PHD THESIS

In Partial Fulfilment of the Requirements for the

Degree of Doctor of Philosophy from Sorbonne University

Specialization: Data Science

Semantic Annotations for Tabular Data Using Embeddings: Application to Datasets Indexing and Table Augmentation

Jixiong LIU

Defended on 22/02/2023 before a committee composed of:

Reviewer   **Heiko PAULHEIM**, University of Mannheim, Mannheim, Germany

Reviewer   **Fatiha SAIS**, Paris Saclay University, Paris, France

Examiner   **Paolo PAPOTTI**, EURECOM, Sophia Antipolis, France

Examiner   **Ernesto JIMENEZ-RUIZ**, City University of London, London, United Kingdom

Thesis Director   **Ulrich FINGER**, EURECOM, Sophia Antipolis, France

Thesis Co-Director   **Raphaël TRONCY**, EURECOM, Sophia Antipolis, France

Thesis Co-Director   **Yoan CHABOT**, Orange, Belfort, France

Dedicated to my family

Acknowledgements

Abstract

With the development of Open Data, a large number of data sources are made available to communities (including data scientists and data analysts). This data is the treasure of digital services as long as data is cleaned, unbiased, as well as combined with explicit and machine-processable semantics in order to foster exploitation. In particular, structured data sources (CSV, JSON, XML, etc.) are the raw material for many data science processes. However, this data derives from different domains for which consumers are not always familiar with (knowledge gap), which complicates their appropriation, while this is a critical step in creating machine learning models.

Semantic models (in particular, ontologies) make it possible to explicitly represent the implicit meaning of data by specifying the concepts and relationships present in the data. The provision of semantic labels on datasets facilitates the understanding and reuse of data by providing documentation on the data that can be easily used by a non-expert. Moreover, semantic annotation opens the way to search modes that go beyond simple keywords and allow the use of queries of a high conceptual level on the content of the datasets but also their structure while overcoming the problems of syntactic heterogeneity encountered in tabular data.

This thesis introduces a complete pipeline for the extraction, interpretation, and applications of tables in the wild with the help of knowledge graphs. We first revisit the exiting definition of tables from the perspective of table interpretation and develop systems for collecting and extracting tables on the Web and local files. Next, we design, implement and evaluate three table interpretation systems, combining heuristic rules and graph embeddings models that tackle the challenges observed from the literature. Finally, we introduce and evaluate two table augmentation applications based on semantic annotations, namely data imputation and schema augmentation.

Abrégé

Avec le développement de l'Open Data, un grand nombre de sources de données sont mises à disposition des communautés (notamment les data scientists et les data analysts). Ces données constituent des sources importantes pour les services numériques sous réserve que les données soient nettoyées, non biaisées, et combinées à une sémantique explicite et compréhensible par les algorithmes afin de favoriser leur exploitation. En particulier, les sources de données structurées (CSV, JSON, XML, etc.) constituent la matière première de nombreux processus de science des données. Cependant, ces données proviennent de différents domaines pour lesquels l'expertise des consommateurs des données peut être limitée (knowledge gap). Ainsi, l'appropriation des données, étape critique pour la création de modèles d'apprentissage automatique de qualité, peut être complexe.

Les modèles sémantiques (en particulier, les ontologies) permettent de représenter explicitement le sens des données en spécifiant les concepts et les relations présents dans les données. L'association d'étiquettes sémantiques aux ensembles de données facilite la compréhension et la réutilisation des données en fournissant une documentation sur les données qui peut être facilement utilisée par un non-expert. De plus, l'annotation sémantique ouvre la voie à des modes de recherche qui vont au-delà de simples mots-clés et permettent l'expression de requêtes d'un haut niveau conceptuel sur le contenu des jeux de données mais aussi leur structure tout en surmontant les problèmes d'hétérogénéité syntaxique rencontrés dans les données tabulaires.

Cette thèse introduit un pipeline complet pour l'extraction, l'interprétation et les applications de données tabulaires à l'aide de graphes de connaissances. Nous revisitons tout d'abord la définition des données tabulaires du point de vue de leur interprétation et nous développons des systèmes de collecte et d'extraction de tables sur le Web et dans des fichiers locaux. Nous proposons ensuite trois systèmes d'interprétation de données tabulaires basés sur des règles heuristiques ou sur des modèles de représentation de graphes, afin de relever les défis observés dans la littérature. Enfin, nous présentons et évaluons deux applications d'augmentation des tables tirant parti des annotations sémantiques produites : l'imputation des données et l'augmentation des schémas.

Contents

Contents

List of Figures

---

[1]https://fr.wikipedia.org/wiki/Championnat_d%27Europe_de_football_2008#1er_tour_-_phase_de_groupes

List of Tables

List of Abbreviations

AQ Ambiguity Quality.

CEA Cell-Entity Annotation.

CPA Columns-Property Annotation.

CSV Comma-Separated Values.

CTA Column-Type Annotation.

GBDT Gradient Boosted Decision Tree.

GP Global Precision.

HTML HyperText Markup Language.

JSON JavaScript Object Notation.

KD tree K-Dimensional tree.

KG Knowledge Graph.

LOD Linked Open Data.

LSTM Long Short-Term Memory.

MLP Multi-Layer Preceptron.

NER Named-entity recognition.

PA Precision inside Ambiguities.

RDF Resource Description Framework.

RF Random Forest.

STI Semantic Table Interpretation.

SVM Support Vector Machine.

TF-IDF Term Frequency–Inverse Document Frequency.

UI User Interface.

URL Uniform Resource Locator.

Chapter 1

Introduction

Structured data sources such as CSV, TSV, or spreadsheet files, are one of the main assets being used to train AI algorithms. On the Web, one can find a large number of Web tables and the number is continually growing. For example, in 2018, [22] extracted 14.1B HTML tables from Google's general-purpose web crawl. This data is the treasure of digital services, as long as data is cleaned, unbiased, and combined with explicit and machine-processable semantics in order to foster exploitation.

The tabular data format is relatively easy to access for humans and machines because there is a structure behind it. For example, in relational databases, attributes from the same concept are organized in the same column, allowing users to extract the desired attributes by querying the column index. However, it is not like textual data which is arguably self-explicit: all the necessary information to understand the content is in the sentence or its surrounding paragraphs, with a structure built from known rules. Structured data like tables have latent meanings that humans can only understand through implicit mechanism (inference) in light of their own knowledge, as there is often no explicit context. Hence, interpreting tabular data is a difficult task, and it has attracted a lot of attention in recent years, with, in particular, the crystallization of research efforts around challenges such as the SemTab series [36, 63, 64].

This thesis manuscript mostly focuses on tables with meaningful content (some tables are only designed for layout usage but do not contain meaningful information), researching how to understand and exploit information from tables. The main goal is to investigate strategies for table extraction, interpretation with Knowledge Graph (KG), and possible downstream tasks following the interpretation of tables.

1.1   Motivation

One of the levers for creating innovative services is the correlation and analysis of heterogeneous data from internal or external sources. With the advent of Open Data, a large number of data sources are made available to communities (including data scientists and data analysts). In particular, structured data sources (CSV, JSON, XML, ...) are the ubiquitous raw material for many data science processes. Moreover, the large number of available datasets makes it difficult for the user to identify the most relevant datasets for particular use cases. Thus, we have recently seen the emergence of specialised search engines for datasets, Google Dataset Search [13, 93] being a prominent example.

Large parts of the knowledge of companies are encoded in tabular data. This work has been done in the context of the Orange[1], one of the leading companies in the domain of telecommunication. Being able to interpret such data is the key to increase business efficiency and to propose innovative services, and Orange is no exception. With more than 140,000 employees worldwide and a heterogeneous client portfolio, Orange produces a phenomenal amount of tabular data every day. These tables are viscerally embedded in internal services and products (e.g., network logs, multimedia catalogs). Hence, they are a source to discover new knowledge. Although this encourages the development of efficient tools to process them, several issues are negatively impacting their use. First, the volume curse makes it difficult to identify the right dataset for a given use case. Then, the knowledge gap between data producers/consumers is exacerbated by our language footprint (seven main languages), the heterogeneous tools producing various table formats, and the experience/jobs of employees leading to similar concepts being expressed by different terms across tables.

Semantic models (in particular, ontologies) allow the implicit meaning of data to be represented explicitly by specifying the concepts and relationships present in the data. The provision of semantic labels on datasets facilitates the understanding and reuse of data by providing documentation on the data that can be easily used by a non-expert. Moreover, semantic annotation opens the way to search modes that go beyond simple keywords and allow the expression of queries of a high conceptual level concerning the content of the datasets but also their structure while overcoming the problems of syntactic heterogeneity encountered in tabular datasets.

However, the practice of annotating data using semantic models remains very marginal because it is a tedious and time-consuming task requiring expertise in the business domain and in Semantic Web tools. Semantic annotation is a process consisting of matching attributes, headers or values on the one hand, with classes or instances from ontologies on the other hand. The annotation work also makes it possible to determine the relationships between

---

[1]https://www.orange.com/en

attributes.

From a human perspective, tables are intuitively easy to understand (just by looking at them), but for a machine, they are difficult to interpret because of their graphical nature: all the little visual details (such as delimiters and orientation) matter a lot. At the same time, the tables themselves usually just contain the data (e.g. statistics), while the surrounding text (headers, captions and description) gives this data meaning. Hence, tabular data is challenging to interpret by machines because of the limited context available to resolve semantic ambiguities, the layout of tables that can be difficult to handle, and the incompleteness of KGs in general.

Current fleet							
Lines	**Manufacturer**	**Class**[a]	**Image**	**Number**	**Car numbers**	**Built**	**Notes**
BART main lines	Rohr	A		59	1164–1276	1968–1975	
		B		380	1501–1913	1971–1975	To be phased out by August 2023 and replaced by the "D" and "E" cars.[70]
	Alstom	C1		150	301–450	1987–1989	
	Bombardier	Movia D		310	3001–3310	2012–	Order being filled/testing, entered service on January 19, 2018.
		Movia E		465	4001–4465		
Oakland Airport Connector	Doppelmayr Cable Car	Cable Liner		4	1.3–4.3	2014	automated guideway transit trainsets
eBART	Stadler	GTW		8	101–108	2016	diesel multiple units

Figure 1.1: An example of a Web table about lines of the railway system in California.

The main idea to make tabular data intelligently processable by machines is to find correspondences between the elements composing the table with entities, concepts, or relations described in KG which can be of general purposes such as DBpedia [19] and Wikidata [128], or enterprise-specific. This problem is known as Semantic Table Interpretation (STI) or Semantic Table Annotation. Classical Natural Language Processing (NLP) tasks for unstructured text handle poorly such tables since they do not leverage the table structure and the underlying semantics [153]. For example, in the table depicted in Figure 1.1, the mention "Rohr" is ambiguous as it can refer to a surname (Q16882196), a manufacturer (Q2391081), or a municipality in Germany (Q583512). However, this ambiguity can be resolved when taking into account the table structure and, in particular, the fact that the "Manufacturer" column only contains companies. KGs can be used to drive the semantic interpretation of tabular data while being themselves the artifacts that can be further enriched from the result of the interpretation process. In this latter case, tabular data becomes a means to either populate a nascent KG or improve the quality of an established one.

Adding a semantic layer on top of tabular data, in order to make the latent meaning explicit and exploitable through a structured and shared format, is an invaluable step towards effi-

cient and intelligent use of data. It opens up opportunities for new semantic-based services: leverage semantic annotation to better index datasets in search engines [15, 23], improve question/answering systems [95, 116, 141], enrich knowledge bases [104, 139, 150] or enhance dataset recommendation [148].

1.2 Research Questions

As seen before, tabular data provide information following a large variety of representation forms. To enable the interpretation and the usage of the rich knowledge contained in these tables at the end, we propose the following research questions starting from the definition of tables to the downstream tasks of semantic table interpretation:

- Wild tables are diverse in their structure and heterogeneous in their contents. What are the different table types that one can encounter on the Web? Can these table types be automatically recognized? How can we extract and represent some basic table features such as the presence of a header or the table orientation to ease the interpretation of tables?

- Given a target knowledge graph and tabular data sources, what possible correspondences can we establish between the elements from tables and knowledge graphs?

  - Can the overlapped information from tables and knowledge graphs support the matching between the two sides?

  - Elements within a table are related to each other and these relationships are sometimes captured in other resources such as language models or encyclopedic knowledge graphs. Are graph embeddings and language models capable of revealing the relatedness between the neighboring cells? Can these latent relationships between the table elements also reveal the shared topics from these tables to improve the disambiguation?

- Finally, what downstream tasks (e.g. cell filling, schema augmentation) could be realized following a semantic table interpretation process, and how can semantic table interpretation can be leveraged in these downstream tasks?

