

UNIVERSITÉ DE TUNIS
INSTITUT SUPÉRIEUR DE GESTION



MASTER RECHERCHE

SPÉCIALITÉ
SCIENCES ET TECHNIQUES DE L'INFORMATIQUE DÉCISIONNELLE
(STID)

OPTION
INFORMATIQUE ET GESTION DE LA CONNAISSANCE (IGC)

Entity Linking in Web Tables Using Knowledge Graph

SAMIA KNANI

SOUS LA DIRECTION DE:

NADIA YACOUBI AYADI MAITRE ASSISTANTE, ISG TUNIS DIRECTEUR DU MÉMOIRE

2018-2019

Laboratoire SMART-LAB

Acknowledgments

God be praised for helping me do this work

I would like to express my deep gratitude to Doctor **Nadia Yacoubi**, my research supervisor, for her patient guidance, enthusiastic encouragement and useful critiques of this research work.

I would also like to offer my special thanks to my parents, **Farida** and **Abdlahfidh** for their support and encouragement throughout my study.

My special thanks are extended to my sisters and my brothers and to all my Friends for their help during my study period.

Samia

Abstract

The Web contains vast amounts of semi-structured data in the form of HTML tables found on Web pages which may serve for various applications. One prominent application, which is often referred to Semantic Table Interpretation, is to exploit the semantics of a widely recognized knowledge bases (KB) by matching tabular data, including column headers and cell contents, to semantically rich descriptions of classes, entities and properties in Web KBs. In this master's thesis, we focus on relational tables which are valuable sources of facts about real-world entities (persons, locations, organizations, etc.) and we propose a robust and efficient approach for bridging the gap between millions of Web tables and large-scale Knowledge graphs such as DBpedia. Our approach is holistic and fully unsupervised for semantic interpretation of Web tables based on the DBpedia Knowledge graph. Our approach covers three phases that heavily rely on word and entity pre-trained embeddings to uncover semantics of Web tables. Our experimental evaluation is conducted using the T2D gold standard corpus. Our results are very promising compared to several existing approaches of annotation in web tables.

Keywords: Linked Data, Semantic Table Interpretation, Entity Linking, Semantic embeddings

Contents

1	Basic concepts	4
1.1	Linked Open Data	4
1.1.1	Linked Open Data Principles	5
1.1.2	Resource Description Framework	5
1.1.3	The SPARQL query language	7
1.1.4	Knowledge Graphs	8
1.2	Web tables	11
1.2.1	Definitions	11
1.2.2	Classifications of Web Tables	12
1.3	Conclusion	15
2	Matching Web Tables to a Knowledge Graph	16
2.1	Introduction	16
2.2	Metadata information	16
2.3	Semantic annotation in Web Tables	17
2.3.1	Entity Linking	18
2.3.2	Column Type Prediction	20

2.3.3	Relation Extraction	20
2.4	Approaches for Web tables annotation	21
2.4.1	Search based approaches	21
2.4.2	Supervised Learning based approaches	26
2.5	Conclusion	27
3	Contributions	28
3.1	Problem Formulation	28
3.2	A Holistic Approach for Semantic Interpretation of Relational Web table	30
3.2.1	Pre-processing	31
3.2.2	Column Type Annotation	31
3.2.3	Entity Linking in Web tables	33
3.2.4	Columns Predicate Annotation CPA	42
3.3	Conclusion	46
4	Evaluation	47
4.1	Experiments	47
4.1.1	Datasets:	48
4.1.2	Knowledge graph	48
4.1.3	Wikipedia2Vec: Pre-trained model of Words and Entities Embeddings	49
4.2	Evaluation	50
4.2.1	Evaluation metrics	50
4.2.2	Experiment Results	50
4.2.3	Results of the Column Type Prediction task	51

4.2.4	Results of the Entity Linking task	52
4.2.5	Results of the Columns Predicate Annotation task	54
4.3	Comparison with other Approaches	55
4.4	Conclusion	56
5	Conclusion	57
	References	59

List of Figures

1.1	The Linking Open Data cloud diagram on 2019	6
1.2	RDF	7
1.3	Example of triples in the Turtle syntax	8
1.4	Difference between Document and Relational Tables	12
1.5	Web table classification	13
1.6	Layout table about Tunisie	13
1.7	Horizontal Web Table	14
1.8	Vertical Web Table	14
1.9	Matrix Web Table	15
2.1	Table Metadata	18
2.2	Steps of Entity Linking task	19
2.3	Column Type Prediction task	20
2.4	Relation Extraction task	21
2.5	Column Type Prediction task	23
3.1	Running example of the Column Type Annotation task	32

3.2	Approach workflow	34
3.3	Architecture of CBOW and Skip-gram models	38
3.4	Example of an Entity-Entity disambiguation graph;	41
3.5	A running example of the CPA task	44
4.1	The distribution of tables per category	48
4.2	The impact of using column types in our system	52
4.3	Comparison between the results of a simple DBpedia lookup and an improved lookup	53

List of Tables

1.1	Overview of common knowledge graphs (reproduced from (Paulheim,2017))	10
2.1	Comparison of web table annotation classes	22
3.1	An example of a Web table describing movie titles	31
4.1	Characteristics of the T2D gold standards.	48
4.2	Results of the three matching tasks	51
4.3	Results of the CTA task using different features	51
4.4	Variation of the local Similarity threshold	53
4.5	Variation of the number of seed nodes	54
4.6	Results of the Columns Predicate Annotation task	54
4.7	Comparison with methods focusing on the CTA task in Web tables	55
4.8	Comparison with methods focusing on the EL task in Web tables .	55
4.9	Comparison with methods focusing on the CPA task in Web tables	56

Listings

1.1	SPARQL query example	8
3.1	SPARQL query used in the Outgoing-Properties extract	44
3.2	SPARQL query used in the Incoming-Properties extract	45
3.3	SPARQL query used in the	45

List of Algorithms

3.1	Graph-based EL in web tables	40
3.2	Pseudocode to find the subject column	43

Introduction

Nowadays, vast amounts of information is available in structured forms like spreadsheets, database relations, and tables; all embedded in HTML Web documents. Among all of these semi-structured data sources, Web tables are considered as valuable sources of high-quality relational data. The Web contains approximately more than 154 million HTML tables of relational data (Cafarella et al., 2008), Wikipedia only containing 3.2 million high-quality tables. Recently, extracting the semantics of Web tables to produce machine understandable knowledge has become an active area of research (Zhang and Ziqi, 2014; Bhagavatula et al., 2015; Efthymiou et al., 2017; Kim et al., 2018). Among the different table types available on the Web, relational tables (also referred to as genuine tables) are of special interest. They describe a set of entities (such as persons, organizations, locations, etc.) along with their attributes. However, unlike tables in relational databases, these relationships are not made explicit in web tables; uncovering them is one of the main research challenges. The uncovered semantics can be leveraged in various applications, including table search (Nguyen et al., 2015), knowledge base construction, population (Zhang et al., 2020), and table completion (Zhang and Balog, 2019). Fortunately, a large number of Knowledge Graphs have been developed rapidly during the last ten years as multi-relational structured sources of knowledge. Knowledge Graphs (KG) (Paulheim, 2017) have widespread applications in information retrieval, text mining, and natural language processing. Cross-domain knowledge bases such as DBpedia (Lehmann et al., 2015), YAGO (Mahdisoltani et al., 2014), or the Google Knowledge Graph (Uyar and Aliyu, 2015) are used as background knowledge in a growing range of applications including the Semantic Interpretation of Web tables (Efthymiou et al., 2017; Bhagavatula et al., 2015; Zhang and Ziqi, 2014; Zhang, 2017; Fetahu et al., 2019).

However, in order to achieve this purpose, a number of tasks need to be implemented:

- The **Column Type Annotation (CTA)** task that identifies the most likely semantic type(s) in a given KG for the core column in a Web table.
- **Entity linking (EL)** is a key step towards uncovering semantics in Web tables by recognizing and disambiguating specific entities (such as persons, organizations, locations, etc.) and linking them to entities in Knowledge graphs.
- **Columns Predicate Annotation (CPA)** refers to the task of associating a pair of columns in a table with the relation that holds between them and/or extracting relationship information from tabular data and representing them in a structured format such as RDF triple (subject-predicate-object).

While large number of existing approaches focus on the task of EL in Web tables (Efthymiou et al., 2017; Bhagavatula et al., 2015; Luo et al., 2018). Few approaches focus on Column Type Annotation (Kim et al., 2018) and Relation extraction in Web tables (Chen et al., 2018a; Fetahu et al., 2019) even if they are crucial tasks for several applications such as Knowledge Base enrichment, completion and refinement. Although, semantic interpretation of Web tables is a crucial and complex process for many reasons: (i) The types of the entities described in a table are not known in advance, and the entities described may correspond to more than one type in the target of KG; however, the poor context offered by Web tables in comparison with texts makes the entity disambiguation in Web tables crucial. (ii) The size of both KGs and Web tables corpora is extremely large; (iii) KGs are not complete enough to cover all the entities of Web tables and novel entity discovery in Web tables is challenging (Zhang et al., 2020).

In this thesis, we propose a holistic approach for Web tables interpretation (annotation) by uncovering their semantics based on to the DBpedia knowledge graph as background knowledge. Thus, we propose a robust and efficient approach for bridging the gap between millions of Web tables and large-scale Knowledge graphs such as DBpedia. The main contributions of our approach are:

- An unsupervised and holistic approach for semantic table interpretation that covers all key tasks of Column Type Annotation, Entity Linking and Columns Predicate Annotation.

- We rely on Dbpedia knowledge graph as background knowledge for semantic interpretation of Web tables, but our approach can be generalised to any cross-domain knowledge graphs.

Outline

The master thesis is structured in four chapters organized as follows:

Chapter 1: Basic concepts introduces the fundamentals and theoretical concepts that our work relies on. This chapter presents the foundations of Linked Data as well as Web tables.

Chapter 2: Matching Web Tables to a Knowledge Graph. In this chapter, we discuss the task of semantic interpretation web tables; we present the state of the art of matching web tables to knowledge graphs.

Chapter 3: Contributions. This chapter describes our contributions and details the different steps of our Holistic Approach for Semantic Interpretation of Relational Web tables.

Chapter 4: Evaluation presents the experiments and results obtained by our proposed method. Furthermore, our results are compared and discussed taking with respect to similar methods for semantic interpretation in Web tables.

Basic concepts

Introduction

This chapter gives an overview of the basic concepts of our research work. The first section introduces the concept of Linked Open Data (LOD) and its fundamental principles: the standard Resource Description Framework (RDF) as the basic language for publishing structured data in the Linked Open Data, as well as the standard SPARQL query language, a protocol to query RDF data. The third section introduces the notion of the knowledge graphs and presents the most commonly investigated knowledge graphs in research works. The last section defines the concept of Web tables and their classification in different research works.

1.1 Linked Open Data

The Web is the most famous internet service composed of interconnected hyper-text documents which makes available a vast amount of data in different formats: text, images, videos and other multimedia elements. Facing the exponential volume of heterogeneous data available, search engines need background knowledge to meet users queries and enrich search results with structured knowledge. Recently, the Linked Open Data (LOD) effort community addressed this issue and makes data understandable by machines.

1.1.1 Linked Open Data Principles

With the emergence of the Semantic Web, the necessity of creating a Web of Data has increased and becomes an objective. In this way, Linked Data has been proposed as a paradigm for publishing and connecting structured data on the Web. More specifically, (Tim Berners-Lee, 2006) presented the four fundamentals of linked data:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards such as RDF, RDFS, OWL, and SPARQL.
4. Include links to other URIs, so that they can discover more things

Based on these principles, Linked Open Data (LOD) was introduced in (Bizer et al., 2011) as an open, interlinked collection of datasets. We can visualize the success of linked data from the growth of LOD Cloud which currently contains 1,239 datasets with 16,147 links. Figure 1.2 illustrates the last updated version of the LOD cloud¹ on March 2019. Some datasets like DBpedia has become the central and the highly interconnected dataset. In the next section, we will present some of these datasets.

1.1.2 Resource Description Framework

As introduced by the World Wide Web Consortium (W3C) recommendation (Carroll and Klyne, 2004), the Resource Description Framework (RDF) is a standard for sharing information in the Web. RDF framework adopts a generic graph-based data model to represent resources by triples in the following format:

`<subject> < predicate > <object>`

An RDF triple contains three components as depicted in Figure 1.1:

¹<https://lod-cloud.net/>

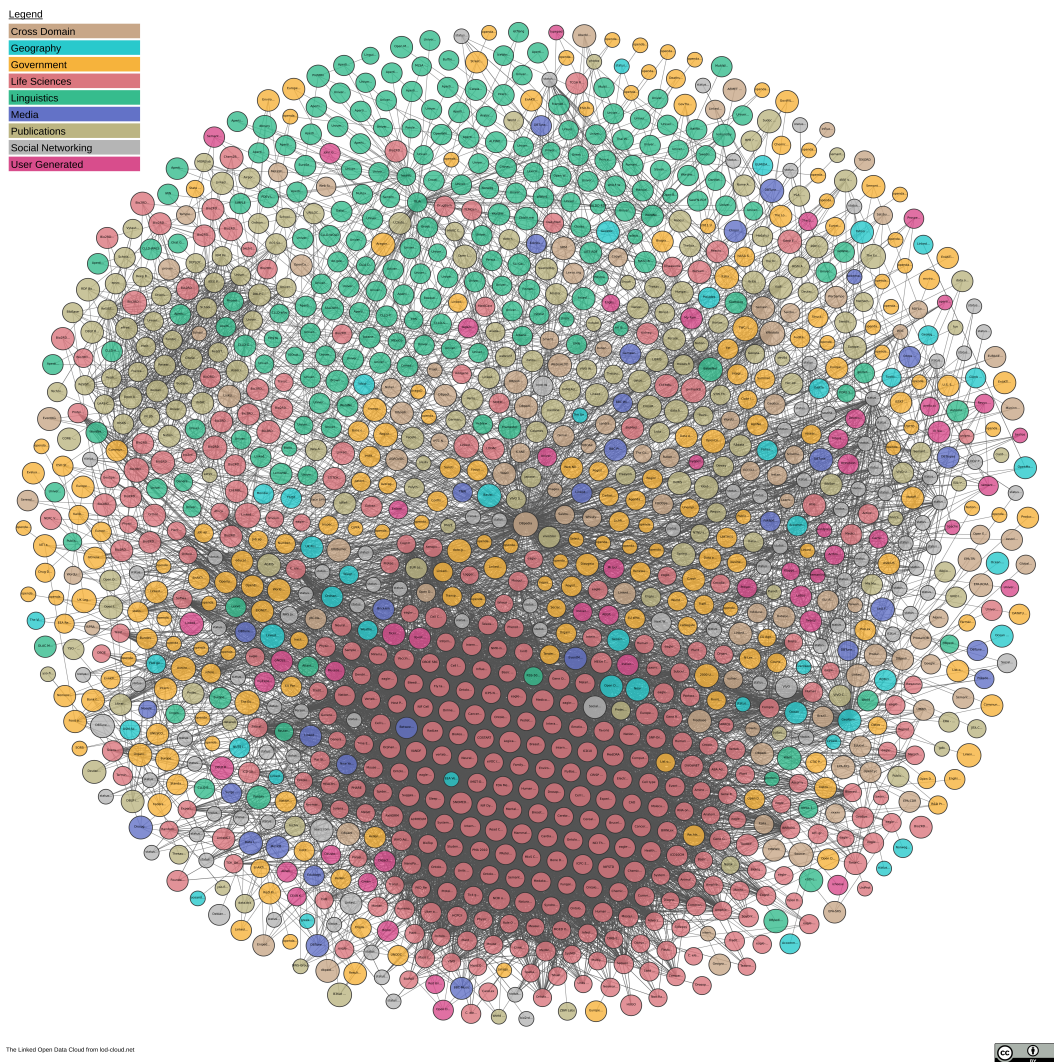


Figure 1.1: The Linking Open Data cloud diagram on 2019

- A **Subject** is an IRI or blank node; represents the resource to be described.
- A **Predicate** which is an IRI; represents a type of property applicable to the resource.
- An **Object** which is an IRI, a literal or a blank node; represents the value of the property for a subject

