Top AUC values with XGBoost parameters Model D.1

AUC	n_estimators	max_depth	learning_rate	reg_alpha	min_child_weight
0.869	500	10	0.0001	0.0001	1
0.869	500	10	0.001	0.0001	1
0.869	500	10	0.01	0.0001	1
0.869	500	10	0.1	0.0001	1
0.868	500	20	0.0001	0.0001	1
0.868	500	20	0.001	0.0001	1
0.868	500	20	0.01	0.0001	1
0.868	500	20	0.1	0.0001	1
0.867	400	10	0.0001	0.0001	1
0.867	400	10	0.001	0.0001	1

D.4.3 Model selection

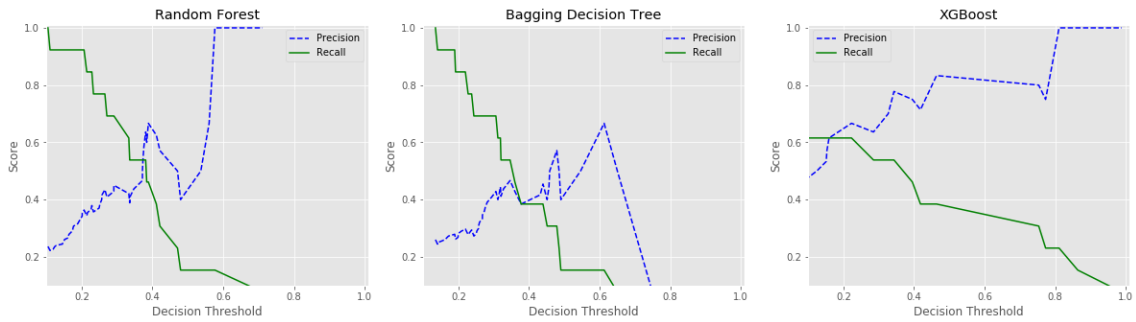


Figure D.26: Precision-Recall curves Model D.4

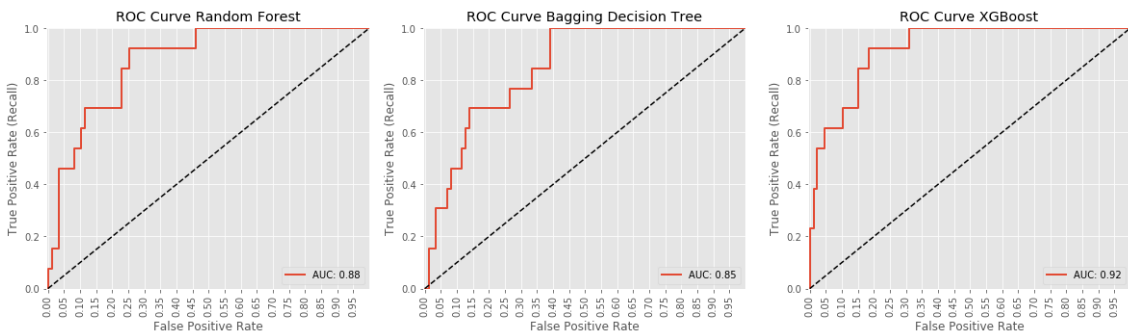


Figure D.27: ROC-curves Model D.4

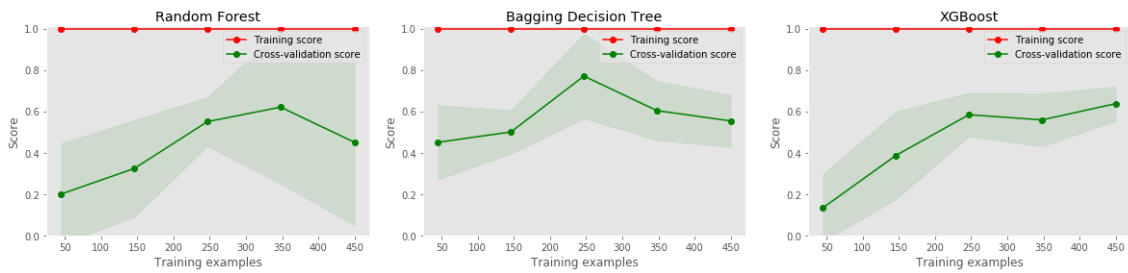


Figure D.28: Learning curves Model D.4 (Using F1-score)

D.5 Evaluation

D.5.1 Correct triples extracted

Table D.17: Examples of triples classified as correct to feed Wikidata

Subject	Predicate	Object
Q15531476 The_Eternal_Sea	P161 cast member	Q4800331 Arthur_Space
Q43788 Madison,_Wisconsin	P361 part of	Q1537 Wisconsin
Q7950076 WHBG	P1408 licensed to broadcast to	Q511935 Harrisonburg,_Virginia
Q2183999 FC_Kommunalnik_Slonim	P159 headquarters location	Q863576 Slonim
Q754132 1989–90_UEFA_Cup	P3967 final event	Q744369 1990_UEFA_Cup_Final
Q1946049 Zagórz	P131 located in the administrative territorial entity	Q1340594 Sanok_County
Q17495911 Sangeet_Singh_Som	P19 place of birth	Q200237 Meerut
Q468264 Candice_Dupree	P54 member of sports team	Q1274643 Phoenix_Mercury
Q16249654 Cut_Bank_(film)	P161 cast member	Q343510 Oliver_Platt
Q7819369 Tommy_Dowd_(baseball)	P413 position played on team / speciality	Q1142885 Outfielder
Q1066134 Prince_of_Persia: _Arabian_Nights	P123 publisher	Q48976504 Mattel_Interactive
Q1394498 M88_Recovery_Vehicle	P495 country of origin	Q30 United_States
Q860670 Rolfe_Kent	P1411 nominated for	Q2357620 Satellite_Award_for _Best_Original_Score
Q16197506 LSU_Tigers_baseball	P286 head coach	Q7152168 Paul_Mainieri
Q580122 1959_Tour_de_France	P710 participant	Q3434758 Robert_Cazala

Q6143618 James_Stevenson_(UK _politician)	P102 member of political party	Q7886824 Unionist_Party_(Scotland)
Q15043347 Million_Dollar_Arm	P750 distributor	Q191224 Walt_Disney_Pictures
Q35 Denmark	P150 contains administrative territorial entity	Q1748 Copenhagen
Q3695083 Cosetta_Campana	P1532 country for sport	Q38 Italy
Q7146275 CorbinDances	P27 country of citizenship	Q30 United_States