1.3 Summary of Contributions

The work during this thesis has led to contributions related to the extraction, the interpretation, and the augmentation of tabular data as follows:

- First, based on existing categorization, we propose a new categorization that reflects

the heterogeneity of table types that one can encounter, revealing different challenges that need to be addressed. Our work further refreshes the definition of tables from the perspective of table interpretation. Facing the numerous tables in the wild, we developed two systems for collecting, managing, and visualizing tables from Web and local files.

- We proposed a pre-processing system for extracting some basic features (e.g., the orientation and the position of the header) of a given relational table.

- We review five major sub-tasks that STI deals with even if the literature has mostly focused on three sub-tasks so far. We review and group the many approaches that have been proposed into three macro families and we discuss their performance and limitations on various datasets and benchmarks proposed by the community.

- We have proposed three Semantic Table Interpretation (STI) systems based on different techniques, which include a continually updating heuristic system namely DAGOBAH SL, and two embeddings systems that aim to leverage KG embeddings for capturing the common theme of the table with the help of unsupervised clustering algorithm and a hybridizing between embeddings and heuristic scores.

- The semantic annotations generated by the STI approaches are as many mapping with the knowledge graph which are beneficial to table augmentation. To study table augmentation, we developed an approach for filling missing cells and adding additional columns to the table.

1.4   Thesis Outline

The remainder of this thesis is organized into five chapters:

- Wild tables are diverse in their structure and heterogeneous in their contents. In Chapter 2, we first define what a table is and what are the types of tables. We also introduce in this chapter our efforts in collecting tables from the Web and our contributions to automatically extracting basic features from relational tables, such as the orientation of the table, the position of the subject column, and the position of the headers.

- In Chapter 3, we aim to cover the up-to-date STI tasks and methods. The table interpretation tasks are divided into five types. We also provide a generic pipeline for the table interpretation, and we survey the existing datasets and benchmarks. Finally, this chapter reviews and groups the approaches into three families and discusses their performance and limitations.

- Chapter 4 introduces DAGOBAH SL, a heuristic table annotation system that aims to map table components (e.g., table cell) with KG elements (e.g., entity). We report the evaluation of this system in SemTab challenge series. Also, the industrialisation of DAGOBAH SL with DAGOBAH API and DAGOBAH UI is introduced.

- Facing numerous limitations with heuristic systems, in Chapter 5, we propose two systems that use KG embeddings, namely DAGOBAH Embeddings and Radar Station, to drill down further into the hidden table topics to improve disambiguation performance.

- Finally, we devote Chapter 6 to the applications after STI. In this chapter, we introduce two tasks: data imputation for filling missing values in the empty cell of the table, and schema augmentation for adding additional columns to the table.

Chapter 2

Collect and Prepare Tables at Scale

Tables often constitute a major source of information since large parts of both companies internal repositories and web pages are represented in tabular formats. Over the last years, with the growing number of tabular data that are being used in the Web and local documents, collecting and extracting these tables gained increasing attention from researchers. Unlike unstructured data (e.g., the plan text), tabular data uses the table structure to manage the information. Hence, understanding the layout and construction rules of a table is crucial. However, as there is an important heterogeneity of tables considering their layout, provenance, and usage, the collection and extraction of tables become a complex task.

In this chapter, we first propose a new fine-grained classification based on existing classifications with a deeper analysis of relational tables in Section 2.1. Later, Section 2.2 provides a list of metadata elements attached to the table carrying semantic information useful for interpretation. In Section 2.3, we introduce two systems, Harvesting Tables in the Wild (HTW) and CorpusWalker, for collecting and managing tables from Web and local files. Finally, we introduce the pre-processing module from DAGOBAH system in Section 2.4, where three features are extracted by the pre-processing module targeting relational tables: i) the orientation of a given table, ii) the existence of headers, and iii) the position of the subject column.

2.1   Defining Tables

A table is a two-dimensional arrangement of data with n lines and m columns. This enables a compact visualization for reading. A cell is the basic element of a table where $\mathcal{T}_{ij}$ ($0 \leq i \leq n-1, 0 \leq j \leq m-1$) indicates the cell from row i and column j of table $\mathcal{T}$. Tables are highly heterogeneous in terms of structure, content, and purpose. Therefore, in need to interpret a table, it is important to identify its type so that potential specificities can be taken into account in the STI process.

We introduce a multi-level classification of tables based on several aspects where our contribution includes the categorisation of the existing works into three sub-dimensions (Structure, inner-relationship, and orientation) and a deeper fin-gained classification of relational tables. This classification of tables is intended to make it easier to define the scope of STI approaches proposed in the literature and to help identify the challenges related to STI tasks. The first classification effort splits tables into two high-level categories: genuine and non-genuine [96, 133]. Genuine tables are firstly defined as two-dimensional structures with simple cells (i.e. short and without any complex structures) and a high level of relativity (syntactically and semantically) within rows and columns in [96]. Non-genuine tables are structures used to group contents for easy viewing [133]. One limitation of this dichotomy is that it does not consider tables with long and complex cell contents which are still semantically coherent. For example, we can observe cells containing a list of comma-separated entities (row "Celebration") or mixing text and entities (row "Significance") in the infobox depicted in Figure 2.4(b). According to the definition provided in [96] and used in [102], this table will not be considered a genuine table while, arguably, this table carries semantic information worth being processed.

In more recent works, [34, 67] proposed two similar classes associated with a set of sub-classes: relational knowledge tables including vertical and horizontal listings, attribute/value table, matrix, etc. and layout tables including tables used for navigation and formatting purposes. This classification focuses mostly on relational knowledge and is therefore not comprehensive enough to cover all possibilities. For example, some tables do not have inter-relation between table elements and are not for layout purposes either. This is the case of the table depicted in Figure 2.4(d) where there is no information about the common relationship between each cell.

To cope with these shortcomings, we propose a new classification of table types, shown in Figure 2.1, that rely on the existing work presented in [39, 67, 68, 96, 102, 135, 149] and consider overlapping dimensions. This classification also contributes to better identification of relational tables in embracing their diversity.

We first consider that tables can be separated into two broad categories:

- **Layout tables** are used to format web pages. Elements of these structures are not semantically consistent and are not linked by semantic relationships. They are used to visually organize the content of a page to maximise user comfort and site usability. A layout table used on Amazon to provide the order interface is given in Figure 2.2.

- **Genuine tables** represent in rows or columns human-understandable knowledge. In the literature, genuine tables are said to be short and without complex structure [96]. We relax this concept by considering that tables with a high level of relativity (syntactically and semantically) within rows and columns are genuine tables, without considering the table complexity. For example, in Figure 2.2, the semantic of the two genuine tables is
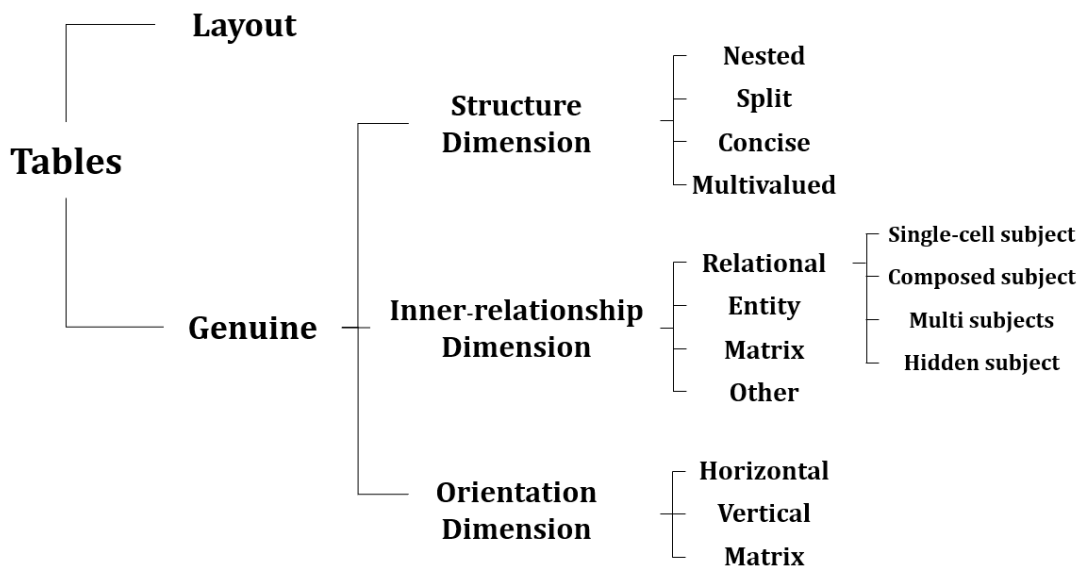
Figure 2.1: Classification of table types with a finer-grained analysis of genuine tables along three dimensions: structure, inner-relationship, and orientation.

Figure 2.2: Illustration of genuine tables and layout tables on the Amazon Website.[a]

---

[a]https://www.amazon.com/Furinno-14035EX-Study-Table-Espresso/dp/B00NIYX9LC

the description attributes (e.g. price, color) of the product.

Genuine tables contain relational knowledge that should be machine-interpretable, and thus, they constitute valid inputs for the STI process. On the contrary, layout tables are convenient for improved visual presentation but the semantic association between their cells is relatively sparse. Hence, they are not eligible for knowledge extraction and interpretation.

Previous works have also proposed to classify tables starting from different entry points and further segment genuine tables along different dimensions [39, 67, 102, 149]. However, the state of the art considers that these table types are mutually-exclusive which does not cover the heterogeneity and complexity of genuine tables. Consequently, we propose to categorize genuine tables using three non-mutually exclusive dimensions: structure, inner relationship, and orientation. Table types are then formed by a composition of these dimensions. For example, the table depicted in Figure 2.3(c) about railway lines is a concise table (structure dimension), a horizontal table (orientation dimension), and a composed-subject relational table (inner-relationship dimension). In the following sections, we further define each of these three dimensions.

2.1.1 Structure Dimension

[67] focuses on the layout structure of a table, which is mainly reflected in the table elements' composition. Accordingly, the subclass "Structure Dimension" of our classification is divided into the following four types of tables illustrated in Figure 2.3:

- **Nested tables** contain one or more tables in one or more of their cells. Figure 2.3(a) depicts a nested table as the main table contains a table about risk levels of hazardous materials in one of its cells.

- **Split tables** contain sub-tables with cells that are independent of those in the other sub-tables. [67] defines split tables as a sequential repetition of rows or columns. We enforce this definition by defining split tables as tables that can be split into sub-tables. To illustrate, in the table in Figure 2.3(b), the infobox of the city of Chicago is composed of several sub-tables. Each sub-table describes one concept of the subject of the original table, e.g. location, area, and population of the main Chicago entity.

- **Concise tables** contain merged cells to avoid repetitions of cells referring to the same content in rows and/or columns. In the table presented in Figure 2.3(c), the first cell of the column "lines" merges six individual cells with the same value "BART main lines".

- **Multivalued tables** contain multiple values in a single cell. For example, in Figure 2.3(d), the cells of the column "lines used" contain a list of route lines.