An RDF graph is a set of RDF triples, in which nodes of the graph are the set of subjects and objects, while edges are predicates that connect subjects to objects. In such a graph-based knowledge representation, entities, which are the nodes of

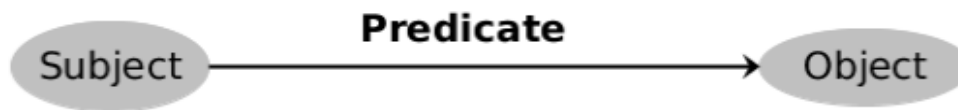


Figure 1.2: RDF

the graph, are connected by relations, which are the edges of the graph. Each entity is uniquely identified with an Internationalized Resource Identifier (IRI). RDF allows to assign a type or a class to entities by using the predefined RDF predicate `RDF:TYPE` (e.g. `(:Berlin, rdf:type, :city)`). More formally, we adopt the definition of (Ristoski, 2018):

Definition 1.1. *An RDF graph is a labeled graph $G = (V, E)$, where V is a set of vertices, and E is a set of directed edges, where each vertex $v \in V$ is identified by a unique identifier, and each edge $e \in E$ is labeled with a label from a finite set of edge labels.*

RDF is a data model and not a format. To this end, there are several machine readable serialization formats for publishing an RDF graph on the Web such as:

- RDF/XML
- Turtle
- N-Triples

A simple RDF description, serialized in Turtle, of the city Mannheim in Germany is shown in Figure 1.1 In this example, the first three rows specify the namespaces. While the last four lines defines 4 RDF triples that provide some descriptions about the resource `ex:Mannheim`. RDF allows to define assertions about a resource.

1.1.3 The SPARQL query language

The SPARQL Protocol and RDF Query Language (SPARQL) is the W3C standard language and protocol to query RDF data (W3C,). Triple patterns are triples where subject, predicate and/or object value can be unknown and replaced by a variable (traditionally identified by a question mark, e.g. `?v` for variable `v`). A

```

1 @prefix exr: <http://example.org/resource/> .
2 @prefix exo: <http://example.org/ontology/> .
3 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5 exr:Mannheim rdfs:type exo:City .
6 exr:Mannheim rdfs:label "Mannheim"@en .
7 exr:Mannheim exo:areaCode "0621" .
8 exr:Mannheim exo:country exo:Germany .

```

Figure 1.3: Example of triples in the Turtle syntax

triple pattern may match different triples in a given RDF graph, hence its name. For instance, the triple pattern $\langle ?s \ p \ ?o \rangle$ matches all triples with the predicate p , and triple pattern $\langle ?s \ ?p \ ?o \rangle$ matches any triple. SPARQL uses triple patterns to query RDF data. The triple patterns evaluation returns mappings between the query variables and their matching values.

A SPARQL query is a graph pattern (set of triple patterns) which must correspond to a sub-graph of the RDF graph queried to provide results. SPARQL queries consists of two main clauses: the first clause "SELECT" aims to specify the query form, while the second clause "WHERE" aims to specify a graph model to be matched by queried graph resources. Consider the example SPARQL query in Listing 1.1:

```

1 PREFIX dbo: <http://dbpedia.org/ontology/>
2
3 SELECT DISTINCT ?film_URI WHERE {
4 ?film_URI rdf:type dbo:Film> .
5 } LIMIT 10

```

Listing 1.1: SPARQL query example

Using DBpedia SPARQL endpoint², this query displays the first 10 film URIs available on DBpedia.

1.1.4 Knowledge Graphs

Recently, the term knowledge graph has been used more frequently instead of the term knowledge base, but no clear definition exists on the meaning of these

²<http://dbpedia.org/sparql>

terms. As indicated in (Paulheim, 2017), the term Knowledge Graph was coined by Google in 2012, referring to their use of semantic knowledge in Web Search (“Things, not strings”), and is recently also used to refer to Semantic Web knowledge bases such as DBpedia, Wikidata, Freebase, YAGO.

A knowledge graph:

- mainly describes real world entities and their interrelations, organized in a graph.
- defines possible classes and relations of entities in a schema.
- allows for potentially interrelating arbitrary entities with each other.
- covers various topical domains.

Knowledge graphs on the Semantic Web are generally based on linked data. It will be used to refer to a machine-readable representation of entities, their properties and taxonomy of classes in which they are organized. Furthermore, knowledge graphs are supposed to cover a wide range of topics that exist in the real world, and are not supposed to be restricted to only one domain (such as geographic entities). For the purpose of this master thesis, only large-scale, cross-domain knowledge graphs are discussed below, as only these knowledge graphs are useful for general profiling of the content of web table corpora. Table 1.1 summarizes the characteristics of the most popular knowledge graphs.

DBpedia

DBpedia is a very popular knowledge base which is extracted automatically from structured data in Wikipedia using the key-value pairs in the Wikipedia infoboxes. This multi-lingual knowledge base includes 125 languages versions which describe together 38.3 million things. Altogether, the most recent DBpedia release (2016-10)³ extracted from the updated Wikipedia dumps since October 2016, it covers 13 billion RDF triples extracted from (the English edition Wikipedia (1.7 billion), other language editions (6.6 billion) and the Wikipedia Commons and Wikidata (4.8 billion)). In addition, the DBpedia knowledge base is located in the center of the LOD cloud since it is highly interconnected with other datasets in the semantic web.

³<https://wiki.dbpedia.org/blog/new-dbpedia-release-%E2%80%93-2016-10>

Table 1.1: Overview of common knowledge graphs (reproduced from (Paulheim,2017))

Name	Instances	Facts	Types	Relations
DBpedia (English)	4,806,150	176,043,129	735	2,813
YAGO	4,595,906	25,946,870	488,469	77
Freebase	49,947,845	3,041,722,635	26,507	37,781
Wikidata	15,602,060	65,993,797	23,157	1,673
NELL	2,006,896	432,845	285	425
OpenCyc	118,499	2,413,894	45,153	18,526
Google Knowledge Graph	570,000,000	18,000,000,000	1,500	35,000
Google Knowledge Vault	45,000,000	271,000,000	1,100	4,469
Yahoo! Knowledge Graph	3,443,743	1,391,054,990	250	800

Freebase

Freebase is a collaboratively, public, editable knowledge base that enables users to add or modify the information in the knowledge base via APIs and a Web interface (Bollacker et al., 2008). Freebase based on the notions of objects, facts, types, and properties. Each object has at least one type and uses properties with these types to represent facts. With more than 50 million entities and 3 billion facts, Freebase considered one of the largest knowledge bases.

Wikidata

The Wikidata knowledge base has the same idea like Freebase. However, it is a collaboratively edited knowledge base successor of Freebase and it is operated by the Wikimedia foundation⁴. Currently, 76 million instances⁵ in Wikidata described in more than 350 languages. The most important aspect of Wikidata is that for each fact a provenance metadata can be included such as the source.

⁴<http://wikimediafoundation.org/>

⁵<https://www.wikidata.org/wiki/Wikidata:Statistics>

Google’s Knowledge Graph

Google Knowledge Graph (GKG) is a multi-lingual knowledge base, realized in 2012, used by Google to add semantic search functionality to its search engine. The GKG contains more than 18 billion facts over 570 million instances. Thus, It is currently supposed the largest knowledge graph and linked data set in the world, but not publicly accessible.

YAGO

YAGO is the abbreviation for Yet Another Great Ontology and similar to DBpedia, it is extracted from Wikipedia. However, the type hierarchy used by YAGO is based on WordNet and contains almost half a million classes. The latest version of YAGO contains more than 10 million instances and 120 million facts (Mahdisoltani et al., 2014). Unlike DBpedia, YAGO aims at an automatic fusion of knowledge extracted from various Wikipedia language editions, using different heuristics which are correct in about 95%.

1.2 Web tables

Currently, tables on web pages ”web tables” are considered as an important source of data since they represent their content in a structured way. Moreover, it is proved that Web tables exist in large quantity on the Web and cover several topics. For example The WebTables systems (Cafarella et al., 2008) extracted approximately 14.1 billion raw HTML tables from the English documents in Google’s main index, of which 154 million are in relational data (high-quality tables).

In this section, we study the notion of table in general and the key concepts associated with this notion as well as the different classifications of Web Tables mentioned in previous research in the field.

1.2.1 Definitions

Most of the research defines a table as a two-dimensional structure provides a compact visualization of data that makes it easier to search and compare data (Zanibbi et al., 2004). This compact form represents data in cells organized in

columns and rows.

(Lautert et al., 2013), define web table as follows:

Definition 1.2. (*Web Table*)

Web Tables WT is a two-dimensional tabular structure used on a web page. It is composed of an ordered set of x rows and y columns where each intersection between x and y determines a cell that has a certain value.

In different works, tables have very distinct characteristics and specification. The following figure distinguishes between two types of tables:

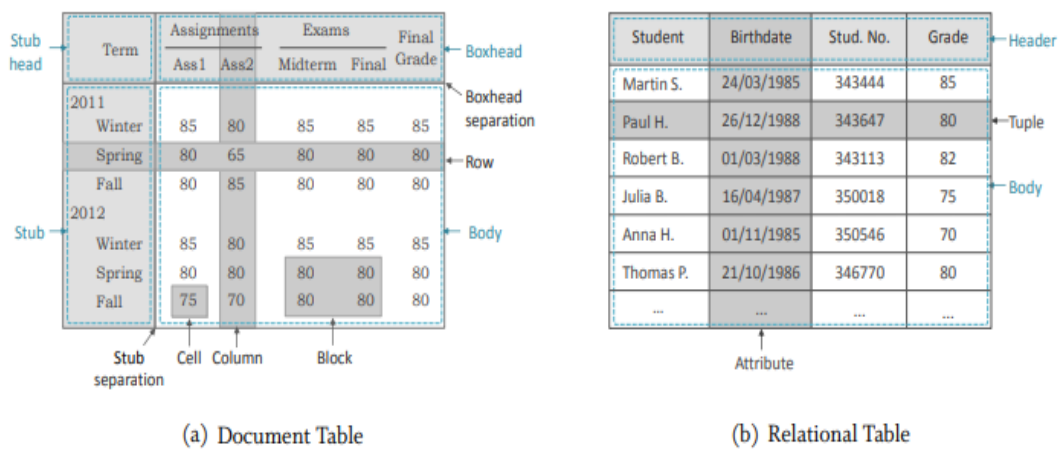


Figure 1.4: Difference between Document and Relational Tables

Figure 1.4 shows the complex layout of document tables and the simple layout of the relational database tables. Despite the structural complexity of Document Tables, humans still try to understand them, whereas it is very challenging to interpret these types of tables. In the other hand, Relational Tables that are set up for programmed handling. Its goal is to store data in simple and modifiable data structures.

For further details about the taxonomy of table types, the rest of this section analyzes and explains in detail the different classifications of web tables, and goes further by providing examples.

1.2.2 Classifications of Web Tables

In order to better understand the table classification task, we propose table type taxonomy similar to that proposed (Cafarella et al., 2008; Limaye et al., 2010;

Yakout et al., 2012). Thus, our proposal classifier illustrated in Figure 1.4 which aims to distinguish between two main types of tables: (a) Layout tables and (b) Content tables (Genuine).

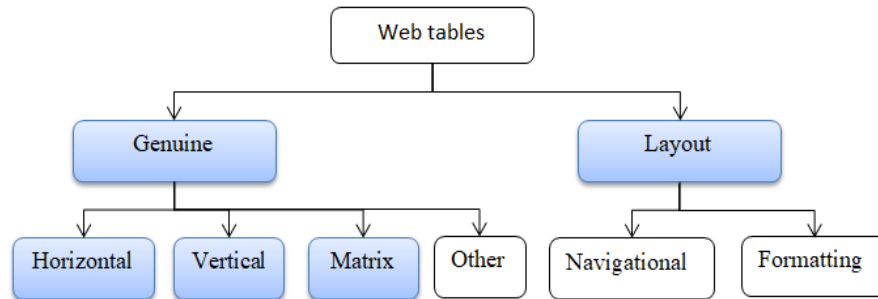


Figure 1.5: Web table classification

As shown in the Figure 1.5 we can distinguish between genuine and non-genuine tables, with more fine-grained classification schemes. Therefore, genuine tables are further divided into three categories: horizontal, vertical and matrix tables. Layout tables are divided into two categories: navigational tables and formatting tables.

Layout Tables:

Layout Tables represent the vast majority of HTML tables on the Web that used for layout purpose. This type of table often contains longer texts, images or hyperlinks and does not contain any relational content. This table category can be divided into formatting table, which used to organize certain elements on the web page, and navigation table, which is composed of cells organized for the navigation purpose. Figure 1.6 shows an example of layout table.