Fire diamond

Standard Representation	Tabular Representation			
	Risk levels of hazardous materials in this facility			
	Health Risk	Flammability	Reactivity	Special
	Level 3	Level 2	Level 1	

(a)

Country	United States
State	Illinois
Counties	Cook, DuPage
Settled	circa 1780
Incorporated (town)	August 12, 1833
Incorporated (city)	March 4, 1837
Founded by	Jean Baptiste Point du Sable
Government	
• Type	Mayor–council
• Body	Chicago City Council
• Mayor	Lori Lightfoot (D)
• City Clerk	Anna Valencia (D)
• City Treasurer	Melissa Conyears-Ervin (D)
Area[3]	
• City	234.21 sq mi (606.60 km^2)
• Land	227.41 sq mi (588.98 km^2)
• Water	6.80 sq mi (17.62 km^2) 3.0%
• Urban	2,122 sq mi (5,496 km^2)
• Metro	10,874 sq mi (28,160 km^2)
Elevation[2] (mean)	597.18 ft (182.02 m)
Highest elevation – near Blue Island	672 ft (205 m)
Lowest elevation – at Lake Michigan	578 ft (176 m)
Population (2010)[6]	
• City	2,695,598
• Estimate (2019)[7]	2,693,976
• Rank	3rd, U.S.
• Density	11,846.55/sq mi (4,573.98/km^2)
• Urban	8,667,303[5]
• Metro	9,533,040 (3rd)[4]
• CSA	9,901,711 (US: 3rd)[4]

(b)

(c)

Lines	Manufacturer	Class	Image	Number	Car numbers	Built	Notes
BART main lines	Rohr	A		59	1164–1276	1968–1975	
	Rohr	B		380	1501–1913	1971–1975	To be phased out by August 2023 and replaced by the "D" and "E" cars.[65]
	Alstom	C1		150	301–450	1987–1989	
	Morrison-Knudsen	C2		80	2501–2580	1994–1996[66]	
	Bombardier	D		310	3001–3310	2012–	Order being filled/testing, entered service on January 19, 2018.
	Bombardier	E		465	4001–4465	2012–	
Oakland Airport Connector	DCC Doppelmayr	Cable Liner		4	1.3–4.3	2013	automated guideway transit trainsets
eBART	Stadler	GTW		8	101–108	2014-2018	diesel multiple units

(d)

Route name	First service	Lines used	Service times
Berryessa/North San José–Richmond line	September 11, 1972	R-Line, K-Line, A-Line, S-Line	Operates during all service hours.
Antioch–SFO/Millbrae line	May 21, 1973	C-Line, K-Line, M-Line, W-Line, Y-Line, eBART	Through-routed with the SFO–Millbrae line on Sundays. Uses DMU technology from Antioch to Pittsburg/Bay Point.
Berryessa/North San José–Daly City line	September 16, 1974	S-Line, A-Line, M-Line	No evening or Sunday service.
Richmond–Millbrae line	April 19, 1976	R-Line, K-Line, M-Line, W-Line	Terminates at Millbrae on weekdays and at Daly City on Saturdays; no evening or Sunday service.
Dublin/Pleasanton–Daly City line	May 10, 1997	L-Line, A-Line, M-Line	Operates during all service hours. Some Sunday service terminates at Montgomery Street station.
SFO–Millbrae line	February 11, 2019 (previously 2003–2004)	W-Line, Y-Line	Through-routed with the Antioch–SFO/Millbrae line on Sundays.
Coliseum–Oakland International Airport line	November 22, 2014	Separate elevated automated guideway transit line (H-Line) not connected to other BART tracks	Operates during all service hours.

Figure 2.3: Examples of different structures of tables: (a) a nested table[a], (b) a split table[b], (c) a concise table (column "lines")[c], (d) a multivalued table (column "lines used")[d].

---

[a]https://en.wikipedia.org/wiki/Table_(information)

[b]https://en.wikipedia.org/wiki/Chicago

[c]https://en.wikipedia.org/wiki/Bay_Area_Rapid_Transit#Rollingstock

[d]https://en.wikipedia.org/wiki/Bay_Area_Rapid_Transit#Infrastructure

2.1.2 Inner-relationship Dimension

The inner-relationship dimension considers the topology of the semantic connection between the cells. [34] first gave a detailed classification of relational knowledge tables into listings, attribute or value tables, matrices, enumerations, and forms. [39] has extracted Web tables that are classified into listings, matrices and other tables to capture enumerations, calendars, etc. [68, 102] have extended this categorization with a fourth kind named "entity tables" while listings are considered as relational tables.

Accordingly, we propose the following types:

- **Relational** tables are structures in which each row (resp. column) provides information about a specific entity, and the corresponding columns (resp. rows) represent attributes that describe the entity. Hence, relational tables are oriented, either horizontally or vertically, depending on the arrangement of the entities and their attributes in the table. If the rows of a relational table contain the entities and the columns the attributes, then the table is horizontal. Otherwise, it is vertical. Relational tables may have a

Name	Pinnacle height	Year	Country	Town	Remarks
Tokyo Skytree	634 m (2,080 ft)	2011	Japan	Tokyo	
Kyiv TV Tower	385 m (1,263 ft)	1973	Ukraine	Kyiv	
Dragon Tower	336 m (1,102 ft)	2000	China	Harbin	
Tokyo Tower	333 m (1,093 ft)	1958	Japan	Tokyo	
WITI TV Tower	329.4 m (1,081 ft)	1962	United States	Shorewood, Wisconsin	
St. Petersburg TV Tower	326 m (1,070 ft)	1962	Russia	Saint Petersburg	

(a)

Bastille Day

Fireworks at the Eiffel Tower, Paris, 2017

Also called	French National Day (*Fête nationale*) The Fourteenth of July (*Quatorze juillet*)
Observed by	France
Type	National day
Significance	Commemorates the Storming of the Bastille on 14 July 1789,[1][2] and the unity of the French people at the Fête de la Fédération on 14 July 1790
Celebrations	Military parades, fireworks, concerts, balls
Date	14 July
Frequency	Annual

(b)

(c)

Perceived Produced	i	e	a	o	u
i	15		1		
e	1		1		
a			79	5	
o			4	15	3
u				2	2

(d)

Demonstrative	Relative	Indefinite	Interrogative
this	who / whom / whose	one / one's / oneself	who / whom / whose
these	what	something / anything / nothing (things)	what
that	which	someone / anyone / no one (people)	which
those	that	somebody / anybody / nobody (people)	
former / latter			

Figure 2.4: Examples of different inner-relationships of tables: (a) a horizontal relational table[a], (b) an entity table[b], (c) a matrix table[c], (d) an enumeration table[d] which belongs to "other genuine tables".

---

[a]https://en.wikipedia.org/wiki/Eiffel_Tower

[b]https://en.wikipedia.org/wiki/Bastille_Day

[c]https://en.wikipedia.org/wiki/Whistled_language#Lack_of_comprehension

[d]https://en.wikipedia.org/wiki/Pronoun#English_pronouns

header, usually in the first row or the first few rows for horizontal tables. As an example, Figure 2.4(a) depicts a horizontal relational table where each row describes a tower with its attributes (e.g. height, year, country, and town).

- **Entity** tables, also known as attribute-value tables, are used to describe a unique entity. An entity table enumerates the attributes of the entity. Infoboxes from Wikipedia are examples of entity tables. For example, the distinct attributes and their values for the entity "Bastille Day" are shown in the table in Figure 2.4(b). It should be noted that entity tables could also be seen as two-column vertical or two-row horizontal relational tables.

- **Matrix** tables present a two-dimensional arrangement of data that should be read simultaneously horizontally and vertically. A matrix associates pairs (row, column) with cell values through one unique property for the whole table. Generally, cells contain numeric or boolean values. Figure 2.4(c) shows a vowel confusion matrix that quantifies the understanding of vowels between people. It associates pairs (vowel produced, vowel perceived) with the number of persons having this perception of the produced vowel. For example, one person perceived an "a" when an "i" was produced. Bold numbers correspond to

(a)

Department	Area (km²)	Population (2011)[37]	Municipalities
Paris (75)	105.4	2 249 975	1 (Paris)
Hauts-de-Seine (92)	176	1 581 628	36 (list)
Seine-Saint-Denis (93)	236	1 529 928	40 (list)
Val-de-Marne (94)	245	1 333 702	47 (list)
Petite Couronne	657	4 445 258	123
Paris + Petite Couronne	762.4	6 695 233	124

(b)

Lines	Manufacturer	Class	Image	Number	Car numbers	Built
BART main lines	Rohr	A		59	1164–1276	1968–1975
	Rohr	B		380	1501–1913	1971–1975
	Alstom	C1		150	301–450	1987–1989
	Morrison-Knudsen	C2		80	2501–2580	1994–1996[67]
	Bombardier	D		*310*	3001–3310	2012–
	Bombardier	E		*465*	4001–4465	2012–
Oakland Airport Connector	DCC Doppelmayr	Cable Liner		4	1.3–4.3	2013
eBART	Stadler	GTW		8	101–108	2014-2018

(c)

Release year	Album	Artist/s	Nationality	Worldwide sales (in millions)	Ref(s)
2002	*Come Away With Me*	Norah Jones	🇺🇸 United States	23.9	[3]
2000	*The Marshall Mathers LP*	Eminem	🇺🇸 United States	23.29	[4]
2002	*The Eminem Show*	Eminem	🇺🇸 United States	22.95	[5]
2000	*Hybrid Theory*	Linkin Park	🇺🇸 United States	20.8	[6]
2015	*25*	Adele	🇬🇧 United Kingdom	20.41	[7]

(d)

3	11 juin 1998	Italie 🇮🇹	2 - 2	🇨🇱 Chili
4	11 juin 1998	Cameroun 🇨🇲	1 - 1	Autriche
19	17 juin 1998	Chili 🇨🇱	1 - 1	Autriche
20	17 juin 1998	Italie 🇮🇹	3 - 0	Cameroun
33	23 juin 1998	Italie 🇮🇹	2 - 1	Autriche
34	23 juin 1998	Chili 🇨🇱	1 - 1	Cameroun

Figure 2.5: Examples of different relational tables: (a) a single-cell-subject relational table[a], (b) a composed-subject relational table[b], (c) a multi-subject relational table[c], (d) a hidden-subject relational table[d].

---

[a]https://en.wikipedia.org/wiki/France#Major%20cities
[b]https://en.wikipedia.org/wiki/Bay_Area_Rapid_Transit
[c]https://en.wikipedia.org/wiki/List_of_best-selling_albums_of_the_21st_century
[d]https://fr.wikipedia.org/wiki/Coupe_du_monde_de_football_1998

correct identifications.

- **Other** genuine tables contain semantic information but do not fit within the aforementioned types. Tables in this class include enumerations and calendars. To illustrate, Figure 2.4(d) is an enumeration table where each column is an independent enumeration of pronouns according to a pronoun type.

The literature considers relational tables as a leaf in the proposed table type taxonomies [68, 102, 135]. However, relational tables exhibit an important diversity, especially in the representations of entities. We propose to further classify them depending on the characteristics of their subjects. The **subject** of a row of a horizontal relational table (resp. column of a vertical relational table) is an entity that is described by the collections of cells in this row (resp. column). For example, in the table depicted in Figure 2.4(a), the entity "Tokyo Skytree" (Q57965) is the subject of the first row as it is described by the other entities of this row: "2011" (Q1994), "Japan" (Q17), and "Tokyo" (Q1490).

We introduce four subtypes of relational tables (Figure 2.5):

- **Single-cell-subject** tables associate each row of a horizontal table (resp. column for a vertical table) to a single subject. Labels of subjects are given in a single column (resp.

row). To illustrate, in Figure 2.5(a), the column "department" contains the subjects. The other columns describe the subjects.

- **Composed-subject** tables require the combination of multiple cells to form the subject of each row (resp. column). For instance, in a table that describes persons with first and last names in different columns, it is necessary to merge these two columns to get the complete identifiers of entities. Similarly, in the table shown in Figure 2.5(b), one can identify subjects (particular train classes) by merging columns "Lines", "Manufacturer" and "Class".

- **Multi-subject** tables contain cells that refer to different subjects while being in the same row. In Figure 2.5(c), a row is composed of two subjects: "Artist/s" is the subject of the column "Nationality" while "Album" is the subject of columns "Release year", "Artist/s", "Worldwide sales", and "Ref(s)".

- **Hidden-subject** tables do not explicitly mention the subject of each row (resp. column). For example, in Figure 2.5(d), each row describes the result of a football match, but the mention of the match itself is not made explicit in the table.

2.1.3   Orientation Dimension

The orientation dimension considers the direction of the relationships inside a table [39, 67]. Indeed, knowing the direction of relationships within a table simplifies its interpretation, e.g. to read the attributes describing a subject.

- In **Horizontal** tables, subjects are described horizontally, which means that each row describes a different subject. For example, in Figure 2.4(a), the subject "Dragon Tower" and its attribute "Harbin" are in the same row.

- In **Vertical** tables, subjects are described vertically, which means that each column describes a different subject. An example of a vertical table is depicted in Figure 2.4(b), where the attributes of the entity "Bastille Day" are in the same column.

- **Matrix** tables are defined as for the inner-relationship dimension. They cannot be interpreted row by row or column by column but rather cell by cell while simultaneously considering both horizontal and vertical headers. For example, the matrix in Figure 2.4(c) should be interpreted cell by cell while taking into account both horizontal and vertical headers to read the number of persons that have a specific perception of a produced vowel.

2.1.4 Table Types Statistics

We introduced in Figure 2.1 a multi-dimensional and fine-grained classification of table types. In this section, we aim to survey how frequent these types of tables can be encounter in the wild. DWTC [39] has randomly selected 26,645 tables from the WDC Web Table Corpus [68] and has concluded that the resulting corpus was made of 96% layout tables and 4% genuine tables. Regarding the structure dimension, [67] has extracted 342,795 Web tables (from various Websites starting from Wikipedia, e-commerce, news and university Web sites) and has identified that 75.5% of the tables are for layout while the remaining tables are relational knowledge tables. [67] has also provided the distribution of nested tables, split tables, concise tables, and multivalued tables among the relational knowledge tables, which are respectively 3.7%, 2.6%, 12.9%, and 74.9%. Regarding the inner-relationship dimension, [68] applied the DWTC framework on the 233 million Web tables of the WDC Web Table Corpus to detect the type of each table w.r.t the inner-relationship dimension. Results show that relational tables, entity tables, and matrices respectively constitute 39%, 60%, and 1% of the corpus.