The screenshot shows a Wikipedia article for Tunisia. At the top right, it says 'Non connecté Dis'. Below that are navigation links: 'Article', 'Discussion', 'Lire', 'Modifier', 'Modifier le code', and 'Voir l'historique'. The main heading is 'Tunisie'. Below the heading, there is a star icon and the text 'Vous lisez un « article de qualité ».' The main text starts with 'La Tunisie (en arabe : تونس, *Tūnis* ^{Écouter?}, [tu.nis]), en forme longue la République tunisienne (en arabe : الجمهورية التونسية, *Al-Jumhūriyyah at-Tūnisīyyah* ^{Écouter?}), est un État d'Afrique du Nord bordé au nord et à l'est par la mer Méditerranée (1 148 kilomètres de côtes), à l'ouest par l'Algérie avec 965 kilomètres de frontière commune et au sud-est par la Libye avec 459 kilomètres de frontière. Sa capitale Tunis est située dans le nord-est du pays, au fond du golfe éponyme. Plus de 30 % de la superficie du territoire est occupée par le désert du Sahara, le reste étant constitué de régions montagneuses et de plaines fertiles.'

Figure 1.6: Layout table about Tunisia

Genuine tables:

A genuine table is a two-dimensional structured element of knowledge that contains coherent and potentially redundant information. It can formalize data in a Relational model. Unlike layout table which contains no knowledge, genuine tables are often rich in simple strings or numeric values. Below, we present three classes of genuine Web tables: horizontal, Vertical and Matrix web tables.

Horizontal Web Table:

A horizontal table presents in each row a specific object, a named entity, and the column provides the attributes that describe the entity. These types of tables often aim to compare or list elements with additional information. The figure 1.7 presents an example of horizontal web table.

Attributs		
Governorate	Population (2014)	Density
Tunis	1 056 247	3 053
Ariana	576 088	1 195
Ben Arous	631 842	830
Nabeul	787 920	283

Entities

Figure 1.7: Horizontal Web Table

Vertical Web Table:

A web table called Vertical if its tuples are arranged vertically. These tables often list multiple attributes for a series of similar entities. An example of Vertical web table is shown in figure 1.8.

Robert De Niro	
Born	August 17, 1943 New York, NY
Nationality	American
Occupation	Actor and director

Figure 1.8: Vertical Web Table

Matrix Web Table:

Matrix tables are widely used to present statistics and comparisons between the

data. In this type of table, the subject represents both the row and column headers; however the object is obtained by the intersection of the 2 subjects together. For example, Figure 1.9 show statistics for car accident; cell c_{33} (intersection of row 3 and column 3) corresponds to the number mechanical failure occurring in the 1980s.

Cause	1980s	1990s	2000s
Pilot error	26	27	30
Weather	14	10	8
Mechanical failure	20	18	24

Annotations in the figure: "Row header" points to the top row; "Column header" points to the first column; "Cell Object" points to the cell containing the value 18.

Figure 1.9: Matrix Web Table

Overall, the diversity of table types presented several challenges. Therefore, several classification approaches have been proposed in order to distinguish between genuine and layouts Web Table class. (Wang and Hu, 2002), for example, proposed a machine learning based approach for the detection of genuine Web tables which take into account Decision trees and Support Sector Machines (SVM) for the classification task.

1.3 Conclusion

In this chapter we introduced the concept of web tables and its main classifications. In addition, we have detailed various common knowledge bases. We also presented at the beginning of this chapter, the RDF framework, the SPARQL query language and linked open data.

Matching Web Tables to a Knowledge Graph

2.1 Introduction

In this chapter, we present the foundations of matching web tables to knowledge bases, which poses several challenges related to web tables as a data source. This chapter is organized as follows. Section 2.2 retrieves table metadata. Section 2.3, details each sub-task of the matching web tables to knowledge bases process by considering as input a tabular structure. In the last section, we present several related works on the semantic annotation of tables.

2.2 Metadata information

In contrast to other structured data sources like relational databases, tables lack any formal metadata or explicit schema (Cafarella et al., 2008). In our work, we are interested in relational web tables, a type of genuine tables that cover a collection of entities characterized by attributes. Indeed, the relational tables rich on data about different types of entities and contain multiple attributes of these entities. However, to work with web tables and exploit their high quality relational data, it is necessary to apply certain methods to retrieve different metadata information depending on the use case of the web table. For example, in our case, the

task of mapping web tables to knowledge bases requires the following metadata information: entity label column, attribute label row, column data types, and language information of the table.

In the following, we list the different types of metadata that are provided for the web tables. In fact, table metadata can include specific information such as table title, header row, label column, header column, column types, data type, table orientation (horizontal or vertical), and data language, they can also contain context information such as the URL of the web page, the text before and after the table (Surrounding context) as well as the title of the HTML page, which aim to recovering the semantics of a table.

Example 2.1. *Figure 2.1 illustrate the metadata retrieved for the table of cities in Tunisia. In this example, the second column represents the entity label column which contains the names of cities, while the rest of columns are of types numeric and string. The first row reserved for the attribute labels. Moreover, the table contains information in English. For the contextual information of the table, it can be determined using the URL, the title of the page web, the text before the table and the title of the table.*

In addition, the recovered metadata can also include the orientation of the table because a web table can be arranged vertically or horizontally. In fact, a relational table arranged horizontally if their rows represent the entities and their columns represent the attributes, and it is arranged vertically if it is the other way around (Lehmberg et al., 2016).

After introducing the general structure of a web table, in the next section we will present the details of the semantic annotation process.

2.3 Semantic annotation in Web Tables

Web table annotation is the process of interpreting the rows of Web tables and matching them to semantically rich descriptions of entities published in Web KBs. This section presents the three main tasks of semantics table interpretation which are as follows:

- **Column Type Prediction:** is task of annotating each column in the table with a class determining from the knowledge base that semantically describes the entities of the column (e.g., Person, Film, etc.).

https://en.wikipedia.org/wiki/List_of_cities_in_Tunisi

List of cities in Tunisia

This is the list of 350 cities and towns in Tunisia. In the list by governorate capitals are shown in bold.

List of most-populated cities

Rank	City	Population	Governorate
1	Tunis	638,845 ^{[1][2]}	Tunis
2	Sfax	272,801	Sfax
3	Sousse	221,530	Sousse
4	Ettadhamen ^[3]	142,953 ^[4]	Ariana
5	Kairouan	139,070	Kairouan
6	Gabès	130,984	Gabès
7	Bizerte	136,917 ^[5]	Bizerte
8	Aryanah ^[3]	114,486 ^[4]	Ariana
9	Gafsa	105,264	Gafsa
10	El Mourouj ^[3]	104,586	Ben Arous

Figure 2.1: Table Metadata

- **Entity linking** in tables which aims to annotate the values of cells in a web table with their corresponding entities from a knowledge base
- **Relation extraction:** aims to determining the relation between the label column and another column in the same web table from the knowledge base.

2.3.1 Entity Linking

Entity Linking (EL) is a key task in the semantic annotation process; it can facilitate several tasks such as knowledge base population, question answering, and information integration. For simplicity, let M denote a collection of mentions in a web tables. Each mention m in M is characterized by its context that comprises all mentions appearing with m in the same row and same column. Given a knowledge base KB containing a set of entities, the goal of EL is to match each mention m in M to its corresponding entity e in the KB . However, some entity mention in a web table may have not referring entity in the KB . In this case, the EL system mapped these mention as NIL and called unlikable mentions.

An EL system must disambiguate mentions in a web table and identify the mapping entity for each mention. It is in this sense that, a basic approach to entity linking has emerged that consists of a three mains steps, as shown in Figure 2.2.

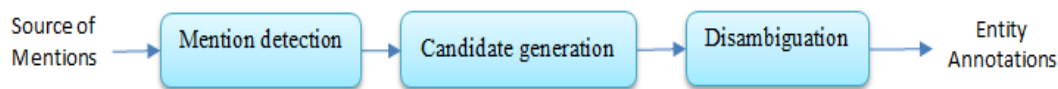


Figure 2.2: Steps of Entity Linking task

Mention detection: also known as extractor or spotter, it is the first step of the entity linking task that aims to detect terms or phrases from which links to entities should be made. Generally, majority of entity linking systems have based their works on a large dictionary of entity surface forms to detect mentions. Indeed, if the text value considered does not correspond to any entry in the dictionary, it will not be considered as a mention. There are also approaches that rely on Named entity Recognition NER techniques in the areas of Natural Language Processing (NLP) to detect text value that can refer to mentions. Moreover, to recognize named entities, it is often desirable to apply certain other pre-processing steps such as capitalization, POS tagging, removal of stop words, phrase chunking (Cheng and Roth, 2013).

Candidate generation: also called searcher since it is based on the lookup method (Hachey et al., 2013). At this stage, a set of candidate entities is generated from the Knowledge Base for each mention detected in the first step. The candidate generation step is often considered as a ranking problem; since at this step the number of candidates for each mention can be reduced in order to keep only candidates highly linked to the mention. Therefore, selecting fewer candidates will reduce execution time and make the disambiguation task easier.

Disambiguation: The last step, which is the most important step and the heart of the entity linking process; for each mention, select an entity or none from the set of candidate entities generated in the previous step. In other words, to resolve the ambiguity, select the entity that makes the most common sense with the target mention. In literature, disambiguation task can rely on prior importance of entities, contextual similarity between mention and entity as well as the coherence among all entities determined using networks structure such as graph. Indeed, most of graph-based approaches adopt collective strategies based on the context similarity between the mention and the entity (Han et al., 2011).

2.3.2 Column Type Prediction

An entity column in a web table is a list of mentions. The Column type prediction task aims to matching the common type of these mentions with a semantic type extracted from KB classes. For example, the first column of table in figure 2.3 composed of “Tunis”, “Sfax”, “Sousse” and “Kairouan” is annotated with `dbo:City` class of DBpedia. There are several recognized approaches that allow

City	Population	Governorate
Tunis	638,845	Tunis
Sfax	272,801	Sfax
Sousse	221,530	Sousse
Kairouan	139,070	Kairouan



City: <http://dbpedia.org/ontology/City>

Figure 2.3: Column Type Prediction task

the column annotation task, such as: Entity search (Mulwad et al., 2010) basing in the idea that each mention in the column mapped to a list of classes and then PageRank-based method is used to affect a score for the entities’ classes, from which the class with the highest score considering as the label class. There are also approaches that determining the column type based on strategies like majority voting (Zwicklbauer et al., 2013; Venetis et al., 2011). Some other studies like (Chen et al., 2018b) use semantic embedding and Convolutional Neural Networks (CNN) based method to automatically trains machine learning models to predict the column type.

2.3.3 Relation Extraction

Relation Extraction refers to the task of matching pair of column in a web table to property in the knowledge base. Figure 2.4 presents an example of a city table annotate with predicate from knowledge base. The table contains cities, their populations and their governorates. Therefore, the “`dbo:City`” label column related to the “`dbo:Population`” column with the relation “`dbo:populationTotal`”. In the same way, the relation between the label column and the “`dbo:Governorate`” is “`dbo:isPartOf`”.

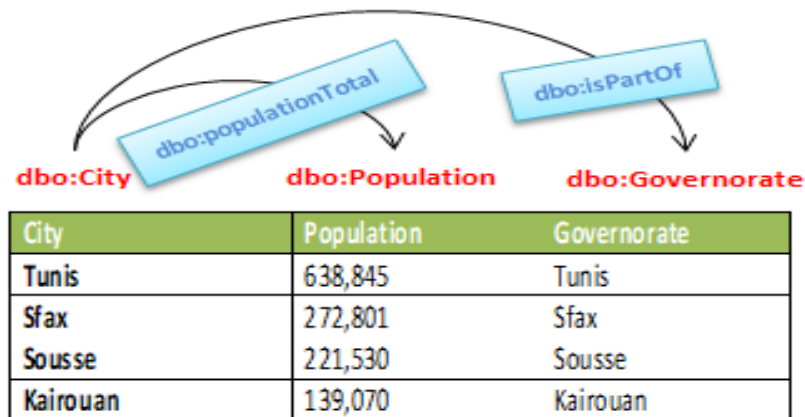


Figure 2.4: Relation Extraction task

Several existing approaches that mapping pair of columns in web table with relations from a knowledge base. Most of these approaches combine the results of entity linking and column type prediction tasks as well as the similarity measure to extract relations between columns (Nguyen et al., 2019; Thawani et al., 2019).

2.4 Approaches for Web tables annotation

The web contains a large variety of tables which can be effectively utilized to provide and collect useful information across multiple domains. (Cafarella et al., 2008) showed that out of 14 billion high-quality HTML tables in the Google crawl, an estimated 154 million HTML tables of relational data. Interpret the semantics of these web tables to produce machine-understandable knowledge has become a widely recognized research area and used in a wide variety of applications. In this section, we will discuss recently proposed methods in the area of semantic table annotation. Therefore, these approaches can be classified into two classes: Search based approaches and Supervised Learning based approaches. Table 2.1 describes these two classes of approaches.

2.4.1 Search based approaches

A majority of recent works based their approach on keyword lookup method. In other words, these approaches search for a particular keyword using a lookup

Table 2.1: Comparison of web table annotation classes

Approach		Cell Annotation	Column Annotation	Relation Annotation	Execution	KB used	Corpus used
Lookup-based	TableMiner	X	X	X	Iterative	DBpedia	Limaye, IMDB, MusicBrainz
	Efthymiou et al. 2017	X	X		Hybrid	Wikidata	T2D gold standard, Limaye gold standard, Own standard extracted from Wikipedia
	T2K Match	X	X		Iterative	DBpedia	T2D Gold Standard
	MTab	X	X	X		DBpedia	Sem-tab challenges datasets
Supervised Learning	TabEL	X			Iterative	YAGO	An existing Web table corpus (created by Limaye)
	DSL		X			Generic	T2D Gold Standard

method with a public API of the target KB, like (Zhang and Ziqi, 2014; Efthymiou et al., 2017; Zhang, 2017) or a manually created search index like (Zhang et al., 2013). Generally these approaches are limited to exact matches and regular expression queries.

TableMiner:

TableMiner (Zhang and Ziqi, 2014) and its improvement TableMiner++ (Zhang, 2017) are semantic web tables annotation tools based on the lookup method using the public query APIs of DBpedia and Freebase for the candidate generation process. These two incremental approaches characterized by an efficient bootstrapping annotation method used to effectively matching columns and cells to entities in the knowledge base. Therefore, authors starting by annotating the cells of each table column. Then, they use the results returned in the previous step to derive the column class. Afterward, the results obtained from the annotated class make it possible to improve the previous annotations of the cells and to reduce further annotations. Overall, the process repeats until the final optimal annotation cell and column is obtained.

MTab:

(Nguyen et al., 2019) presents MTab, which combines the voting algorithm and the probability models to solve two major problems: (a) DBpedia lookup does not usually get relevance entities for non-English queries and (b) mapping cell values to corresponding values in a KG is less effective because the corresponding value in KG is rarely equal with a query value.