Regarding the orientation, the distribution of horizontal tables and vertical tables is 54.9% and 45.1% for entity tables, and 94% and 6% for relational tables in the WDC Web Table Corpus. In [67], the authors show that 70% of the relational knowledge tables are horizontal.

Our proposed classification goes further into the details. However, identifying some table type such as hidden-subject tables remains an open scientific challenge. To date, we have not identified an approach to automatically classify tables with a level of granularity close to the classification proposed in this paper.

2.2 Table Metadata

Tables do not always appear alone in real-life scenarios. Alongside the data contained in a table, metadata and the context in which a table appears are also valuable information for STI. For example, if a table has been published on a Web page describing the Bundesliga, it is probably more relevant to football than any other sport. Hence when extracting the table, it would be useful to collect both the table itself and its metadata.

Several STI works stress that one of the challenges to be addressed is the loss of context when annotating a table [149]. Indeed, tables do not constitute the unique source of information that can be used by STI processes since the context in which they appear may provide complementary or novel information. Such non-table information constitutes the metadata of tables and is defined as additional data that can be extracted from information sources to provide additional context for the interpretation. For example, metadata can describe the characteristics and content of the original data, and thus can be used to organize, retrieve,
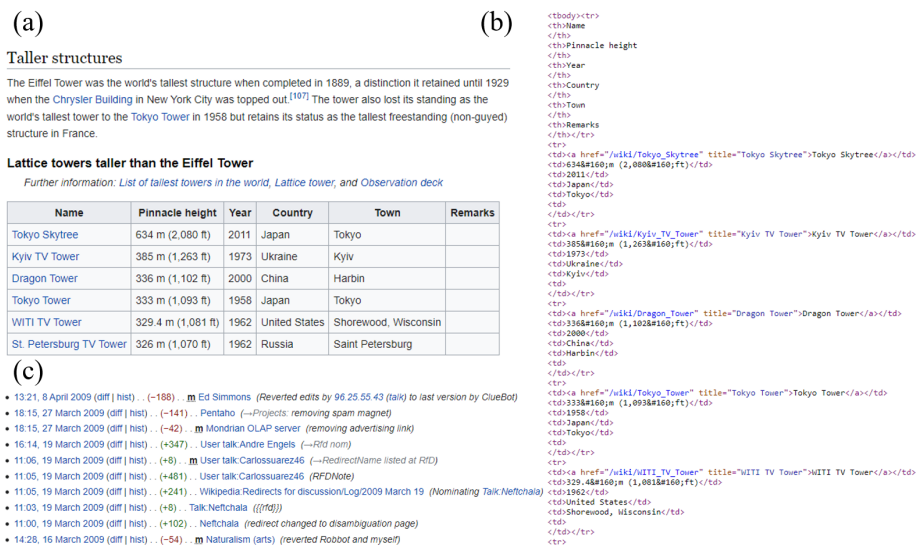
Figure 2.6: Metadata of the Web table "Lattice towers taller than the Eiffel Tower"[a]: (a) descriptive metadata: its surrounding text, (b) structural metadata: `<td>`, `<tr>` and `<th>` tags indicate table cells, row ordering, and the presence of headers, (c) administrative metadata: the page history of the table[b].

---

[a]https://en.wikipedia.org/wiki/Eiffel_Tower

[b]https://en.wikipedia.org/w/index.php?title=Special:Contributions/Andre_Engels

preserve, and manage extracted knowledge units. Depending on its structure, purpose, and provenance, metadata is split into descriptive, structural, and administrative metadata [99]. Such a definition was originally used in digital collection [143] and is applicable to table metadata as well. Each type of metadata in a table context can provide a deeper understanding of the table.

Descriptive metadata is used to describe the target data by providing, e.g. its source, explanatory notes, or other contextual information. For example, the descriptive metadata of the table "Lattice towers taller than the Eiffel Tower" depicted in Figure 2.6(a) can include its provenance (i.e. the Uniform Resource Locator (URL) of the page) and its surrounding text. Indeed, texts surrounding tables are potential sources of contextual information, and thus valuable metadata, since they often explain a nomenclature or verbalize salient information. The different relationships between texts and tables, including titles and captions or even simple co-occurrences between a table and the surrounding texts, are useful indicators to guide and improve the annotation and knowledge extraction processes. However, this table-text complementarity is little used in the STI domain so far. Descriptive metadata provides additional information that enhances the process of approaches such as [17, 38].

Structural metadata describes the structural schema of composite objects or relationships between objects. For example, the `<td>`, `<tr>`, and `<th>` tags in the Web table in Figure 2.6(b)

allow to detect table cells, row ordering and the presence of headers. Such structural patterns could benefit STI approaches such as [121, 130].

Administrative metadata often captures information such as the process of creation or data acquisition for a table. For example, in Figure 2.6(c), the page history details how, when, by whom, and for which purpose data has been produced or altered, allowing to assess the quality and validity of the table. Additionally, the creation or modification date of a table can indicate the freshness of its information, and thus allows to assess the risk of extracting outdated information.

It should be noted that table metadata can appear in different forms since tables have different formats and structures. For example, in some approaches, table headers are available as metadata [25]. Furthermore, if a table element consists of a hyperlink (e.g. hyperlinks in infoboxes of Wikipedia), this mapping relationship also constitutes metadata.

2.3 Collecting Tables

There is an immense number of high-quality tables that can be found on the Web, to name a few, [23] extracted 154 million high-quality relational tables from the Web and [16] extracted 1.6 million relational tables from Wikipedia. With the help of additional operations, such as sorting, filtering, and joins, the rich information from wild tables can be turned into knowledge to support decision-making. However, regarding the different creator's purposes, formats, and structure complexities, it becomes increasingly difficult for a machine to access this data in a meaningful way. In this section, we present our efforts in collecting tables from the Web and the local files. We first introduce related work for collecting tables in Section 2.3.1. Then, we present the student project that we have guided for Web table extraction in Section 2.3.2. Finally, we provide an overview of the project CorpusWalker from Orange in Section 2.3.3.

2.3.1 Related Work

The process of extracting tables from webpages includes filtering irrelevant tables, proper formatting and consistent storage of tables to create a corpus. This includes the classification of tables into one or more semantic (such as subject header, and table header) and/or syntactic (such as row and columns) features. [10] used a collection of simple rules and machine learning classifiers to extract tables with an overall accuracy of 96%. [23] extracted approximately 14.1 billion raw HTML tables from the English documents in Google's main index, where 154 million tables are identified as high-quality relational tables. The authors also proposed a new object derived from the database corpus: the attribute correlation statistics database (AcsDB) that records corpus-wide statistics on co-occurrences of schema elements. [68] leverages

DWTC framework [40] to further extract 1.78 web pages from Common Crawl[1], where 233 million genuine tables are further been identified.

2.3.2 Harvesting Tables in the Wild (HTW)

One of the projects that we worked on with students from EURECOM aims to collect HyperText Markup Language (HTML) tables from the wild as the upper stream application for our semantic annotation system (this system aims to build a semantic layer on top of the input tables to make them interpretable. It will be detailed in Chapter 4 and Chapter 5). We managed to build a Web scraping and table processing pipeline from the ground up. This system provides services for crawling, storing, and posting Web tables. In this section, we present the system created in the framework of the project HTW and report our results.

System

Figure 2.7 provides an overview of this system. The table collection pipeline contains four major components. They are i) a Scrapy[2] Spider module for downloading web pages and extracting tables, which also normalizes table format, ii) a Kafka message queue for buffering the extracted tables, iii) an ArangoDB[3] database for managing the storage, iv) an ingestion service for linking the collected tables with downstream annotation systems and the storage database. In this section, we introduce our efforts in the first three modules, and the table annotation service is described in Chapter 4.

We use Scrapy[4] for crawling Web tables due to its high performance in treating the data stream. Starting from an input Web page, it extracts all hyperlinks (inside HTML <a> tags) from this page and generates new requests for these links. We define a list of whitelisted domains. web pages on these domains will always be crawled. Furthermore, the crawler will follow all links on these web pages. Once the crawler arrives on a Web page that is not in the whitelist, it still crawls that particular Web page but does not follow any links on it. Websites in the whitelist are manually selected from Alexa Top 500 ranking[5], Moz Top 500[6] and CommonCrawl Top 1,000 domains ranked by harmonic centrality [7] where the non-English domains are filtered. We provide the full domain names of the whitelist in Table 2.1. We parse the table with the tool introduced by [82] according to the HTML labels. Layout tables, complex structured tables (such as concise tables or nested tables), and small tables (tables with less than two rows) are

---

[1]https://commoncrawl.org/
[2]https://scrapy.org/
[3]https://cloud.arangodb.com/
[4]https://scrapy.org/
[5]https://www.alexa.com/topsites
[6]https://moz.com/top500
[7]https://commoncrawl.org/2020/10/host-and-domain-level-Web-graphs-julaugsep-2020/
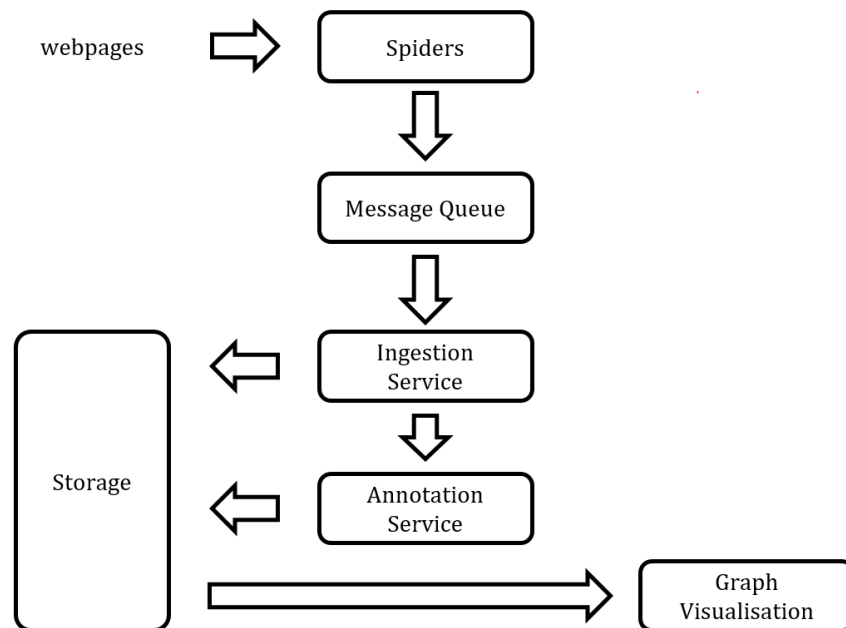
Figure 2.7: Architecture of the project HTW

ignored during the extraction.

Message queues are extensively used for asynchronous communication. A message queue serves as a buffer, meaning that the crawler is never slowed down by a slow ingestion rate (e.g. due to a high load on the database). Moreover, as the ingestion service (consumer) is not interacting directly with the crawler (producer), it does not become a bottleneck and allows the ingestion service to process items out of the message queue at any speed. This gives the ingestion service more headroom to perform time-consuming tasks, such as augmenting the tables with data obtained from external API. Kafka[8] is chosen as the queue engine since it follows a log-committed approach for a message bus and hence can also be used as a temporary store of messages for a desirable unit of time.

Finally, we select ArangoDB database as our storage backend, since it supports more data models (e.g., document, graph, table, full-text) compared to other NoSQL databases such as MongoDB or Neo4j. The stored data format is based on the JavaScript Object Notation (JSON) format merging from data used in approaches [40, 46]. This new format covers the following information from the provenance of the table:

- Page title

- Table title

---

[8]https://kafka.apache.org/

android.com	apache.org
bbc.co.uk	blogspot.com
creativecommons.org	doi.org
en.wikipedia.org	europe.eu
gamepedia.com	github.com
github.io	iana.org
imdb.com	medium.com
merriam-Webster.com	microsoft.com
mozilla.org	nasa.gov
nih.gov	noaa.gov
schema.org	statista.com
w.org	wikibooks.org
wikimedia.org	wikiquote.org
wordpress.com	wordpress.org

Table 2.1: Selected domain names in the whitelist for the crawler.

- Paragraph before/after the table

- Most frequent terms

- Language of the page

Result

We performed a table collection process with our pipeline during three days. With a very limited amount of time, we collected 3,469 tables from 2,002 Websites on 135 unique domains. With these results, we also explore the graph visualization capabilities.