Actually, MTab consists of seven steps as shown in the framework illustrated

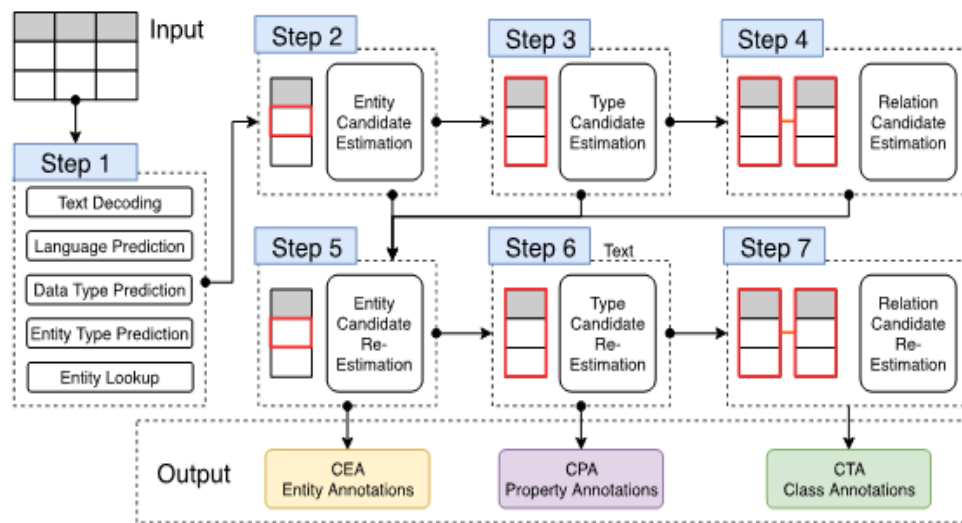


Figure 2.5: Column Type Prediction task

in figure 2.5 These steps can be summarized in three parts:

- The first part (step 1) focus on data preprocessing such as: Text decoding, Language prediction, Data type prediction, Entity type prediction and Entity Lookup parameters.
- In the second part; first (step 2) for each cell, ranking list of candidate entities is estimated based on multiple lookup methods. These entities are scored based on their ranking. After scoring, these scores are normalized and associated as probabilities of entities. Second, (step 3) columns are categorized to Numerical columns and Textual columns. For numerical columns, authors use EmbNum to find relevance relations, then, they use DBpedia ontology to infer types based on “Domains” of relation. For textual Columns, the candidate type is determined using the candidates types of numerical columns, entity lookup types for all column cells, types of SpaCy entity type for all

column cells and the Normalized Levenshtein distance between table header and DBpedia classes. Thus, for each candidate type a normalized probability is associated. Third, (step 4) to estimate the relation candidate between two columns, authors calculate the relation scores of two cells in the same row and then aggregate these scores for all rows.

- In the last part, (step 5) authors re-estimate entity candidates for table cells and select the entity with highest estimation score as annotation. (step 6 and step 7) They also re-estimate type and relation candidate with majority voting based on the results of cell annotation.

T2K Match:

(Ritze et al., 2015) proposed T2K Match, a specialized web table to knowledge base matching system which performs entity-level and schema-level matching .In other words, T2K Match links web tables to classes, rows to entities and columns to properties from a knowledge base.

- First of all, T2K Match selects the most unique text column to be the entity label column i.e., the most distinct column that includes the names of the entities described in the table.
- Afterwards, similarity measures are determined between the Web Tables and DBpedia in order to correspond between the cells of the table, the entities and the properties of the knowledge base. Then these matches are used to determine the majority class of the entities in the web table.
- Last, entity and property correspondences are revised iteratively and the classes that do not belong are deleted to obtain a final mapping.

Efthymiou et al. 2017:

(Efthymiou et al., 2017) adopt an unsupervised hybrid annotation method based on three alternative approaches for matching entities whose contextual information may vary from poor (in Web tables) to rich (in KBs):

- **Lookup-based method:** For each cell in the label column of the web table, this method perform several lookup service on an index created from label and description properties of all entities in the target KB and from the

returned candidates, a unique entity is mapped to each mention in the web table based on the minimal entity context provided in the web table, frequent types as well as the binary relation between entity described in a the table row and entity mentioned in the same row. However, if no results were returned, search for any entity with a similar label that having the extracted relation with entity in the same row.

- **Entity Embeddings Method** is an instance of global disambiguation techniques family, assuming that the entities that appear in sentences tend to form coherent sets with respect to the topic. This assumption also applies to entities described in tables, where columns are usually strongly typed and hence coherent at least with respect to types and topics.

Over all, consider only columns with text values, and regard each string value as an entity mention e . First, disambiguate the entities of all rows, using the contents of the label column cells as mentions and using the rest of row as context. Second, create a disambiguation graph of all the candidates entities for all the entities mentions. Third, add a directed edge weighted with normalized cosine similarity between each pair of entities not candidates for the same mention. Subsequently, create an assignment for each node by applying a weighted PageRank algorithm to calculate the relevance of each node and finally, select the nodes with the highest score from the set of candidates for each mention.

- **Ontology Matching Method:** Here, each row of the table describes a real-world entity and each column represents a property, with the exception of the label column, which defines the class of the table. All the entities in the table are instances of this class. During a first scan, identify columns with entity references and perform a small number of lookups in FactBase using the first few values from these columns. In a second scan, create a new instance of the table class for each row, whose property values are the cell contents of this row for the respective column. Finally, Call an ontology matching tool with the table ontology and the DBpedia ontology after blocking (a pre-processing step of candidate mappings selection), as input, and return the mapping results.

Authors also proposed an hybrid method for Web table annotation. They exploited the benefits of combining Lookup-based method and Entity Embeddings Method, as following:

- **Hybrid I** : If FactBase lookup method provides a mapping for the entity of a row, then this hybrid method keeps this mapping. Otherwise, it uses the annotation provided by the embeddings for this row, if one exists.
- **Hybrid II** : Same as Hybrid I in inverse order, i.e., using the embeddings first, before Lookup-based method.

2.4.2 Supervised Learning based approaches

Recently, works has tended to apply supervised learning to the semantic table annotation process where supervised task infers outputs data from labeled training data. However, the majority of these works is introduced as "semantic labeling" (Pham et al., 2016; Ramnandan et al., 2015) and generally focus on the Named Entity column annotation.

DSL:

DSL (Domain-independent Semantic Labeler) (Pham et al., 2016) is a tool for precise annotation of columns using similarity measures and machine learning techniques. In this system, the first step is to select a set of candidate classes from the ontology and training data which is a table corpus with the columns already labeled. Afterward, a set of similarity metrics can be applied to these annotations along with the column cell values to obtain feature vectors. Authors exploits 5 similarity metrics which are: attribute name similarity, standard Jaccard similarity, TF IDF cosine similarity, distribution similarity and histogram similarity. After calculating the similarity for each candidate classes, 2 classifiers (Logistic Regression and Random Forests) are trained per column class in the training data set to choose the best classifier for semantic labeling. However, Logistic Regression achieves higher performance than that of Random Forests.

TabEL:

TabEL is a supervised machine learning approach for Semantic table interpretation proposed by (Bhagavatula et al., 2015). This system takes the problem of annotating tables as an entity linking task which aims to match each value cell in the table to the concept of ontology. Hence TabEL based on three main steps described as follows:

- **Mention identification:** select potential mentions that can be linked to their referent entities in the knowledge base by calculating the prior probability for a given cell and a concept estimated from hyperlinks on the Web and in Wikipedia.
- **Entity candidate generation:** For each potential mention, a set of most related candidate entities is generated also using thus calculated probabilities.
- **Disambiguation:** employs a pre-trained Iterative Classifier with several features that ranks the candidates by the maximum likelihood and jointly disambiguates all mentions in a given Web table.

TabEL relies on a greedy approach which is time consuming and hard to implement. Indeed, the TabEL system outperforms all EL systems for texts, authors do not conduct any experimentations with EL systems dedicated for tabular structures. Therefore, the source code of TabEL is not yet released and available for testing.

2.5 Conclusion

In this chapter we introduced the metadata information of a Web Table. Further, we presented the three main tasks of semantics table annotation : Entity Linking, Column Type Prediction and Relation Extraction. Finally, we have listed several related works and we have classified these works into two types of approaches: Search based approaches and Supervised Learning based approaches.

Contributions

Introduction

In this chapter, we introduce our novel approach for Semantic Table Interpretation (STI) for relational tables. It performs all the three matching tasks continuously: column type detection, entity linking and relation extraction task. Our approach is an unsupervised method which leverages the context of Web tables to better capture their semantics. Our approach exploits mainly distributional vector representations of words and entities combined with a collective strategy to infer disambiguate and elicit the semantics behind a Web tables. This chapter consists of two section. The section 3.1 introduces a problem formulation of the STI process. Section 3.2 explains in further details our STI approach.

3.1 Problem Formulation

The semantic table interpretation task takes as input a relational web table and a target knowledge base (KB), and generally performs three sub-tasks as follows:

1. **Entity linking (EL)** is the task of matching mentions (phrases) of the cells of a web table to their referent entity in a given KB.
2. **Column type identification** is the task of associating to a given tables's column a KB type of entities. In our approach, we rely on DBpedia ontology

concepts for column type annotation.

3. **Relation extraction** is the task of associating a pair of columns in a table with the KB relation or property that holds between each pair of entities in a given row of two columns.

In our work, we focus on semantic interpretation of relational tables. Typically, each row in a relational table describes a real-world entity, while each column contains the value of the corresponding property/predicate in a given KB. The KB contains information related to these entities and values. Hence, it is essential to understand the overall theme and context of a given web tables for correctly matching web tables data on to the KB.

For any relational table $T = (M, H)$ as found on the Web, where $M = \langle m_1, m_2, \dots, m_l \rangle$ is a sequence of column mentions, $H = \langle h_1, h_2, \dots, h_n \rangle$ is a sequence of heading labels. Each mention m_i in M is characterized by its surface form and its local context $Cxt(m_i)$. Given a knowledge graph K containing a set of entities $\langle e_1, e_2, \dots, e_j \rangle$, a set of properties/classes $\langle c_1, c_2, \dots, c_k \rangle$ and a set of properties/predicates $\langle p_1, p_2, \dots, p_l \rangle$. While entity linking aims to identify and assign each potential mention m_i in cells of T to a correct entity e_j in K , class matching and relation extraction aims to match tables with properties of K (c_k and p_l , respectively).

Before mentioning the different matching steps of our approach, we examine the challenges of semantic annotation of web tables:

Lexical Variations. An entity often has several forms of mention (surface forms), like abbreviations (United States of America vs. USA), shortened forms (Osama Bin Laden vs. Bin Laden), alternate spellings (Osama vs. Ussamah vs. Oussama), and aliases (Osama Bin Laden vs. Sheikh Al Mujahid).

Entity Ambiguity. is the most important challenge to the EL problem: single mention can be matched to multiple KB entries. For instance, we can find 21 persons named "Adam Smith" in DBpedia.

The types of the entities mentioned in a table are not known beforehand and they can correspond to none or to several types in the knowledge graph.

The columns to be used to check for matches may differ from one table row to another.

The surface form of a mention in a Web table and its corresponding entity in a given KB, may significantly differ.

In recent years, a set of systems for the interpretation of Web tables has been developed. Most of these systems focus on a specific task, e.g., (Efthymiou et al., 2017; Bhagavatula et al., 2015; Luo et al., 2018) on the entity linking, the method of (Kim et al., 2018) on the property matching and the approaches introduced by (Chen et al., 2018b; Fetahu et al., 2019) on the class matching task. Few approaches focus on all the three matching tasks like TableMiner+ system (Zhang, 2017) and T2K match system (Ritze et al., 2015).

Previous works in Named Entity Disambiguation (NED) are generally based on the local contexts of the entity mentions. In recent years, some of the tabular annotation systems mainly use the coherence between candidate entities to improve the efficiency of the disambiguation task (Zwicklbauer et al., 2016) (Phan et al., 2018). These approaches are well known as collective because they aim to jointly resolve several mentions by linking them to their associated entities in a knowledge base.

Based on this new idea, our proposed approach for semantic tables interpretation adopts a collective strategy similar to (Zwicklbauer et al., 2016) and takes into account the richness of the context to improve the quality of the web tables matching process.

3.2 A Holistic Approach for Semantic Interpretation of Relational Web table

In this Section, we describe our approach for semantic annotation in web tables using the DBpedia Knowledge Graph. It performs all the three matching tasks continuously: Column Type Annotation(CTA), Entity Linking in Web tables (EL) and Columns Predicate Column (CPA) tasks. Our approach exploits mainly vector representations of words and entities on all three matching tasks and based on collective strategy which combines local and global features to infer entity linking decisions.

Rank	Title	Year	Director's
1	The godfather	1972	Francis Ford Coppola
2	Citizen Kane	1941	Orson Welles
3	Casablanca	1942	Michael Curtiz
4	Lawrence of Arabia	1962	David Lean
5	dr. Strangelove or: how I learned to stop worrying and love the bomb	1964	Stanley Kubrick
6	Apocalypse now	1979	Francis ford Coppola
7	The godfather: part ii	1974	Francis ford Coppola

Table 3.1: An example of a Web table describing movie titles

3.2.1 Pre-processing

This section performs some pre-processing steps that are applied on the web tables and aims to prepare the data inside the table. Indeed, this step aims to normalize and to clean the string values of data in order to avoid errors related to word ambiguity and to obtain good results in the KB lookup step. In the first instance, web tables contain values that can include HTML tags like the entity used to represent a non-breaking space " ," these HTML tags need to be cleaned. Furthermore, we remove special characters like parentheses, punctuation or slashes and additional whites-paces from cells in web tables and we transform text into lowercase.

3.2.2 Column Type Annotation

Column Type Annotation (CTA) aims to associate to each table's column a semantic type/category of the KB. In our approach, it is the task of matching the dominant type of cells in a target column to a recognized knowledge graph concept. For example, the column 2 of Table 3.1 is annotated with *dbo : Film* and *dbo : work*, two classes of the Dbpedia Ontology. The CTA task is crucial for the remaining two table annotation tasks, namely entity linking (section 3.2.3) and Columns Predicate Annotation (3.2.4).

In our work, we perform the CTA task by combining KB lookup and word embeddings representation. Hence, our CTA task combines several features ex-

tracted from the context of the Web table. CTA Features cover table metadata such as table name, header of columns and content cells (the set of mentions exist in the column of a Web table).

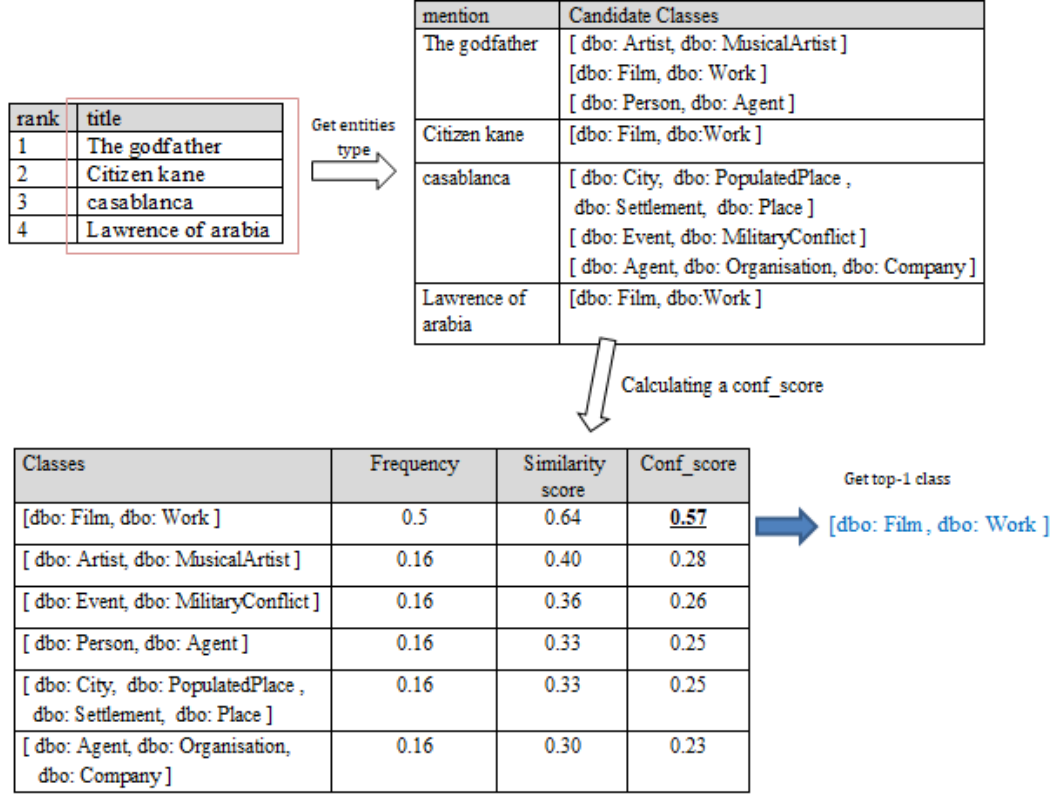


Figure 3.1: Running example of the Column Type Annotation task

First, we rely on a KB lookup method to associate to each mention of a given column a set of candidate classes collected from the DBpedia ontology. Hence, candidate classes are ranked according to a majority vote.

For each candidate class, we calculate a confidence score depending on three features: the column context which is represented by of all mentions in the same column, the column header name, and the majority vote score estimated based on the frequency of occurrences of each candidate class. We combine frequency, lexical and semantic features in the following simple formula:

$$Conf_Score_c = \alpha * \left(\frac{freq(c)}{nb_C} \right) + \beta * CC_{score} + \gamma * lev(hn, clabel) \quad (3.1)$$

Where $\frac{freq(c)}{nb_C}$ is the majority vote of each candidate class determined by the frequency of this class divided by the total number of candidate classes, lev is the

levenshtein similarity that represents the syntactic similarity between both column header name and DBpedia ontology class surface form and CC_{score} is the semantic similarity between the column header name (hn) (if it exists) and the candidate class label ($clabel$).

$$CC_{score} = \begin{cases} Cos_Sim(hn, clabel) & \text{if header is available} \\ Cos_Sim(ml, clabel) & \text{else.} \end{cases} \quad (3.2)$$

Voting with majority can contribute to a highly confident prediction of CTA. Referring to the example of the table 3.1, the KB lookup method for the mention "Casablanca" should return $dbo : City$ DBpedia ontology class as semantic type instead or in addition to $dbo : Film$ class. Nonetheless, the $dbo : City$ concept will be eliminated by majority vote score according to our confidence score. Note that the semantic similarity determined by the cosine similarity between both vector representations of the column header cell and the candidate class. However, only relying on the header cell results is insufficient because the header is not always available and arbitrary terms can be used. Therefore, if there are no headers, CC_{score} calculated using the cosine similarity between the mention label (ml) and the class surface name ($clabel$), after representing each cell value in the target column with a vector using word embeddings. Finally, the candidate class with the highest score will be selected as the best column type.

3.2.3 Entity Linking in Web tables

The Entity Linking (EL) task aims to match each mention of a table cell with KB entities. The first step allows to associate to each table mention a set of candidate entities. The second step disambiguate entities by combining a graph-based collective strategy and distributional vector representations (a.k.a. embeddings) to select the most suitable entities. Figure 3.2 summarizes the different steps of our EL task.

Candidate entities generation

The EL task starts with a candidate generation step. Thus, for each mention in a table cell, we need to determine an initial set of candidate entities from the given

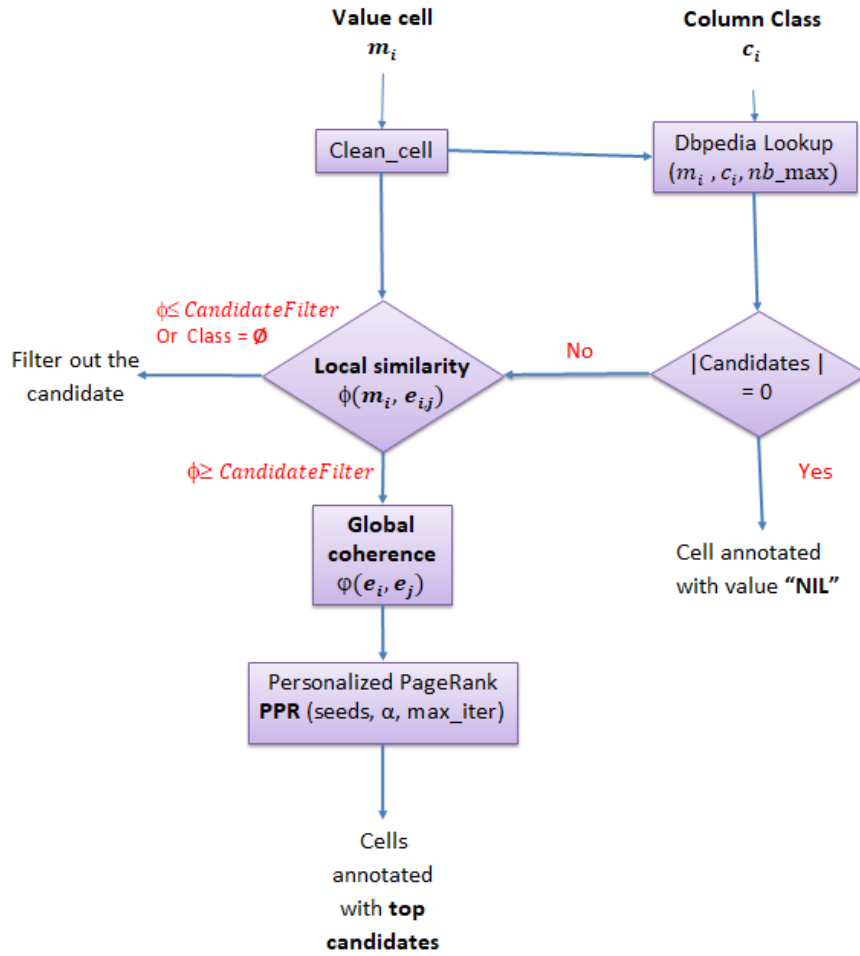


Figure 3.2: Approach workflow

KB. Since the candidate set can be large, we apply a feature-based ranking procedure based on contextual and string similarities calculated between each mention m_i and each entity $e_{i,j}$ in the candidate set E_{m_i} , in order to filter the most potential candidates and reduce the number of candidates. Hence, we compute a score for each candidate entity based on the following three similarity values:

- **Semantic similarity** between a mention m_i and an entity $e_{i,j}$ calculated as the cosine distance between their vector representations as word and entity embeddings which are n-dimensional vectors of words representing entities in the target KB.
- **Contextual information** between mention context and the description of the candidate entity extracted from the target knowledge base, the context of a

mention m_i represents all the mentions that exist in the same row and column of the target mention.

- **String similarity** represents the syntactic similarity between the string value of the mention m_i and the `RDFS:LABEL` value of the entity $e_{i,j}$.

Thus, we keep only the candidate entities having a similarity score exceeds the *CandidateFilter* threshold. Once the candidate entities have been filtered using the similarity metrics, a set of candidate entities is associated for each mention m_i .

Running Example: To illustrate the detail of the candidate generation step, table 3.1 serves as an example of a movies web table¹ that contains a list of films with their titles, release year and director, as well as a ranking attribute. The header row gives the attributes labels of the described entities. Each row in the table describes a real-world entity (e.g., the second row describes The godfather film), and each column contains the value of the corresponding property, e.g., (“Year”,1972), (“Director”,Francis Ford Coppola). First, we predict the class of each column table as mentioned in section 3.2.2. Then, we rely on the DBpedia lookup service which can be used to search for DBpedia URIs by associated keywords. Therefore, the latter takes as input a string value for which a DBpedia URI must be found, a DBpedia class of the column to which the string value belongs as well as the maximum number of results that we want to return.

Example 3.1. *For the example Table 3.1 , the CTA method assign the [dbo:Film, dbo:Work] classes to the “Title” column and the [dbo:Person, dbo:Agent] classes to the “Director’s” column, however the other columns have no matches detected in the DBpedia ontology. Then, using the returned column classes and the values of the table cells, a set of possible candidate entities for each mention is generated using the DBpedia lookup method. Taking for example the mention “Lawrence of Arabia” which is the intersection of line 4 and column 1, it returns the following list of candidate entities:*

- **Label:** *A Dangerous Man: Lawrence After Arabia*
Uri: *http://dbpedia.org/resource/A_Dangerous_Man:_Lawrence_After_Arabia*
- **Label:** *Lawrence of Arabia*
Uri: *http://dbpedia.org/resource/Lawrence_of_Arabia*

¹The table originates from the T2D gold standard: 58891288.0_1117541047012405958

- **Label:** *Lawrence of Arabia: The Authorised Biography of T.E. Lawrence*
Uri: http://dbpedia.org/resource/Lawrence_of_Arabia:_The_Authorised_Biography_of_T.E._Lawrence
- **Label:** *Lawrence of Arabia (film)*
Uri: [http://dbpedia.org/resource/Lawrence_of_Arabia_\(film\)](http://dbpedia.org/resource/Lawrence_of_Arabia_(film))
- **Label:** *Saint Lawrence River*
Uri: http://dbpedia.org/resource/Saint_Lawrence_River
- **Label:** *Saudi Arabia*
Uri: http://dbpedia.org/resource/Saudi_Arabia
- **Label:** *T. E. Lawrence*
Uri: http://dbpedia.org/resource/T._E._Lawrence

This example shows that many lookup results can be returned for the query "Lawrence of Arabia", but we want to select a single entity of an acceptable type (e.g., Film). To achieve this goal we need to find a way to deal with massive amounts of candidates per mention. Hence, to keep the list of candidates short and to improve efficiency, we eliminate noisy candidates by assigning a score for each candidate; this score presents the similarity between the mention and each entity in the list of candidates. The next step which is the collective-based Entity Disambiguation in web tables aims to select the most suitable entity for each mention.

Collective Strategy for Entity Disambiguation

Our proposed Entity Disambiguation approach is similar to that proposed in (Zwicklbauer et al., 2016) which is a collective EL approach for text disambiguation. Therefore, in our approach, we examine the relatedness of entities using a coherence Graph such as candidate entities as vertices and edges reflecting the semantic similarity between each pair of entities that correspond to two different mentions in a Web Table i.e., there is no edge between the candidate entities of the same mention. The intuition behind this approach is that entities in the same column should be closed in the embedding space as they share semantic characteristics.

In the disambiguation graph, there are only weighted directed edges between the

nodes (entities). These edges have the following format $(e_1; e_2; etp(e_1; e_2))$ where $etp(e_1; e_2)$ represents Entity Transition Probability (ETP) which describes the likelihood to walk from node e_1 to node e_2 . In our case, we use the cosine similarity to measure the ETP between two non-zero d-dimensional vectors corresponding to the target entities (e_1, e_2) . Cosine Similarity is one of the most frequently used metrics for words similarity Moreno et al., 2017. Given two entities vectors W_{vec1}^{\rightarrow} and W_{vec2}^{\rightarrow} , the cosine similarity is calculated as follows:

$$Cos_Sim(W_{vec1}^{\rightarrow}, W_{vec2}^{\rightarrow}) = \frac{W_{vec1}^{\rightarrow} \cdot W_{vec2}^{\rightarrow}}{\|W_{vec1}^{\rightarrow}\|, \|W_{vec2}^{\rightarrow}\|} \quad (3.3)$$

where the returned result of *Cos_Sim* varies between -1 and 1.

Despite the performance of cosine similarity in semantic measurement, the quality of the results may differ depending on the method used to determine the vector representations of terms. Indeed, there are a lot of techniques used to mapped terms to n-dimensional vectors like One-hot vector representations which generally do not provide any information on the meaning of term (Cerda et al., 2018) and TF-IDF (term frequency- inverse document frequency) technique where TF is how often the term appears in the document whereas IDF is the number of documents in which the term appears (Robertson, 2004). However, word embeddings has proven to be one of the most popular and beneficial methods in several works (Efthymiou et al., 2017; Ritze et al., 2015). With Word2Vec framework, words are represented in a continuous vector space by associating related words to relatively close points to display their semantic similarity.

In order to learn word embeddings, (Mikolov et al., 2013) proposed two popular models which are Continuous Bag of Words (CBOW) that used to predict a word using a set of words constituting its surrounding context and Skip-gram that used to predict the context of each word. Figure 3.2 shows the architecture of the CBOW and Skip-gram models.

As illustrate in figure 3.2, the CBOW model predicts the current word $w(t)$ using its context $w(t - n), \dots, w(t - 2), w(t - 1), w(t + 1), w(t + 2) \dots, w(t + n)$ which is a set of words in the form of vectors, while the Skip-gram model predicts each word in the context using the word $w(t)$.