In Figure 2.8, the purple circles represent page vertices (a Web page that has been accessed), while the black circles represent table vertices (a table that has been extracted). The virtual start node is "HTW Start" (where the crawl started), but the start node can be adjusted through the User Interface (UI) to start traversing the graph. Alongside the edges, the edge type is displayed, in this example either "hyperlink" (one Web page links to another Web page) or "page-contains" (a table is embedded on a Web page). The search depth and the maximum number of nodes displayed in the graph can be dynamically adjusted in the UI to facilitate the exploration, though loading large graphs all at once makes the visualization unclear and has performance issues.

2.3.3 CorpusWalker

At Orange, following the same proposal of the project HTW, we have built a data collection tool named CorpusWalker. As illustrated in Figure 2.9, CorpusWalker integrates a series of
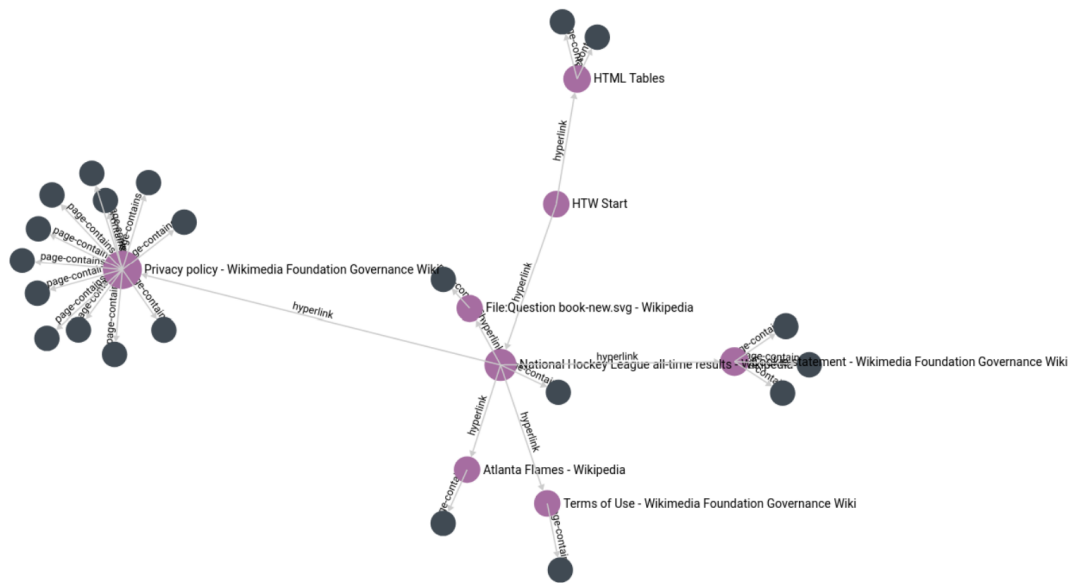
Figure 2.8: Visualisation of provenance topology of the document

functionalities for data collection, data management, search engine, data visualization, etc. It is also compatible with our table annotation tools (detailed in Chapter 4) to provide a data repository for the annotation results. In this section, we explain the system design of CorpusWalker and report the result.
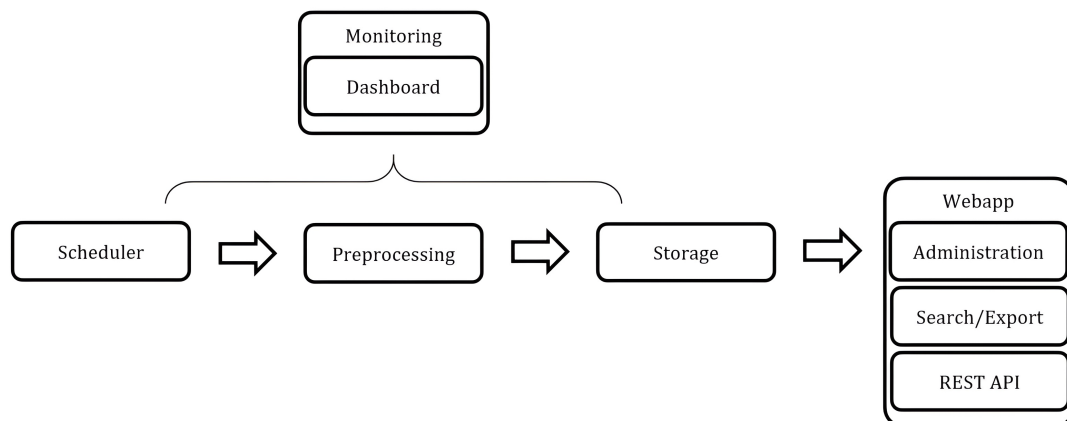
Figure 2.9: Architecture of CorpusWalker

System

CorpusWalker does not only focus on collecting Web tables, but also on ingesting textual data. The scheduler module from Figure 2.9 manages the collection of the documents. Scrapy crawler and MediaWiki REST API[9] are leveraged by CorpusWalker for collecting data from HTML Websites and Wikipedia. In the system, we transform the original data into HTML or JSON files, where each data source is seen as a **corpus**, and each page is a **document**. Figure 2.10 shows the data source that has been collected with CorpusWalker.
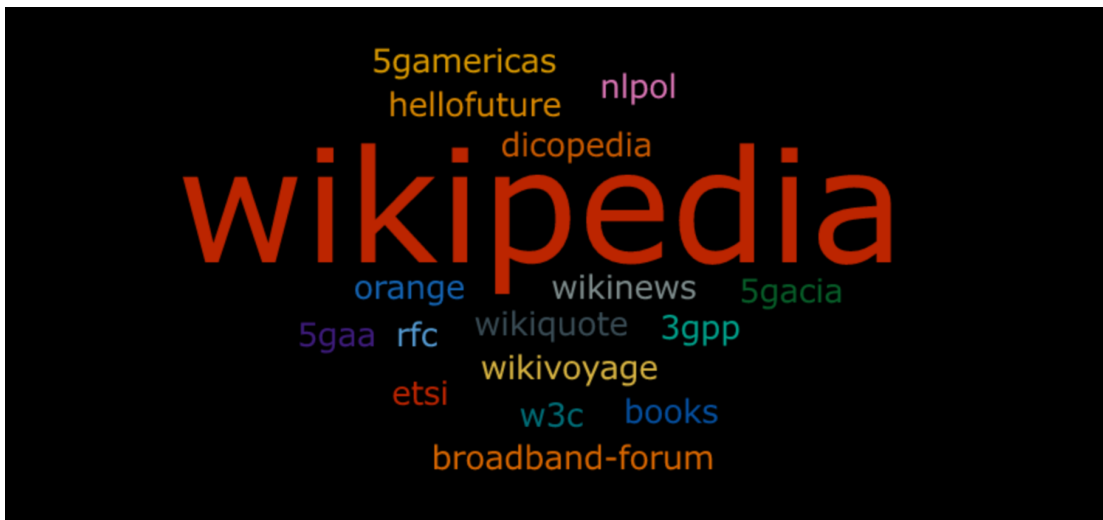
Figure 2.10: Word cloud of the data sources of CorpusWalker
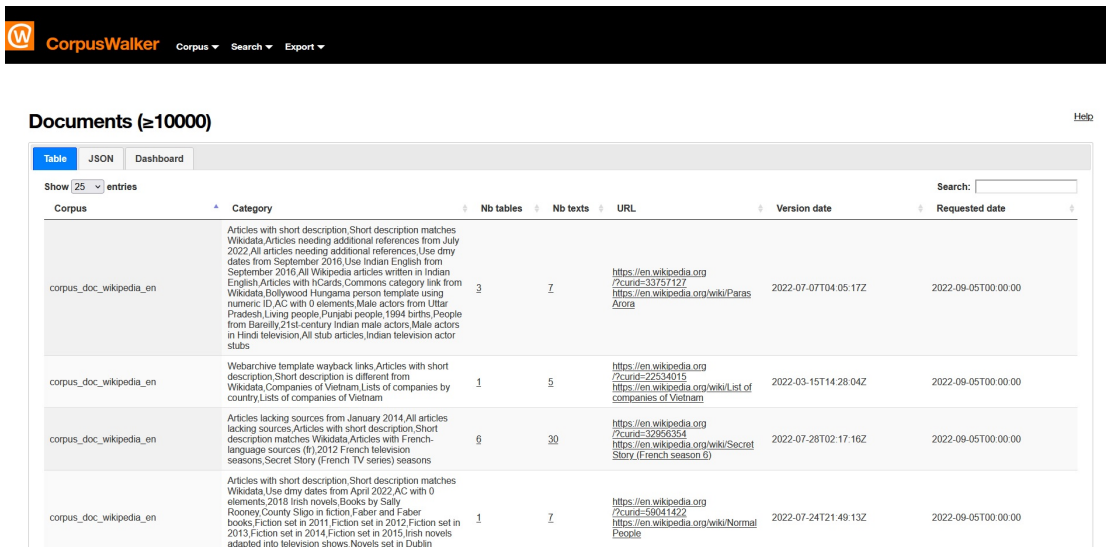
In the Preprocess module (which differs from the table pre-processing introduced in Section 2.4), we aim to extract features from the corpus and add additional information about the dataset. In this module, we first parse each HTML page with BeautifulSoup[10] and each MediaWiki document with Wikitextparser[11], where tables and textual documents are identified and labeled. We further add the content analysis information in the documents which describes the number of rows, columns, and cells for tables, also the word occurrence count in both texts and tables. Finally, we aggregate these documents with semantic annotation where mentions from the document are linked with a given KG, e.g., Wikidata. For tables, we leverage DAGOBAH-API (detailed in Chapter 4) to extract basic features of the table with table pre-processing end-point introduced in Section 2.4, and we annotate the table elements (e.g., cells, columns) with Wikidata by DAGOBAH-SL system (detailed in Chapter 4). For textual documents, we leverage KEFT [111] system for entity linking with Wikidata (KEFT system is not presented since it is not in the scope of the thesis.).

After having generated annotations and computed the statistic analysis about the dataset,

---

[9]https://www.mediawiki.org/wiki/API:REST_API
[10]https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[11]https://wikitextparser.readthedocs.io/en/latest/

Figure 2.11: An example of the administration page of CorpusWalker

we store this additional information with the original documents in a platform powered by Elasticsearch[12]. Elasticsearch is a distributed, free open-source search and analytics engine for all types of data including text, numeric, geospatial, structured, and unstructured data. It is currently widely used in various IT companies. In CorpusWalker, Elasticsearch searches the document according to the input of string characters or Wikidata entities. CorpusWalker will list all associated documents with their statistic with a given input.

All the modules could be supervised by the monitoring module, which provides dashboards for data visualization and execution supervision, also configuration management pages for administration (as the example shown in Figure 2.11). The visualization is powered by Kibana[13], Grafana[14] and StatsD[15] for dashboards. In Figure 2.12, we illustrate interface of document management in CorpusWalker. It provides statistics and information about the documents extracted from the Wikipedia corpus. These information include the number of tables, number of texts, time of the generation, and the collection criteria (in column "Category"). This page also allows users to search the document according to both String text and Wikidata Entities.

Result

Until November 2022, CorpusWalker has collected 9.64 million documents from 19 corpora, where 3.84 million tables have been identified by the system. The total file size after processing is 34.34G. CorpusWalker is capable of handling multiple languages, including English (69.29%),

---

[12]https://www.elastic.co/elasticsearch/

[13]https://www.elastic.co/fr/kibana/

[14]https://grafana.com/

[15]https://www.datadoghq.com/
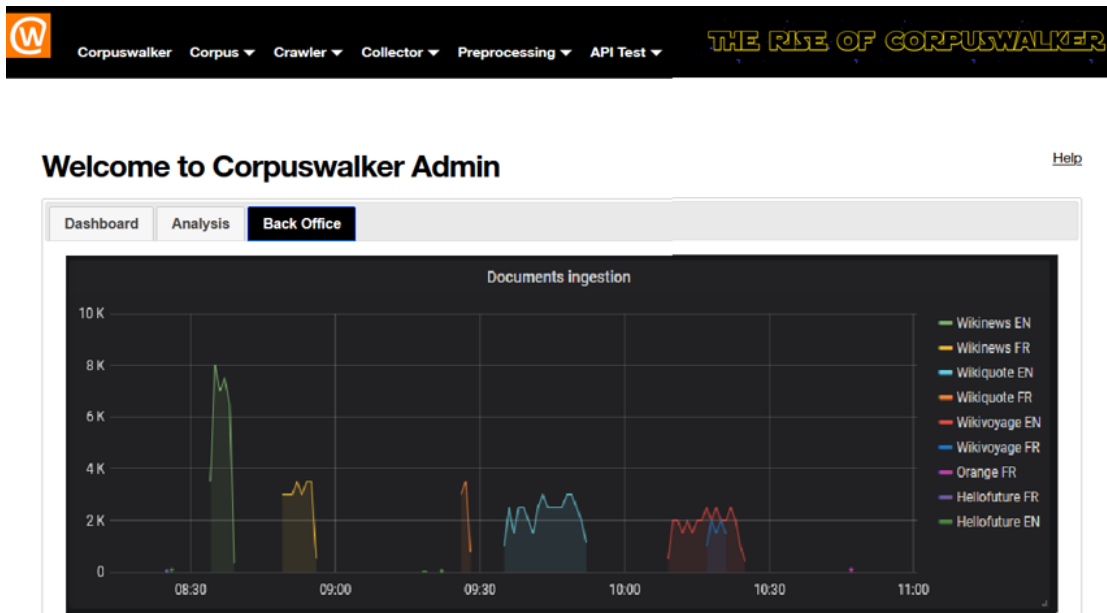
Figure 2.12: Interface of the document management in CorpusWalker

French (26.16%), Romanian (4.51%), and Polish (0.05%). This rich data source allows us to test our system in greater depth and provides a rich source of data for our knowledge base. CorpusWalker is a useful system for data collection, extraction, and management. In the future, we aim to improve the coverage of CorpusWalker by considering more data sources with heterogeneous data formats. Also, optimizing our current table identification and extraction algorithm is another direction that we aim to enhance in future work.