In our work, the similarity measure calculated using neural network language

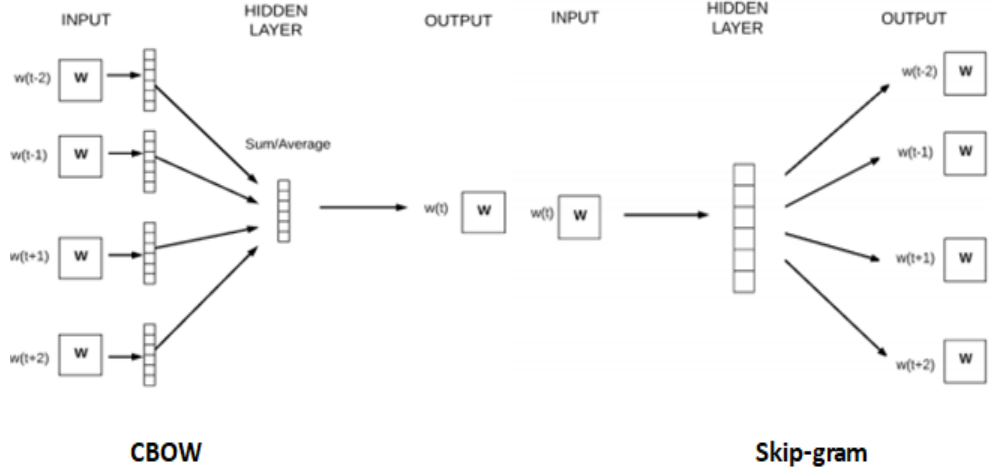


Figure 3.3: Architecture of CBOW and Skip-gram models

models which are the word and entity embeddings.

$$ETP(e_1, e_2) = \text{cosine_similarity}(\text{embedding}(e_1), \text{embedding}(e_2)) \quad (3.4)$$

The details of the collective disambiguation strategy will be described in the following steps:

1. Measure the local similarity score between the mention m_i and the entity $e_{i,j}$ (as described in the candidate generation step)
2. Estimate the global coherence score between each two entities based on the assumption that two entities are related if their vector embeddings are highly close.
3. Create a weighted directed graph $G=(V, E)$ where V is the set of candidate entities while E represents the edges between every pair of entities weighted by the ETP value.
4. Given the weighted graph, we apply a Personalized PageRank (PPR) algorithm to determine a PPR score for each entity candidate.

Local similarity score $\phi(m_i, e_{i,j})$:

We adopt the similarity score as introduced in the candidate generation step as a local similarity score between a mention m_i and the candidate $e_{i,j}$. Therefore,

the local similarity score determined using three different metrics, including the contextual similarity $CxtSim(M_i, E_i)$, where M_i is the set of all mentions appearing with m_i in the same row and which represents the context of the target mention m_i while E_i is the set of words appearing in the short description of the entity $e_{i,j}$ extracted from the knowledge base which used as a context of the entity $e_{i,j}$. The String Similarity score $SSim(m_i, e_{i,j})$ whereas we apply two different approaches to calculate $SSim(m_i, e_{i,j})$ which are the levenshtein distance and the jaccard similarity, and finally the semantic similarity $SemSim(m_i, e_{i,j})$ between the candidate entity and the mention.

$$\phi(m_i, e_{i,j}) = \alpha * CxtSim(M_i, E_i) + \beta * SemSim(m_i, e_{i,j}) + \lambda * SSim(m_i, e_{i,j}) \quad (3.5)$$

We exploit the cosine similarity measure to estimate the Semantic similarity $SemSim(m_i, e_{i,j})$ between the mention m_i and the entity $e_{i,j}$ but here we adopt both word and entity embeddings since our experiments shows that joint modeling of words and entities in the same continuous space improves the quality of word and entity embeddings and benefits the task of entity disambiguation. Let T denote the target token where T="Saudi Arabia", we can consider T either as a set of two different words (e.g., Saudi, Arabia) or as an ENTITY (e.g., [Saudi_Arabia]).

Global coherence score (e_i, e_j):

Several previous studies in EL exploit the relatedness measure between the entities to generate a disambiguation context. Thus, in comparison with previous work, in our case, we use both word and entity embeddings similarities because the integration of entity embedding to estimate the semantic coherence between the concepts proved effective for entity disambiguation in recent works (Efthymiou et al., 2017, Efthymiou et al., 2017).

$$\varphi(e_i, e_j) = a \times [Sim(emb(e_1), emb(e_2))] + (1 - a) \times Sameclass(e_1, e_2) \quad (3.6)$$

The global coherence score is computed using a cosine similarity between vector representations of two entities as well as a simple function called $Sameclass(e_1, e_2)$ which aim to improve the score value if the two target entities have the same class.

Collective Disambiguation Strategy

In our approach, we create a weighted and directed graph with edges connecting each two entities referred to two different mentions. The edges are weighted

by the score of the global context matching $\varphi(e_i, e_j)$ which describe the ETP of walking from one node to another. An example of an entity-entity disambiguation graph is illustrated in Figure 3.4 .

Algorithm 3.1 Graph-based EL in web tables

Input: N mentions (m_1, \dots, m_N) each mention m_i has a list of candidate Entities

$CE_{m_i} = [e_1, \dots, e_{|CE|}]$

Output: $\Gamma^* = (e_1, \dots, e_N)$ with e_i being the entity matching to m_i

Begin

// Local similarity candidate Filter

for $m_i \in M$ and $|CE_{m_i}| \geq 1$ **do**

 FinalCan = ϕ

if $localSimilarityScore(m_i, e_i) \geq thresholdSim_{m,e}$ and $class_{e_i} \neq \phi$ **then**

 FinalCan = FinalCan + e_i

if FinalCan == ϕ **then**

$CE_{m_i} \leftarrow NIL$

else

$CE_{m_i} \leftarrow FinalCan$

// Create a Weighted Direct Graph WDG

$nodes = \phi$

for m_i in M **do**

for $e_j \in CE_{m_i}$ **do**

$nodes \leftarrow nodes + e_j$

WDG = Create_DiGraph(nodes, GlobalSimilarity($node_{e_i}, node_{e_j}$))

Matrix = Convert_to_Matrix(WDG)

// Apply Personalized PageRank PPR

for m_i in M **do**

 PPR = Personalized_PageRank(Matrix)

 Rank entities

if $CS_{m_i} \geq 1$ **then**

 select for each m_i the entity with the highest score

else

m_i annotate with NIL

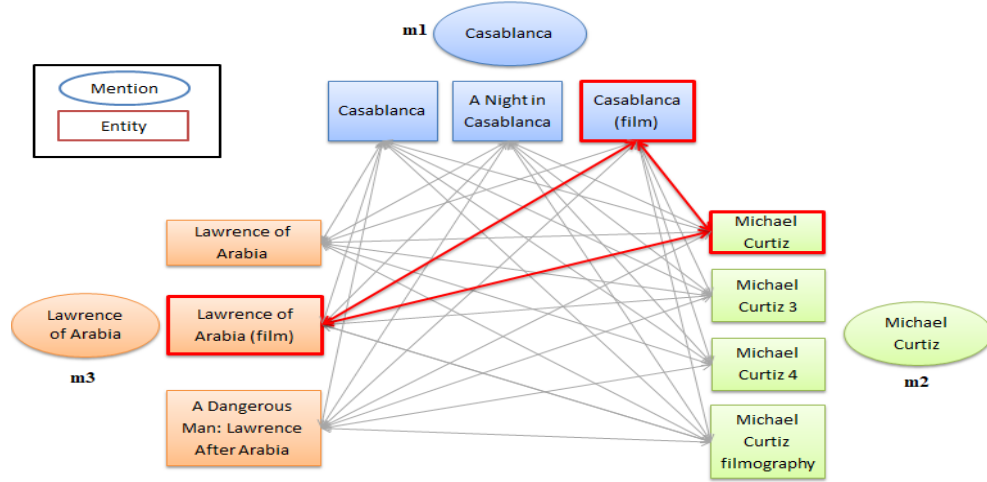


Figure 3.4: Example of an Entity-Entity disambiguation graph; Where both "Casablanca" and "Lawrence of Arabia" mentions have three possible candidates, while "Michael Curtiz" mention has four possible candidates. The weight $\varphi(e_i, e_j)$ edges is the cosine similarity between the vectors representations of the candidates entities e_i, e_j as well as the $\text{Sameclass}(e_1, e_2)$ binary function which aim to improve the score value if two target entities e_i and e_j have the same class.

Based on the disambiguation graph, we perform a random walk simulated by a PageRank procedure which aims to encode the structure of the graph in the form of a transition matrix in order to determine a ranking list of relevance scores for each entity candidate. Depending on this list, our approach decides which entity candidate is the correct entity that corresponding to the target mention.

In our approach, we use Personalized PageRank (PPR) Huang et al., 2014 which is more general than PageRank version. The personalized PageRank based on two parameters, the jumping constant α and the seed s . In fact, a seed can be considered as either a vertex or a probability distribution on vertices i.e. a probability distribution on a seed node set which serves as the personalization context. On this way, given a weighted graph $G(V, E)$, a damping factor α and a set of nodes $S \in V$ denoted as the seed vector \vec{s} , if we denote the Personalized PageRank (PPR) scores of the nodes in V with a vector π , then

$$\pi = (1 - \alpha) * s + \alpha * T_G * \pi \quad (3.7)$$

where T_G denotes the transition matrix corresponding to the graph G (and the underlying edge weights). s is a re-seeding vector, such that if $v_i \in S$, then $\vec{s}[i] = \frac{1}{\|S\|}$ and $\vec{s}[i] = 0$, otherwise. Therefore, rather than leaping to a random node in V with probability α , the random walk jumps to one of the nodes from the set of seed S .

3.2.4 Columns Predicate Annotation CPA

The structure of a web table inherently provides high quality semantic relations between its columns. However, extracting these relations is a challenging task since it requires the identification of unique structural information of each table. In the perspective of the knowledge base, these relations are considered as properties or attributes.

In general, every relational web table has a subject column which is a key of the web table and hence every other column represents a binary relation with the subject column. Within this context, we aim to identify the semantic relations of the subject column in a table with other columns (both Named entity and literal-columns) in the same table using the linkable table ,in others terms, with the help of entity linking and class matching tasks, we turn to determine which relation might exist between the subject column and the attribute columns in order to set the overall meaning of the table.

Subject Column Detection SDC

Subject column, also called entity label column or key column, contains the names of the entities described by the web table. Assume that each web table has exactly one subject column, to detect the latter we apply several heuristic as shown in Algorithm 2.

Algorithm 3.2 Pseudocode to find the subject column

```

SubjColm= None
// SubjColm is the most unique column in the table
for c in C do
  if data_type(c)==str then
    if len(mention)>=3 and len(mention)<=200 then
      if "name" or "title" in header(c) then
        if confidenceScore(c) >= 0.3 then
          SubjColm= c
        else if confidenceSc(c) == max(confidenceSc(C)) then
          SubjColm= c

```

First of all, the column needs to be of data type `String` and its cells have an average length between 3 and 200 characters. Since the table header is not always available and that arbitrary terms can be used, we cannot rely solely on the header to detect the subject column. To the best of our knowledge, we calculate a confidence score which is defined as the ratio of annotated mentions to all mentions of a column, reduced by the ratio of non-annotated mentions, e.i. mentions mapped to value "NIL" :

$$confidenceScore(c) = \frac{\#annotatedMentions}{\#totalMentions} \quad (3.8)$$

The confidence score is important for detecting the correct subject column. It rewards columns with many annotated cell and penalises columns with many "NIL" annotations. Hence, two rules decide which column is selected as subject column: (1) If the column header contains either the term "name" or "title" and the confidence score above 0.3, this column chosen as subject column; (2) otherwise, the column with the highest confidence score is chosen and in case of a tie, the left-most candidate is chosen.

Note that the purpose of subject column detection is not limited to identify the main class of web tables, but also used by later processes such as columns predicate annotation.

In our case, we firstly begins by generate a set of candidate properties using the following three matchers for the columns predicate annotation task:

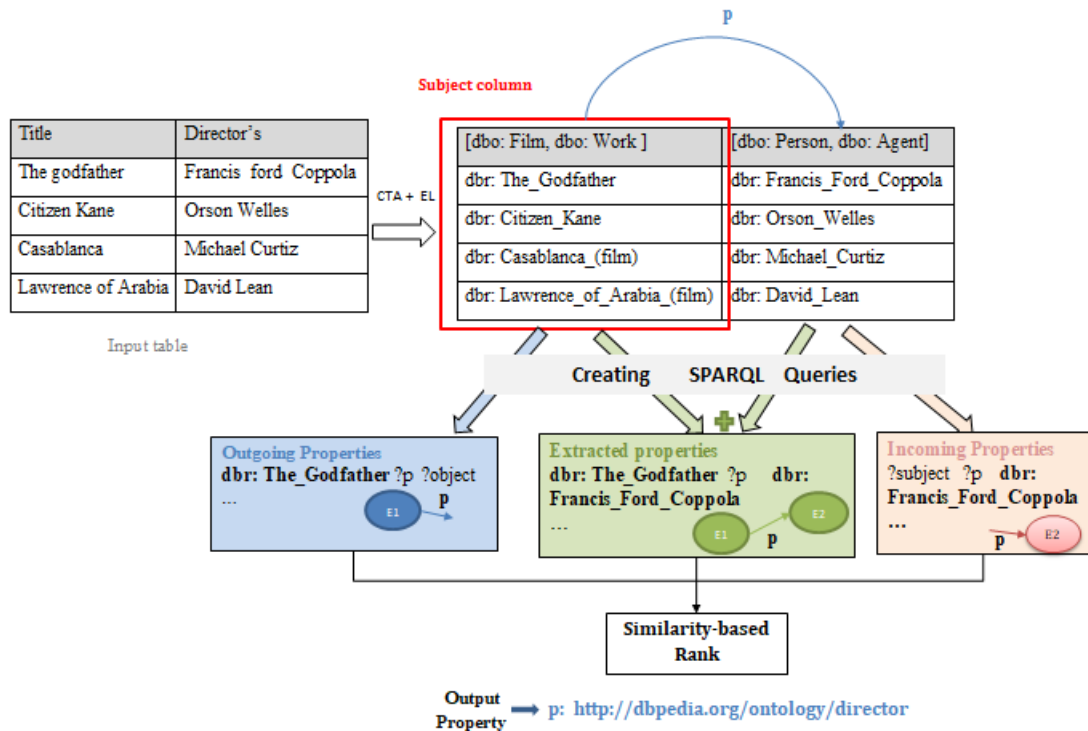


Figure 3.5: A running example of the CPA task

a) Outgoing-Properties

We introduce a subject column based search mechanism for the property selection. Indeed, for each entity in the subject column, we query the DBpedia knowledge base to find all the possible relationships emerging from that entity. As mentioned in the SPARQL query given in Listing 3.2, we ask for all properties in the DBpedia knowledge base that corresponds to the relations that a subject column expect from the other columns of the table. For example, the subject column c_0 of table in Figure 3.7 can have the outgoing properties *dbp.areaTotal*, *dbp.averageDepth*, *dbp.elevation*, *dbp.length*, *dbp.location*...

```

1 PREFIX dbr: <http://dbpedia.org/resource/>
2 select distinct ?predicate ?object where{
3 %URI% ?predicate ?object .
4 }