2.4 Pre-Processing Tables

Tables vary in their form and structure as described in Section 2.1. Starting from 0, it is hard to apply a common rule to all tables. For example, one could say one row describes an entity in a horizontal relation table, but the same argument can not be used in vertical tables since a row in a vertical relational table describes an attribute, but not an entity. In order to correctly process the information contained in the tables, it is first necessary to infer several characteristics of the shape of the table. In a context of real exploitation in which there is sometimes little or no knowledge about the tables, the information produced by this chain is decisive for the quality of the annotations. Hence, it is necessary to know some features of the table, such as the orientation, the table type, and the header position, of a given table before any operation in this table.

In this section, we focus on relational tables, and we introduce our first prototype of pre-processing module aiming to extract the orientation, the headers, the key column position,

and each column's primitive type of input table. In Section 2.4.1, we review the current state of the art of similar tasks from the literature. We then describe our system in Section 2.4.2. Finally we analyze our first prototype based on an evaluation of the SemTab dataset and describe our future work in Section 2.4.3

2.4.1 Related Work

Orientation Detection

The orientation detection task aims to assign a given genuine table an orientation, following the table types defined by Section 2.1.3. In most table interpretation approaches, the systems assume that the input table's orientation is always known and horizontal [1, 26, 90, 110, 122].

[97] assumes that the cells in the columns are similar for horizontal tables, and vice versa for vertical tables. A token type is assigned to each cell token, and each cell is transformed into a vector composed of token types from the given cell. Average row/column similarity is calculated by summing up the distance between the last row/column and each preceding row/cols for determining the table orientation. [34] adopts supervised machine learning on top of 21 features extracted from the tables, including the maximum number of rows for each column and the presence of certain HTML tags. The approach labels each table into one of the ten table types, including vertical relation tables, so-called "vertical listing" and "horizontal listing". Gradient Boosted Decision Tree (GBDT) and Support Vector Machine (SVM) are chosen as the classification model and the evaluation shows GBDT achieves a better result. DWTC [39] further refines the feature number to 10 and applies SVM to this approach. DeepTable [50] decomposed the system into two representation models for column-wised and row-wised features, where column-wised (resp. row-wised) features are generated based on an item set from cells in the same column (resp. row). Each cell is transformed into embeddings with the help of Long Short-Term Memory (LSTM) and Multi-Layer Preceptron (MLP) layers. Finally, the representations are concatenated and fed to a softmax classifier for table orientation classification. Although one representation is enough to predict the input table's orientation, adding another representation from another direction allows the model to identify matrices more effectively.

Header Detection

A table header row for a horizontal relational table is the top row or rows that acts as a title for the type of information one can find in each column. It is common to manually bold the top row to signal this information visually. Knowing the presence and the position of table headers separate the schema of the class described by table columns and the data rows. [45] proposes heuristic approaches for calculating a relevance score between the first row and

the rest of the rows for evaluating the difference in font size or data types to predict the presence of the header. It also proposes a machine learning pipeline that extracts row-wised features like a number of characters and sends these features into a classifier, such as SVM and Random Forest (RF). [130] proposes to consider applying a threshold over a score generated by hybridizing the Probase's type and the syntax features (e.g., color, HTML tags, etc.).

Key Column Detection

The key column of a table contains the identifiers of the subjects described by the other columns. Hence, the key column is also known as the subject column while others columns could be seen as the attributes of the given subject. DWTC [39] framework supposes that the key column of a given relational table is the column with the highest unique value number among all literal columns (number and date are not included). MantisTable [32] further considers aggregating features of i) the fraction of empty cells, ii) the fraction of cells with unique content, iii) the distance from the first literal column, and iv) the average number of words in each cell to compute a relevant score for detecting the position of the key column. TableMiner+ [153] introduces a similar approach where the authors also add the Web search score to evaluate the Web frequency of words used in the column header and a given cell from the column.

2.4.2 System Description

The pre-processing toolbox of DAGOBAH, partly based on the DWTC-Extractor, generates four different types of information. The precision of the toolbox was evaluated on the round 1 of SemTab 2019 (Table 2.2) and compared to a modified version of the DWTC (which does not use HTML tags and thus supports more formats).
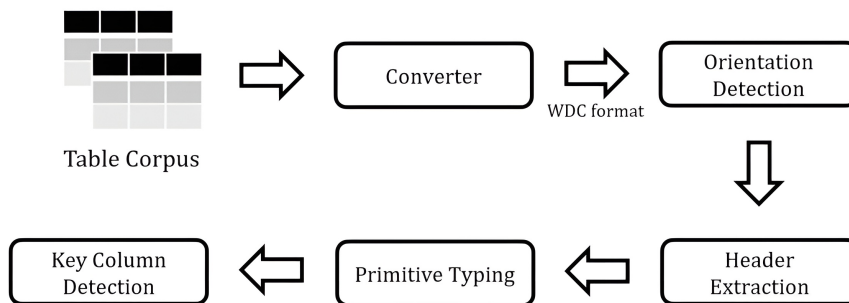
Figure 2.13: Illustration of the pre-processing pipeline

Table Orientation Detection. The initial algorithm proposed in the DWTC-Extractor is based

Table 2.2: Precision of pre-processing tasks

Task/Tool	DWTC	DAGOBAH
Orientation Detection	0.9	**0.957**
Header Extraction	Not evaluated	**1.0**
Key Column Detection	0.857	**0.986**

on the length of the strings and the assumption that the cells in the same column have a similar size. However, the robustness of the DWTC algorithm can be improved. Indeed, for example, two strings representing very different elements may have the same length (for example "Paris" and "$10cm^2$"). DAGOBAH introduces a new algorithm based on a primitive cell typing system with eleven types (string, floating numbers, date, etc.). Based on these types t_i, a homogeneity score is computed on each row and each column x (Equation 2.1). The mean of all rows and all columns is then compared, and depending on the ratio, the table is said "HORIZONTAL" or "VERTICAL".

$$Hom(x) = [\frac{1}{len(x)} \sum_{t_i \in x} (1 - (1 - 2 * \frac{count(t_i)}{len(x)})^2)]^2 \tag{2.1}$$

Header Extraction. The algorithm used in DAGOBAH is based on the primitive types defined above. The header extractor assumes that the header of a column contains mainly strings and, in most cases, does not share the type of column cells. The use of these two heuristics allows for identifying if a table contains or not a header with good accuracy (see Table 2.2). It should be noted that the DWTC framework also offers a header detection tool, but it contains several bugs that make its evaluation impossible.

Key Column Detection. In our implementation, the first step aims to identify a first low-level type for each cell among five given types (Object (mentions that are potentially lookup-able in a knowledge base), Number, Date, Unit (e.g. "12 km"), and Unknown (containing all the unclassified columns)) with regular expressions. Then we annotated each column with a type based on majority voting. The key column is an Object column containing a large number of unique values and is located on the left side of the table.

2.4.3 Result

The precision of the toolbox was evaluated on the round 1 of SemTab 2019 [63], and compared to a modified version of the DWTC (which does not use HTML tags and thus supports more formats). The result shows that we have optimized the result from the baseline system DWTC in Orientation Detection and Key Column Detection. This pre-processing toolbox was especially useful during the first round to automatically identify the information contained in the header as well as the column to be annotated in each table. In addition, when the goal is to enrich a knowledge graph with the information contained in tables, key column detection is a critical

element in determining the subject of the generated RDF triples. The availability of targets during the second round made the use of this pre-processing chain obsolete. However, this does not affect the usefulness of such tools in real-world applications.

2.5   Discussion

In this chapter, we first introduce a classification of tables. We propose that table could be seen as a combination of classes from three dimensions (structure, inter-relationship, and orientation). We have also proposed a deeper classification of a relational table using the subjects of the table as the criteria. It highlights some current limitations of understanding relational tables. For example, to the best of our knowledge, the hidden subject is not discussed in the literature. Later, we introduce the metadata that we could collect to better describe a given table.

We then emphasize our efforts in collecting Web tables with the student project HTW and CorpusWalker. We have built a collection, extraction and management pipeline that continuously enriches our data lake and provides fruitful information. In these two systems, we built Web scraping and table processing pipelines from the ground up, using basic components such as Scrapy, Kafka message queue, and ArangoDB document store. This pipeline can ingest data not only from the World Wide Web directly but also through the Common Crawl index and local files. We implemented advanced table parsing and specified a JSON data format for storing the tables, taking into account work that has previously been done in the field. While implementing our pipeline, we tried to keep the system modular and extensible, so that it could readily be reused by others, by adhering to modern software development practices.

Last but not least, we introduce our pre-processing module for extracting three basic features (table orientation, header position, and key column position) of a given relational table.

However many works are still not been handled yet. In the future, we first aim to propose a system that can automatically classify tables according to our definition illustrated in Section 2.1 (e.g., entity tables, relational tables, etc.). Also, we would like to drive a deeper evaluation of the tables collected with CorpusWalker, digging more into the topic distribution of wild tables. Also, proposing a suitable and complex table benchmark based on the extracted tables is in our planning. Finally, we aim to evaluate our pre-processing system with other datasets and study if it exists other useful features that we could extract from our pre-processing pipeline.

Chapter 3

Semantic Table Interpretation Tasks and Methods

The tabular data format is compact, readable and simple to process, but it does not self-explain the meaning of the information. However, it potentially contains rich semantic information that still needs to be interpreted. Hence, interpreting tabular data becomes a crucial task and it has attracted a lot of attention in recent years, with, in particular, the crystallization of research efforts around challenges such as the SemTab series [36, 63, 64]. The main idea to make tabular data intelligently processable by machines is to find correspondences between the elements composing the table with entities, concepts, or relations described in KG which can be of general purposes such as DBpedia [19] and Wikidata [128], or enterprise specific. This problem is known as STI or Semantic Table Annotation. KGs can be used to drive the semantic interpretation of tabular data while being themselves the artefacts that can be further enriched from the result of the interpretation process. In this latter case, tabular data becomes a means to either populate a nascent KG or improve the quality of an established one. Adding a semantic layer on top of tabular data, in order to make the latent meaning explicit and exploitable through a structured and shared format, is an invaluable step towards efficient and intelligent use of data.

Tabular data is challenging to interpret by machines because of the limited context available to resolve semantic ambiguities, the layout of tables that can be difficult to handle, and the incompleteness of KGs in general. Classical Natural Language Processing (NLP) tasks for unstructured text handle poorly such tables since they do not leverage the table structure and the underlying semantics [153]. For example, in the table depicted in Figure 2.5(b), the mention "Rohr" is ambiguous as it can refer to a surname (Q16882196), a manufacturer (Q2391081), or a municipality in Germany (Q583512). However, this ambiguity can be resolved when taking into account the table structure and, in particular, the fact that the "Manufacturer" column only contains companies.