```

Listing 3.1: SPARQL query used in the Outgoing-Properties extract

b) Incoming-Properties

Incoming-Properties refers to all properties coming into all entities in the Named

Entity column except the subject column, in the perspective of other entities. It defines the behavior of an entity in relation to all subject entities. Given e is an entity of a NE-column derived from the triples containing e as object, denoted by $\langle ?subject, ?predicate, e \rangle$. For an example, column c_1 from the above table shown in figure can have the Incoming-Properties *dbp.location*, *dbp.deathPlace*, *dbp.birthPlace*.

```

1 PREFIX dbr: <http://dbpedia.org/resource/>
2 select distinct ?subject ?predicate where{
3 ?subject ?predicate %URI%.
4 }

```

Listing 3.2: SPARQL query used in the Incoming-Properties extract

c) Extracted-Properties

Here, we aim to extract all possible relations between a subject column and a NE-column. For this purpose, we query Dbpedia for the properties between all pairs of entities for each row in the primary and secondary columns, as shown in tuple illustrate in Listing 3.4 $\langle \%URI1\%, ?predicate, \%URI2\% \rangle$, where $\%URI1\%$ and $\%URI2\%$ are entities in the subject column and the NE-column respectively. As instance, the Extracted-Properties between column c_0 and column c_1 are: *dbp.location*

```

1 PREFIX dbr: <http://dbpedia.org/resource/>
2 select distinct ?predicate where{
3 %URI1% ?predicate %URI2% .
4 }

```

Listing 3.3: SPARQL query used in the

At the end of the properties extraction process a set of candidate properties is generated. In next step, we rank these candidate properties by importance in order to determine the best possible candidate relation between two given columns. To do so, we assign a score for each candidate relation $\langle subject_column, property, object_column, score \rangle$ by utilizing the following metrics:

- **Similarity.** For each candidate in the Outgoing-Properties set, we compute a string similarity(Levenshtein) and a cosine similarity(words embeddings) between the name of the property candidate and the header text of the object column (if exists).

- **Property validity.** We determine the validity of a property by checking whether a candidate property is apparent in both the outgoing-properties and in the incoming-property. If so, a property validity function returns 1 or 0 otherwise. The purpose of this function is to identify one or more valid properties that can link the subject column to the object column.
- **Property confidence function.** Specifies how likely the relation holds. This function based on the frequency of occurrences of each candidate in the Extracted-Properties. It varies between 0 and 1 while 1 declares that the property of the greatest frequency is most likely to hold.

Finally, we aggregate all metrics listed above to predict a final score S_p for each candidate property, i.e.,

$$S_p = \alpha * Sim(p, h_c) + \beta * V_p + \gamma * Conf_p \quad (3.9)$$

Where Sim include the Levenshtein similarity and the cosine similarity between the header text of a column h_c and the name of a candidate concept p , V_p is the validity function of p and $Conf_p$ is the Property confidence function.

Finally, the candidate property with the highest score S_p is chosen as the best binary relation associating a subject column with a secondary column.

3.3 Conclusion

In this chapter, we introduced our proposed approach for matching web tables to knowledge bases. Indeed, this chapter gives all the detailed steps of our approach. First, we described how to clean up a web table, then we show how we generate the candidates entites from DBpedia knowledge graph. Last, we described the main step of our approach: Entity Disambiguation step from which we created the disambiguation graph and we applied a personalized pagerank (PPR) to select the correct entity for each mention in the web table.

Evaluation

Introduction

In this chapter, we will experimentally evaluate our method for web tables annotation. First, we will present our experimentation in section 4.1. Second, we will present the results for each matching task and we analyze the utility of different features that serve as input for these tasks by applying different combinations of features.. The last section aims to compare our results with other approaches.

4.1 Experiments

Our matching method is fully implemented in python. As input for the matching process, We exploit DBpedia as knowledge graph and T2D gold standard (version 1) as datasets of web tables. For more details, this section presents the datasets and the knowledge graph used in our implementation as well as the practical steps necessary to realize and evaluate our approach.

Table 4.1: Characteristics of the T2D gold standards.

Name	Nb.tables	Rows per table			Columns per table			Structuredness
		min	max	avg	min	max	avg	
T2D	233	6	586	123	3	14	4.95	0.97

4.1.1 Datasets:

In our experiments, we use the tables of the first version T2D-instance gold standard¹ which contains 233 Web tables, with manual annotations of 26124 entities extracted from the 2014 version of DBpedia.

T2D has an average of 123 rows per table, and 4,95 columns per table. It is highly structured (0.97), meaning that very few cells are empty in tables. Table 4.1 summarizes the main features of the T2D gold standard.

The T2D gold standard includes cross-domain tables from multiple websites covering a wide range of challenges. Figure 4.1 shows the distribution of tables per category.

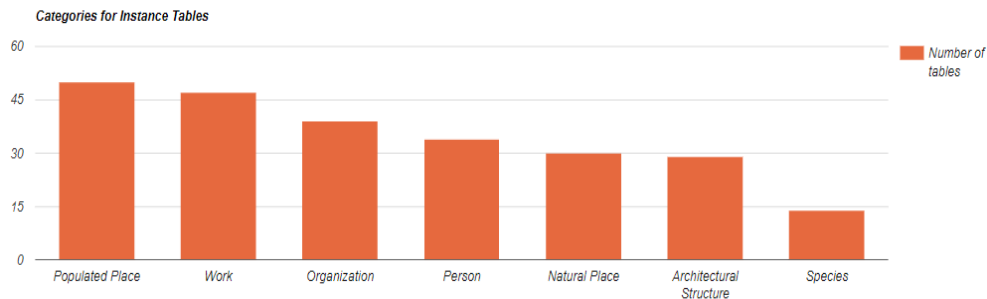


Figure 4.1: The distribution of tables per category

4.1.2 Knowledge graph

As knowledge graph, we decided to use Dbpedia knowledge base which is one of the most common knowledge bases. DBpedia is a very popular knowledge graph that is located in the center of the LOD cloud because it is highly interconnected

¹<http://webdatacommons.org/webtables/goldstandard.html>

with many other datasets in the Semantic Web. Furthermore, The DBpedia knowledge base presents major advantages compared to the existing knowledge bases: it covers current areas and is known as the cross-domain knowledge base; it evolves automatically as Wikipedia changes, and it is truly multilingual. Indeed, DBpedia provide 125 languages versions which describe together 38.3 million things. Especially, the English version of the DBpedia knowledge base describes 4.58 million things².

There are different ways to extract knowledge from dbpedia. In our implementation we choose to work with dbpedia API to generate a set of candidates entities for each string mentioned in a web table.

4.1.3 Wikipedia2Vec: Pre-trained model of Words and Entities Embeddings

We base our work on semantic embeddings method which uses distributed representations (embeddings) of the rich entity context in a KB to estimate the semantic similarity between entities.

Several neural network language models have been adopted in recent years, one of the most popular is word2vec that embeds words in a lower-dimensional vector space. It is a two-layer neural networks that can works with two model architectures: CBOW model and Skip-Gram model.

To the best of our knowledge, we use Wikipedia2Vec, a model for obtaining embeddings of entities and words simultaneously. The strong point of this method is that it based on word2vec and aims to jointly place words and entities in the same continuous vector space. Indeed, similar words and entities are close together in the same vector space (Yamada et al., 2018). This is what allows us to easily calculate the cosine similarity between any pair of items (entity-entity, word-entity or word-word).

However, Wikipedia2Vec has been used in several recent works such as: Entity linking (Chen et al., 2019), Named entity recognition (Lara-Clares and García-Serrano, 2019) and Entity typing (Yamada et al., 2018).

²<http://wiki.dbpedia.org/about>

4.2 Evaluation

In this section, we analyze the results achieved by our system as well as the usefulness of the various functionalities which serve as input for the mapping of web tables to knowledge bases.

4.2.1 Evaluation metrics

In literature, web table annotation systems are mainly evaluated with effectiveness measures. For which, subject column detection is evaluated by Precision and the confidence of the three annotation tasks of Semantic Table Interpretation are evaluated using the standard Precision, Recall and its combination F-measure. These measures are calculated based on the following formulas:

$$Precision = \frac{TP}{(TP + FP)} \quad (4.1)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (4.2)$$

$$F - measure = \frac{(2 * Precision * Recall)}{(P + R)} \quad (4.3)$$

While TP refers to the number of true positives which are the set of correctly annotated cells, FP represents the number of false positives which are the set of mentions that are annotated with wrong entities where the correct entity exists among the candidate set and FN the number of false negatives: the set of mentions that are annotated with wrong entities where the correct entity does exist among the candidate set. Note that the precision of the subject column calculated by dividing the number of subject columns correctly determined by the number of total tables.

4.2.2 Experiment Results

In this section, we evaluate the overall performance of our approach. Table 4.2 illustrates the results of matching the ambiguous tables of T2D to the DBpedia

Table 4.2: Results of the three matching tasks

Task	Precision	Recall	F-measure
CTA	0.84	0.91	0.87
EL	0.87	0.86	0.87
CPA	0.69	0.71	0.70

Table 4.3: Results of the CTA task using different features

Method	Precision	Recall	F-measure
String Similarity	0.65	0.34	0.45
frequency-based	0.81	0.78	0.79
Embedding-baseline	0.83	0.81	0.82
All	0.90	0.85	0.87

knowledge base using our approach. For the subject column detection, an F-measure of 0.97 is observed, this means that for almost all tables the correct subject column is determined. This is an important finding since an incorrect subject column mostly results in incorrect correspondence for the properties correspondence task. For the class matching task, an F-measure of 0.86 can be achieved, followed by the performance for entities with 0.86 and properties with 0.70.

4.2.3 Results of the Column Type Prediction task

Table 4.3 presents the results of our Column Type Prediction experiments. Considering only the string similarity (Levenshtein) between the candidate class and the column header to find the correct column class, the precision is 0.65 and the recall 0.34. Meaning that for only fewer than half of the tables, the correct class is assigned, due to the Levenshtein similarity that compares the sets of tokens from two strings or applies a character-based comparison. On the other hand, by only using the frequency-based matcher to find the correct column class, an F-measure of 0.79 can be achieved.

In order to see the importance of applying methods that rely on context features, we evaluate the embeddings method independently of the others, an F-measure

of 0.82 is obtained which is greater than the result of the frequency-based and the string similarity methods. we can therefore conclude that the use of embeddings is the most important feature for the class matching task. When we combine all the previous methods, we reach an F-measure of 0.87. Knowing that the class correspondence task has a strong influence on the other two correspondence tasks, their performance can be considerably reduced with an incorrect class.

4.2.4 Results of the Entity Linking task

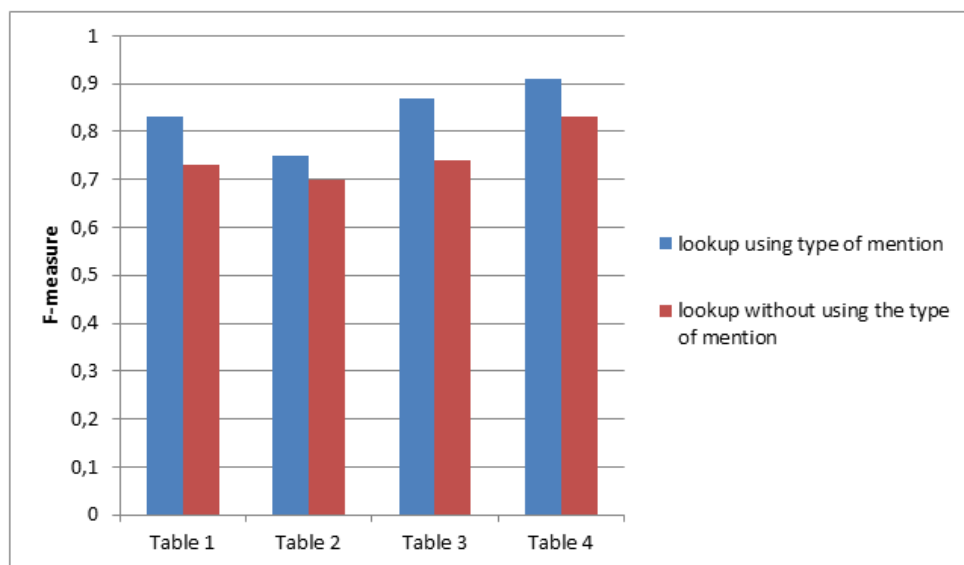


Figure 4.2: The impact of using column types in our system

Lookup method. Generally, an improved lookup method is necessary to achieve high-quality results. Here we try to add some specializations to the lookup method in order to improve the chances of finding the correct entity corresponding to the target mention. Indeed, such specialization is applied to the query to increase the cohesion of the search results; Figure 4.3 compare between results of a simple DBpedia lookup and an improved lookup.

Similarity threshold. After computing the local similarity between the mentions and their candidate entities, we apply a similarity threshold which kept all candidate above a configurable threshold as final candidates i.e., if the result is below

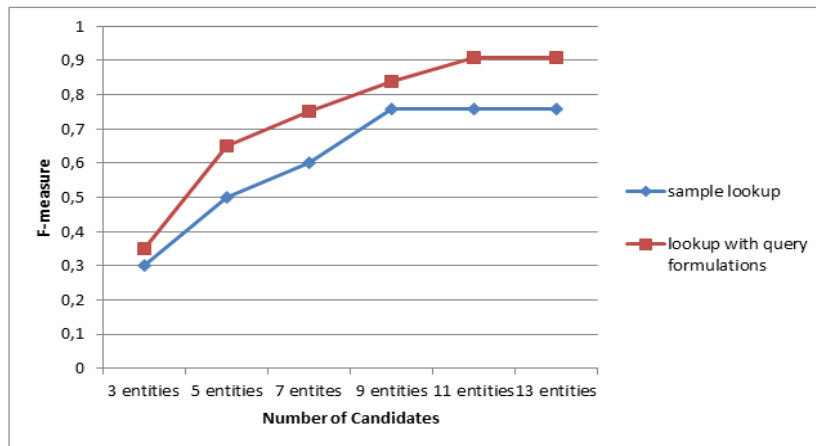


Figure 4.3: Comparison between the results of a simple DBpedia lookup and an improved lookup

Table 4.4: Variation of the local Similarity threshold

<i>CandidateFilter</i> threshold	F-measure
0.3	0.79
0.4	0.95
0.5	0.93
0.6	0.93
0.7	0.90

the given threshold, the entity will be removed from the list of candidates. Hence, to fix the value of this similarity threshold, we evaluate the F-measure result by adjusting his value. Table 5.2 show the effect of varying the similarity threshold.