This section aims to provide a general introduction to the STI domain, where we first introduce the notion of KG in Section 3.1. In Section 3.2, we define five sub-tasks that are relevant to STI,
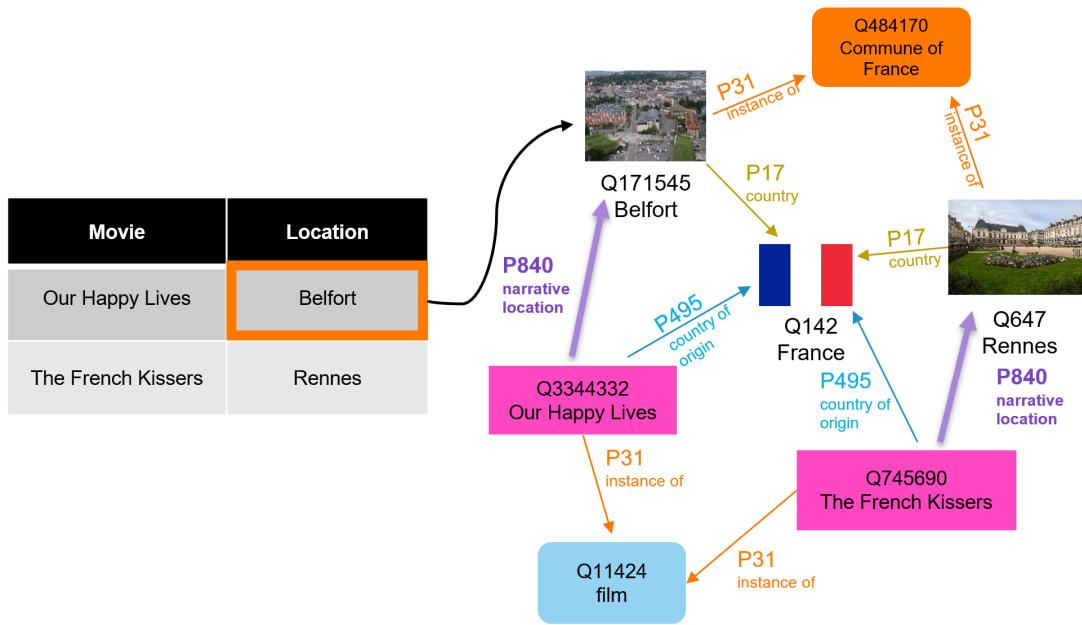
Figure 3.1: Illustration of an STI task with a given KG (Wikidata).

namely: cell-entity annotation, column-type annotation, columns-property annotation [63, 64], topic annotation, and row-to-instance annotation [104]. Section 3.3 proposes a macro-common pipeline that fulfills the tasks of STI from pre-processing of the input table to the final annotation results. In Section 3.4, we make an inventory of the gold-standard datasets commonly used to evaluate STI approaches and we analyse their strengths and weaknesses. In Section 3.5, we review the many approaches that have been proposed grouping them into three families (not mutually exclusive) respectively based on heuristics, feature engineering, and deep learning. We then describe the current performances of STI systems on selected datasets. Finally, we discuss the current approaches in Section 3.6.

3.1 Knowledge Graph

Knowledge graphs are often associated with linked data technologies and projects since they focus on interrelations between concepts and entities [42]. In essence, KGs are semantic networks that formally describe things or entities of the real world and their relationships [53]. Each entity is identified by a globally unique URI [14]. Atomic elements of KGs are triples ⟨subject, predicate, object⟩. For example, as the KG illustrated in Figure 3.1, the country of the city Belfort can be represented by the triple ⟨Belfort, country, France⟩ where the predicate country qualifies the relationship holding between the entity Belfort and the value France.

KGs can be categorized into domain-specific (or vertical) KGs, encyclopedic KGs or common-sense KGs depending on their content. A domain-specific KG focuses on describing a partic-

ular field of interest. Such a KG is expected to present advantages in terms of accuracy and in-depth domain knowledge coverage. It can effectively support knowledge reasoning and knowledge retrieval for specific domain applications [138]. To illustrate, in the "Crop, Pest, and Diseases" field, domain-specific KGs play a substantial role in agriculture [123], greenhouse environment [142], and economic benefit analysis [125]. The Bio2RDF project which focuses on Life Sciences [12] is an example of a domain-specific KG that is largely used.

An encyclopedic KG is generally large, spanning multiple domains, and is often openly and collaboratively edited. This openness is reflected by the Linked Open Data cloud [18] which includes the following largest encyclopedic KGs:

- **DBpedia** [19] is one of the main hubs of the Linked Open Data (LOD) cloud because of its numerous interlinks with other KGs. It was created by researchers from the University of Leipzig and the University of Mannheim in Germany by extracting multilingual structured data from Wikipedia (e.g. infoboxes). It is maintained up-to-date thanks to frequent extracts from Wikipedia. As of early 2016, DBpedia contained more than six million instances and 200 million facts. Moreover, the DBpedia project provides tools such as DBpedia Spotlight [77] that are convenient for mapping mentions contained in unstructured data with KG entities.

- **Wikidata** [128] is a project hosted by the Wikimedia Foundation which aims to fuel the infoboxes displayed on each Wikipedia page. Similarly to Wikipedia, it is collaboratively edited by thousands of volunteers. As of early 2021, Wikidata had facts about 90 million entities with labels expressed in more than 350 languages. Wikidata provides a separate page for each entity, has a unique digital identification mechanism, and a lineage system that allows to trace facts to their sources.

- **Freebase** [20] was developed by MetaWeb since 2007 until Google acquires it in 2010. Its content comes also from collaborative editing and structural data automatically imported from Wikipedia and other Websites. At the beginning of 2014, Freebase had 68 million entities and nearly one billion facts. Freebase ceased operations in May 2015, and most of its data was transferred to Wikidata.

- **YAGO** (Yet Another Great Ontology) [120] is a comprehensive knowledge base constructed by researchers from the Max Planck Institute (MPI). While the first versions of YAGO were made out of information extracted from the Wikipedia infoboxes and attached to a schema made of the WordNet synsets, the latest version of YAGO now contains entities extracted from Wikidata anchored to the Schema.org schema. In 2020, YAGO released its fourth version containing 67 million entities and 340 million facts.

KGs constitute essential assets to support the STI process. Indeed, understanding the content

of a table comes down to identifying the entities mentioned in the table cells and the relationships between them. Therefore, mapping table content to KG entities can help identify latent relationships, and thus understand the table semantics. The key to map tables and target KGs is to examine the overlap of information between them. The wider the overlap is, the less difficult it is to find the mappings.

It should be noted that each KG may present its own (dis)advantages to support the STI process. For example, Wikidata provides rich content and numerous aliases for each entity to cover a wide set of real-world synonyms. However, annotating a cell with such an encyclopedic KG based on a string-similarity mapping may lead to a significant number of candidates due to the presence of numerous homonyms. This would, in turn, make disambiguation more challenging. For example, over a hundred entities have labels or aliases that contain the word "France". The Wikidata data model is also complex as it provides qualifiers that may need to be specifically taken into account during the STI process. On the other hand, DBpedia provides a reduced number of types curated in the DBpedia Ontology, which makes the typing of table elements easier but potentially reduces the specificity of the annotations. Regarding vertical KGs, the lack of knowledge from other domains may decrease the robustness of the KG applications. Additionally, string matching may not be able to handle the specificities of sophisticated domain-specific relations, schema, or entities, increasing the interpretation complexity. For example, in the biology domain, genes and proteins often share the same labels. To guide the choice of the supporting KG, it is noteworthy that selecting KGs with the highest overlap with the dataset's content will maximize the system's performance. Besides, combining several KGs will maximize the coverage and granularity.

3.2 Semantic Table Interpretation Tasks

An annotation task can be defined by the table elements required to be annotated and by the type of candidates (individuals, concepts, or properties of the KG). As the example from Figure 3.1, one could map the cell Belfort with the entity Q171545 from the knowledge base with the help of the table context (the narrative location of the **movie Our Happy Lives**). Based on our study of literature, we propose to decompose STI into five main tasks: cell-entity annotation, column-type annotation, columns-property annotation [63], topic annotation, and row-to-instance [103] (illustrated in Figure 3.2).

- **Cell-Entity Annotation** (CEA) is also known as Entity Linking. It aims to annotate a cell with a KG entity. For example, in Figure 3.2, a CEA task needs to match the cell mention "Suisse" with the entity Q165141 if Wikidata is the target KG.

- **Column-Type Annotation** (CTA) aims to map a column with a KG entity type. The difficulty of the CTA task lies in selecting an adequate type granularity in a potentially
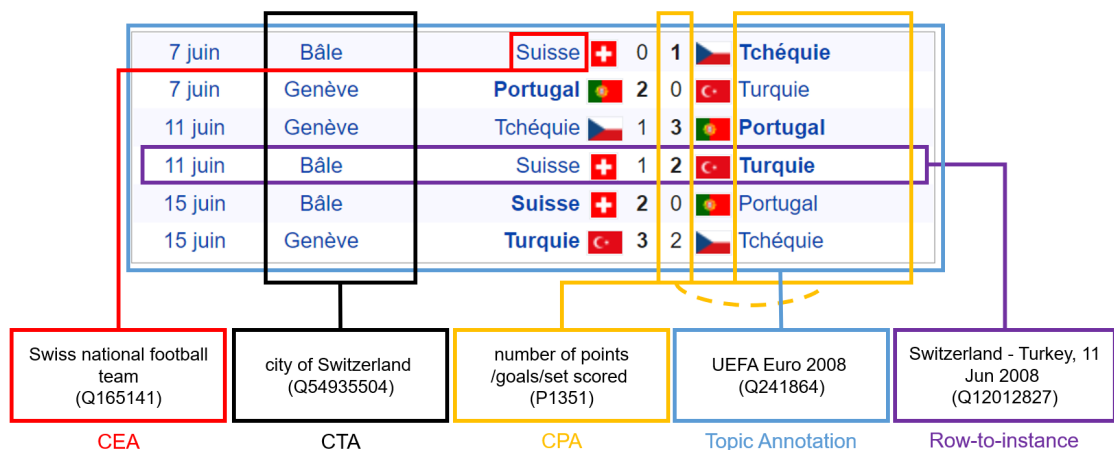
Figure 3.2: Illustration of five STI tasks for a table describing the UEFA Euro 2008 group A results [a].

[a]https://fr.wikipedia.org/wiki/Championnat_d%27Europe_de_football_2008#1er_tour_-_phase_de_groupes

complex type hierarchy structure. An entity may have multiple types and types represented in complex hierarchical trees or even cyclic graphs (e.g. Wikidata type topology). The type selected for a given column must be representative of the individuals it contains and carry a maximum of information. If the selected type is too broad (e.g. the second column of the table in Figure 3.2 is annotated as a "geographic entity" (Q27096213) in Wikidata rather than "city of Switzerland" (Q1545591)), the annotation will carry little information. Conversely, a type that is too specific may not be representative for all values in a column, leading to an accuracy degradation in downstream tasks. In Figure 3.2, the label "city of Switzerland" (Q14770218) would no longer be compatible with the second column if other groups and cities that hold UEFA Euro 2008 games such as Vienna (Q1741) are included in the table.

- **Columns-Property Annotation** (CPA) aims to annotate a column pair of a relational table with a property. For example, the relationship between the last column and the numeric column circled in orange in Figure 3.2 should correspond to the predicate "number of points/goals/set scored" (P1351) in Wikidata.

- **Topic annotation** aims to annotate the entire table with a concept or an entity from the target KG. Figure 3.2 illustrates that the entire table is about the entity "UEFA Euro 2008" (Q241864) in Wikidata.

- **Row-to-Instance** annotates an entire row of a relational table with a KG entity. In this task, each row is treated as an entity, which is considered the subject of the row. Row-to-instance differs from the CEA task as it may be able to discover more entities leveraging the context of the row, especially in the case where the subject of the row is hidden (e.g.

33

on hidden-subject tables). For example, in Figure 2.5, the fourth row is represented by ("Switzerland - Turkey, 11 Jun 2008" (Q12012827)) which can not be extracted by the CEA task.

3.3   Generic Pipeline

The study of STI approaches in the literature allows identifying a recurrent pipeline of modules used by the vast majority of systems. This section introduces a macroscopic view of a four-stage STI pipeline: pre-processing, candidate generation, table elements processing, and iterative disambiguation. It should be noted that the order of these modules may vary from one approach to another. In addition, some approaches iterate between the annotation tasks and disambiguation stages to improve the accuracy of annotations [104, 153].

Pre-Processing

The quality of the STI output is greatly influenced by the quality of the input data. A pre-analysis of the data is, therefore, necessary and is often the first step of an efficient STI system. For example, knowing the orientation of a relational table can help to identify a column and its header information to disambiguate the cells.

The goal of the pre-processing module is to quickly and concisely summarize and analyse an input table to ease the annotation process by converting it into an interoperable format. The pre-processing analysis can be decomposed into the following two tasks.

First, **format normalization** allows to transform the original data into a format acceptable to an STI system. Indeed, table sources are diverse, so does their representation in terms of formats (e.g. Comma-Separated Values (CSV), JSON, HTML), charsets being used (e.g. UTF-8, Unicode), languages (e.g. English, French), or content expressions (e.g. missing value). This task also aims to clean up invalid or erroneous data for better compatibility among different data sources, e.g. with syntactic corrections by fuzzy matching in [26, 89].