The experimental results illustrated in table 4.4 shows that our system can achieve high results only if the similarity threshold between both mentions and candidates entities is equal to 0.4.

Personalized PageRank PPR. At this stage, we have a list of refined candidate entities. After creating the weighted directed graph where nodes are candidate entities for all mentions, we apply the PPR algorithm to select for each mention, the entity that have the highest score. However, to calculate the PPR score of all the nodes of the graph, PPR takes 4 input parameters:

Table 4.5: Variation of the number of seed nodes

Number of seed nodes	Precision	Recall	F-measure
1	0.82	0.87	0.84
2	0.83	0.87	0.85
3	0.84	0.86	0.85
4	0.87	0.86	0.86
5	0.87	0.86	0.86
6	0.83	0.80	0.81

Table 4.6: Results of the Columns Predicate Annotation task

Matchers	Precision	Recall	F-measure
Outgoing Properties	0.61	0.65	0.63
Incoming + Outgoing Properties	0.65	0.69	0.67
Extracted-Properties	0.56	0.44	0.50
All	0.69	0.71	0.70

- The disambiguation graph in the form of an adjacency matrix.
- Number of iterations: 50 PPR iterations since the overall results do not change with more iterations.
- The PageRank jump probability α : α is a damping factor with a default value of 0.85.
- Select the seed nodes: a small set of nodes which serves as the personalization context. Table 4.5 describe the effect of varying the number of seed nodes on the final results. Hence, the suitable number of seed nodes that guaranty the effectiveness of our system is 4.

4.2.5 Results of the Columns Predicate Annotation task

The results of the Columns Predicate Annotation experiments using different combinations of matchers are illustrated in Table 4.6. Indeed, if we only take the extracted-Properties between each two entities in the same row, we get a rather low recall of less than 0.5. Based on these results, we are already know that the properties extracted from the triple $\langle e1, p, e2 \rangle$ is not necessarily a useful feature

Table 4.7: Comparison with methods focusing on the CTA task in Web tables

Method	CTA		
	Precision	Recall	F-measure
T2K Match	0.94	0.94	0.94
Our method	0.84	0.91	0.87

Table 4.8: Comparison with methods focusing on the EL task in Web tables

Method	EL		
	Precision	Recall	F-measure
DBpedia Lookup	0.79	0.73	0.76
T2K Match	0.90	0.76	0.82
[Efthymiou et al., 2017]	0.87	0.83	0.85
Our method	0.872	0.862	0.867

for all tables. However, if we consider only the Outgoing-Properties matchers to find the correct properties between columns, the precision is 0.61 and recall 0.65.

Including values, covered in the incoming-properties matcher, increases both the precision and the recall by 0.04. When we use all matchers together, a precision of 0.69 with a recall of 0.71 can be achieved. This confirms that the outgoing-properties matcher is the most important matcher for the task of columns predicate annotation .

4.3 Comparison with other Approaches

In this section, we aim to estimate the performance of our system by comparing our results with other recent approaches to table matching like DBpedia Lookup, (Efthymiou et al., 2017), T2K Match and its extended version T2K Match++. Tables 4.7, 4.8 and 4.9 summarise the comparison between these approaches tested on the T2D corpus. As illustrated in table 5.2, our method outperforms the DBpedia lookup, (Efthymiou et al., 2017) and the T2K match method, with an increase of 0.1 in F-measure over the Dbpedia lookup, 0.04 over T2K Match and an improvement of only 0.02 over the (Efthymiou et al., 2017). For the extended version of T2K (T2K Match++),it have a high F-measure of 0.87 almost similar to our results.Hence, we can say that our system considered to be one of the best web

Table 4.9: Comparison with methods focusing on the CPA task in Web tables

	CPA		
Method	Precision	Recall	F-measure
T2K Match	0.77	0.65	0.70
Our method	0.69	0.71	0.70

table annotation system.

4.4 Conclusion

During this chapter, we performed an experimental investigation towards our new method of semantic table interpretation which is presented in the previous chapter. We first introduced the datasets features (T2D), the exploited Knowledge base (DBpedia) as well as the pre-trained embeddings using in order to evaluate the performance of our method. After evaluating our method, results shows the high quality of our method compared to other matching system.

Conclusion

Web tables are a source of valuable data that has been widely used in a variety of cases, such as fact search, knowledge base augmentation or Entity Linking. And despite that, there is only limited approaches that focus on relational data like web tables.

In this master thesis, we propose a novel unsupervised approach for semantic interpretation of web tables by solving all the three matching tasks namely Column Type Annotation (CTA), Entity Linking (EL) and Columns Predicate Annotation (CPA) using DBpedia KB. More precisely, CTA, aims to match table columns with widely recognized concepts like semantic classes of a KB. We mainly combined two baselines which are semantic embeddings and majority voting.

The EL task builds a weighted directed graph to find the best candidate mapping between a mention in the Web table and the corresponding entity in the KB. This collective strategy based on two main features concerning the local similarity between mentions and entities as well as the global coherence between entities using words and entities embeddings representations. Thus, our EL method working in three main steps: First, a set of candidate entities was generated from Dbpedia for each mention in the web table. This list is then passed to a filter simulated by a local similarity between a mention and each candidate entity in the set of entities. The local similarity calculated in combination with the string similarity and the cosine similarity between embeddings of both mention and entity candidate. Once incorrect matches are eliminated using a similarity threshold, we created a direct disambiguation graph weighted with a global coherence between entities candidates to different mentions, then we applied a personalized pagerank PPR in order

to select for each mention, the entity that have the highest ranking score. The last task, CPA, presents a novel method to analyse the behaviour between row entities to collectively deduce the possible relations between table columns. For this task, we categorize the properties generated between columns into three groups: (1) Outgoing-Properties, (2) Incoming-Properties, and (3) Extracted-Properties.

To evaluate the performance of our system, we exploit a corpus of 233 web tables extracted from T2D gold standard which captures correspondences for all the three matching tasks between tables and DBpedia KG. By observing the results of the evaluation, our system obtained relatively high quality results for the EL task and acceptable results for the other two tasks compared to state-of-the-art Web table annotation methods. As future work, we would like to improve the following tasks:

- We aim to improve our approach especially the CTA task using machine learning technique to automatically train prediction models for annotating types of entity columns that are assumed to have no metadata.
- We plan to extend our proposed method to other related tasks such as the discovery of new entities in Web tables for KB population, augmentation and refinement.

References

- Sparql 1.1 overview, w3c recommendation 21 march 2013. Accessed: 2020-03-12.
- Berners-Lee, T. (2006). Linked data. last change 2009/06/18.
- Bhagavatula, C., Noraset, T., and Downey, D. (2015). Tabel: Entity linking in web tables. In *Proceedings of the 14th International Semantic Web Conference (ISWC'15)*, page 425–441.
- Bizer, Christian, Heath, T., and Berners-Lee, T. (2011). Linked data: The story so far. *Semantic services, interoperability and web applications: emerging concepts*. IGI Global, page 205–227.
- Bollacker, K. D., Evans, C. J., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*.
- Cafarella, M. J., Halevy, A. Y., Zhang, Y., Wang, D. Z., and Wu, E. (2008). Uncovering the relational web. In *Proceedings of the 11th International Workshop on Web and Databases (WebDB 2008)*, Vancouver, Canada.
- Carroll, J. J. and Klyne, G. (2004). Resource description framework (RDF): Concepts and abstract syntax.
- Cerda, P., Varoquaux, G., and Kégl, B. (2018). Similarity encoding for learning with dirty categorical variables. *Machine Learning*, 107:1477–1494.
- Chen, H., Wadhwa, S., Li, X. D., and Gregoric, A. Z. (2019). Yelm: End-to-end contextualized entity linking. *ArXiv*, abs/1911.03834.
- Chen, J., Jiménez-Ruiz, E., Horrocks, I., and Sutton, C. A. (2018a). Colnet: Embedding the semantics of web tables for column type prediction. *CoRR*, abs/1811.01304.

- Chen, J., Jiménez-Ruiz, E., Horrocks, I., and Sutton, C. A. (2018b). Colnet: Embedding the semantics of web tables for column type prediction. In *AAAI*.
- Cheng, X. and Roth, D. (2013). Relational inference for wikification. In *EMNLP*.
- Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., and Christophides, V. (2017). Matching web tables with knowledge base entities: From entity lookups to entity embeddings. In *International Semantic Web Conference*, 260–277. Springer.
- Fetahu, B., Anand, A., and Koutraki, M. (2019). TableNet: An approach for determining fine-grained relations for wikipedia tables. *WWW '19: The Web Conference on The World Wide Web Conference WWW 2019*, pages 2736–2742.
- Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R. (2013). Evaluating entity linking with wikipedia. *Artif. Intell.*, 194:130–150.
- Han, X., Sun, L., and Zhao, J. (2011). Collective entity linking in web text: a graph-based method. In *SIGIR '11*.
- Huang, S., Li, X., Candan, K. S., and Sapino, M. L. (2014). “can you really trust that seed?”: Reducing the impact of seed noise in personalized pagerank. *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 216–223.
- Kim, S., Li, G., Feng, J., and Li, K. (2018). Web table understanding by collective inference. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 217–226.
- Lara-Clares, A. and García-Serrano, A. M. (2019). Lsi2_uned at ehealth-kd challenge 2019: A few-shot learning model for knowledge discovery from ehealth documents. In *IberLEF@SEPLN*.
- Lautert, L. R., Scheidt, M. M., and Dorneles, C. F. (2013). Web table taxonomy and formalization. *SIGMOD Record*, 42:28–33.
- Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.
- Lehmborg, O., Ritze, D., Meusel, R., and Bizer, C. (2016). A large public corpus of web tables containing time and context metadata. In *WWW*.
- Limaye, G., Sarawagi, S., and Chakrabarti, S. (2010). Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3:1338–1347.
- Luo, X., Luo, K., Chen, X., and Zhu, K. Q. (2018). Cross-lingual entity linking for web tables. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings*

- of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 362–369. AAAI Press.
- Mahdisoltani, F., Biega, J. A., and Suchanek, F. M. (2014). A knowledge base from multilingual wikipedias-yago 3 une base de connaissances des wikipédias plurilingues – yago 3.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, abs/1310.4546.
- Moreno, J. G., Besançon, R., Beaumont, R., D’hondt, E., Ligozat, A.-L., Rosset, S., Tannier, X., and Grau, B. (2017). Combining word and entity embeddings for entity linking. In *The Semantic Web (ESWC)*, pages 337–352, Cham. Springer International Publishing.
- Mulwad, V., Finin, T. W., Syed, Z., and Joshi, A. (2010). T2ld: Interpreting and representing tables as linked data. In *ISWC PostersDemos*.
- Nguyen, P., Kertkeidkachorn, N., Ichise, R., and Takeda, H. (2019). Mtab: Matching tabular data to knowledge graph with probability models. In *OM@ISWC*.
- Nguyen, T. T., Nguyen, Q. V. H., Weidlich, M., and Aberer, K. (2015). Result selection and summarization for web table search. In *2015 IEEE 31st International Conference on Data Engineering*, pages 231–242.
- Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.
- Pham, M., Alse, S., Knoblock, C. A., and Szekely, P. A. (2016). Semantic labeling: A domain-independent approach. In *International Semantic Web Conference*.
- Phan, M. C., Sun, A., Tay, Y., Han, J., and Li, C. (2018). Pair-linking for collective entity disambiguation: Two could be better than all. *IEEE Transactions on Knowledge and Data Engineering*, 31:1383–1396.
- Ramnandan, S. K., Mittal, A., Knoblock, C. A., and Szekely, P. A. (2015). Assigning semantic labels to data sources. In *ESWC*.
- Ristoski, P. (2018). Exploiting semantic web knowledge graphs in data mining. *PhD thesis*.
- Ritze, D., Lehmborg, O., and Bizer, C. (2015). Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics WIMS ’15*, pages 1–6.

- Robertson, S. E. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 60:503–520.
- Thawani, A., Hu, M., Hu, E., Zafar, H., Divvala, N. T., Singh, A., Qasemi, E., Szekely, P. A., and Pujara, J. (2019). Entity linking to knowledge graphs to infer column types and properties.
- Uyar, A. and Aliyu, F. M. (2015). Evaluating search features of google knowledge graph and bing satori: Entity types, list searches and query interfaces. *Online Information Review*, 39:197–213.
- Venetis, P., Halevy, A. Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering semantics of tables on the web. *PVLDB*, 4:528–538.
- Wang, Y. and Hu, J. (2002). Detecting tables in html documents. In Lopresti, D., Hu, J., and Kashi, R., editors, *Document Analysis Systems V*, pages 249–260, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Yakout, M., Ganjam, K., Chakrabarti, K., and Chaudhuri, S. (2012). Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD Conference*.
- Yamada, I., Shindo, H., and Takefuji, Y. (2018). Representation learning of entities and documents from knowledge base descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 190–201, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Zanibbi, R., Blostein, D., and Cordy, J. R. (2004). A survey of table recognition. *Document Analysis and Recognition*, 7:1–16.
- Zhang and Ziqi (2014). Towards efficient and effective semantic table interpretation. In Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., and Goble, C., editors, *The Semantic Web – ISWC 2014*, pages 487–502, Cham. Springer International Publishing.
- Zhang, S. and Balog, K. (2019). Auto-completion for data cells in relational tables. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.
- Zhang, S., Meij, E., Balog, K., and Reinanda, R. (2020). Novel entity discovery from web tables. *Proceedings of The Web Conference 2020 (WWW '20)*, page 11.
- Zhang, X., Chen, Y., Chen, J., Du, X., and Zou, L. (2013). Mapping entity-attribute web tables to web-scale knowledge bases. In Meng, W., Feng, L., Bressan, S., Winiwarter, W., and Song, W., editors, *Database Systems for*

- Advanced Applications*, pages 108–122, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Zhang, Z. (2017). Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, vol. 8, no. 6, pp. 921-957.
- Zwicklbauer, S., E., C., G., M., and Seifert., C. (2013). Towards disambiguating web tables. In *Proc. of the 12th Int. Semantic Web Conference*.
- Zwicklbauer, S., Seifert, C., and Granitzer, M. (2016). Robust and collective entity disambiguation through semantic embeddings. In *SIGIR '16*, pages 425–434.