Second, **informational analysis** consists of extracting the potential information contained in the table as much as possible before the annotation. The information carried by a table includes the table types (e.g. relational, matrix) [68, 149], its orientation [24, 50], primitive types for its columns (resp. rows), its header positions, if any, and its key column position which carries the row's subject [24]. Such information helps the system to understand the scenario and to perform different operations in different situations. For example, the CTA task is only suitable for relational tables, and it is related to the table orientation: for horizontal tables (resp. vertical tables), it will assign a type to columns (resp. rows).

Candidate Generation

The annotation is usually selected within a candidate list that depends on the table elements and the corresponding task. For example, the first step of the CEA task is to generate a list of candidate entities while the first step of the CTA task is to generate a list of candidate types. Thus, STI systems automatically generate, manually add, or filter this candidate list in advance (e.g. an annotation system based on supervised learning should be trained with pre-set labels, those labels being the candidates of the annotation process).

String similarity-based lookup is a standard method to generate candidate entity sets in CEA and row-to-instance tasks. Specifically, the syntactic similarity is calculated between cell mentions and entity labels to select the most relevant entities as candidates. The choice of the matching algorithm depends on the application scenarios. For example, [59] uses Levenshtein-based distances whereas some public knowledge bases expose a public index allowing users to extract and generate candidates, e.g. DBpedia spotlight [77].

In other tasks like CTA (resp. CPA), candidates are the set of types (resp. predicates) available in the target KG. The generation of type candidates is sometimes equivalent to the retrieval of the CEA candidates' types [59, 79]. However, in some learning-based methods [98, 144], type candidates are manually selected for training.

Some mentions can correspond to a large number of entities, e.g. several thousands of entities whose labels contain the mention "France" can be retrieved from Wikidata. The number of candidates may affect the efficiency of the annotation system as calculating and evaluating all possible candidates may require a large amount of time. Hence, some STI systems prune the number of candidates [86, 122]. This filtering process can be applied to reduce the size of the final candidate set by tuning an acceptable entity-mention similarity threshold. However, one should be aware that an inadequate threshold risks to filter the correct entity out. Another way is to sort the importance of candidates by studying the attributes of the target ontology. For example, [26, 79] leverage the BM25 [106] weights from an Elasticsearch index, that can help to select the candidates with the highest usage rate.

Table Elements Processing

Processing different table elements is the core of an STI system. Each table element follows a particular rule according to the table type. Given an input table that is relational and horizontal, each row describes an entity and its attributes, while column elements share the same entity type. Given $\mathcal{T}_{ij}$, the target cell to be annotated from a horizontal table $\mathcal{T}$, we identify six table elements that can be leveraged to produce annotations.

$\mathcal{T}_{ij}$ indicates the target cell itself. Some studies use the string similarity as one of the compo-

nents of candidate confidence [59,79]. For example, in Figure 3.2, for searching CEA candidates for the cell "Suisse" in the first row, we should consider the entities containing the mention "Suisse" in their label. The correct annotation is Q165141 which has the English label "Swiss national football team" and the French label "équipe de Suisse de football".

$\mathcal{T}_{i*}$ indicates the row context of the target. Some studies leverage the matching degree between the attribute values of the target entity and the information provided in the table for the CEA task. Most of the annotation tasks consider a single-cell subject relational table scenario. Based on this assumption, it is reasonable to make this comparison for calculating the confidence of the candidate. For example, in Figure 3.2, knowing the date ("11 juin") and city ("Bâle") can help to annotate the third row with the right football game (Q12012827) based on its neighbouring nodes in the KG.

$\mathcal{T}_{*j}$ indicates the column context of the target. For a horizontal relational table, the information carried by cells in the same column of the table is somewhat similar (e.g. cells referring to the same concept or the same unit). For example, in Figure 3.2, the cells in the second column are cities. Having this information could help to choose the right candidate between the city of "Bâle" (Q78) and the family name "Bâle"(Q107983752).

$\mathcal{T}_{**}$ indicates intra-columns relationships from the target table. Intra-column relationships provide a global representation of a table. For example, in Figure 2.5(c), knowing the relation between the column "Album" and the column "Artist/s" could help to filter out people who did not publish any music album.

$\mathcal{T}_{0*}$ indicates the header of the target table. The table header often directly explains the contents of the column. Making full use of the information from the table header can help to find the column type or properties more efficiently. For example, in Figure 2.6(a), the headers "Year", "Country", and "Town" directly denote the concepts of the columns.

$\mathcal{T}_{out}$ indicates contextual elements encompassing the table. High-quality metadata can help the interpretation of the table. For example, the table's title can potentially determine the domain of the table content, or a hyperlink in a table cell can reveal the identity of the entity. In addition, the text surrounding the table is usually correlated with the content of the table. Leveraging this correlation is used by some STI approaches [38]. Another way to enrich the information used for the disambiguation of the target table relies on inter-table relationships [129].

From the elements mentioned above, the initial annotation step is based on row interpretation, column interpretation, entire table interpretation, or metadata interpretation. In row interpretation, each row in the table describes an entity's attributes. Column interpretation uses the entities in the same column of the relational table with high mutual similarity. This feature can help to constrain the range of candidates for a table cell. Entire table interpretation considers

all cells from the table as the context for the disambiguation and is usually performed using deep learning models such as [38, 115].

Iterative Disambiguation

When an STI system jointly undertakes multiple tasks among the five tasks defined in Section 3.2, one task can provide additional useful information for solving the other tasks. For example, when knowing the type annotation of a column (CTA), candidate entities for its inner cells (CEA) that do not belong to the CTA type are less likely to be correct candidates [26]. We call this process iterative disambiguation. This iterative technique is frequently used in heuristic-based approaches (Section 3.5.1) in which a pipeline including specific ordered tasks and being executed once or in a loop is explicitly defined in two ways: (1) predefinition of a pipeline, e.g. [26] performing sequentially CEA, CPA, CTA, CEA disambiguation with CPA or (2) repeat a set of tasks multiple times and stop when it converges to a stable result [104, 153].

3.4 Datasets and Benchmarks

Table 3.1: Gold standard datasets for evaluating STI approaches. Table type refers to the classification introduced in Section 2.1 where R=relational table, SR=single-cell relational table, V=vertical, H=horizontal.

Gold standard		Table Type	#Tables	Avg. #Rows	Avg. #Col	#Entities	#Class	#Relations	Origin	KG	Year
Limaye[a]	Manual	SR-H	437	37	2	10,930	747	90	Web	Wikipedia, YAGO	2010
	Wiki_link	SR-H	6,085	20	2	131,807	-	-	Web	Wikipedia	2010
T2D[b]	T2D[c]	SR-V	762	157	5	25,119	7,983	-	Web	DBpedia	2015
	T2Dv2[d]	SR-V	779	84	5	26,106	755	-	Web	DBpedia	2017
WDC[e]		All	233,000,000	12	4	-	-	-	Web	-	2015
TabEL[f]		R	1,652,771	11	5	3,000,000	-	-	Wikipedia	YAGO	2015
QuTE[g]		R	1,766,721	13	5	-	-	-	Wikipedia	-	2021
SemTab 2019[h]	R1	SR-H	64	142	5	8,418	120	116	Web	DBpedia	2019
	R2	SR-H	11,924	25	5	463,796	14,780	6,762	Web	DBpedia	2019
	R3	SR-H	2,161	71	5	406,827	14,780	7,575	Synthetic	DBpedia	2019
	R4	SR-H	713	63	4	107,352	1,732	2,747	Synthetic	DBpedia	2019
SemTab 2020	R1	SR-H	34,295	7	5	985,110	34,294	135,774	Synthetic	Wikidata	2020
	R2	SR-H	12,173	7	5	283,447	26,727	43,753	Synthetic	Wikidata	2020
	R3	SR-H	62,614	6	4	768,325	97,586	166,633	Synthetic	Wikidata	2020
	R4	SR-H	22,207	21	4	1,662,164	32,462	56,476	Synthetic	Wikidata	2020
	ToughTables	SR-H	180	1,080	5	663,830	539	-	Synthetic	Wikidata	2020
SemTab 2021	R1-DBP	SR-H	180	1,081	4	663,656	540	-	Synthetic	DBpedia	2021
	R1-WD	SR-H	180	1,081	4	667,244	540	-	Synthetic	Wikidata	2021
	R2-Bio	SR-H	110	2,449	6	1,381,325	657	547	Synthetic	Wikidata	2021
	R2-Hard	SR-H	1,750	17	3	47,440	2,191	3,836	Synthetic	Wikidata	2021
	R3-Hard	SR-H	7,207	9	2	58,949	7,207	10,695	Synthetic	Wikidata	2021
	R3-Git	SR-H	1,101	59	17	-	123 / 60	-	Github	DBpedia, Schema.org	2021
	R3-BiodivTab	SR-H	50	260	24	31,468	614	-	Open data	Wikidata	2021
SemTab 2022[i]	R1	SR-H	3,691	6	3	26,198	4,551	5,745	Synthetic	Wikidata	2022
	R2-Hard	SR-H	4,648	6	3	22,006	4,534	3,954	Synthetic	Wikidata	2022
	R2-2T (DBp)	SR-H	144	1,008	4	486,203	429	-	Synthetic	DBpedia	2022
	R2-2T (WD)	SR-H	144	1,180	4	586,118	443	-	Synthetic	Wikidata	2022
	R3-Biodiv	SR-H	45	24	276	31,942	526	-	Open data	Wikidata	2022
	R3-Git	SR-H	6,892	14	62	-	6,229/1,001	-/4,412	Web	DBpedia/Schema.org	2022

[a]https://zenodo.org/record/3087000#.YbY5Lp7MJPY

[b]The # class for the T2D dataset is the sum of the number of "table classes" and "column properties" in the original dataset.

[c]http://Webdatacommons.org/Webtables/goldstandard.html

[d]http://Webdatacommons.org/Webtables/goldstandardV2.html

[e]http://www.Webdatacommons.org/Webtables/

[f]http://Websail-fe.cs.northwestern.edu/TabEL/

[g]https://www.mpi-inf.mpg.de/research/quantity-search/quantity-table-extraction

[h]The SemTab series are indexed at https://www.cs.ox.ac.uk/isg/challenges/sem-tab/

[i]https://github.com/sem-tab-challenge/2022/tree/main/datasets

Several datasets have been proposed to evaluate STI approaches. Some datasets attempt to establish gold standards in which table components (cells, rows, columns, or cell pairs) are associated with KG components (entity, class, or property), while others collect high-quality tables to support STI training. In this section, we detail the most popular datasets (Table 3.1) that are used by STI approaches.

Limaye [72] is one of the earliest gold standards used in the community. Limaye aims to annotate Web tables using the YAGO KG. The dataset is divided into four subsets according to the data source, the labelling method, and application scenarios. Three subsets are manually labelled while the fourth one is automatically labelled. The automatically labelled subset contains annotation errors [80] which were corrected by [16]'s work in 2015. Later on, [41] updated the disambiguation links to the DBpedia KG.

T2D [105] is taken from the Web Data Commons project.[1] DBpedia is used as the target KG and extensive metadata such as the context of the table and whether the table has a header or not is provided. The addition of a small number of non-overlapping tables that do not have a mapping relationship with DBpedia makes this gold standard closer to real-world datasets. A second version of the gold standard adding negative examples has been published and named T2Dv2 [68]. T2D and its supplementary version T2Dv2 [68] have been largely used to evaluate approaches in [25, 56, 153], and together with Limaye, they became the de facto gold standard datasets to use for evaluating STI approaches. However, [43] pointed out that T2D has partial annotation errors and lacks fine-grained annotations since a large number of tables only point to the root class `owl:Thing`. [43] has proposed a revised version of the dataset named T2D*.

WDC [68] leverages the DWTC framework [40] to crawl tables from the Web and to distinguish between different types of Web tables according to the inner-relation dimension and the orientation dimension that have been defined in Section 2.1. The crawler has extracted a total of 10.24 billion tables from which 0.9% are relational tables, 1.4% entity tables, and 0.03% are matrix tables, amounting to 233 million tables available in the corpus. The rest are labelled as layout tables or other tables. WDC also provides a subset containing 90 million relational tables and a subset containing 50 million English relational tables.

TabEL [16] (also named WikiTables) has collected 1.6 million Wikipedia tables which contain the class attribute "wikitable" from the November 2019 XML English Wikipedia dump. During the extraction, the system collects the hyperlinks in table cells and metadata of the table, such as the table caption and the page title. Thus, the mappings between YAGO entities and table cells are easily derived. The types of tables in this dataset are unknown. **QuTE** [52] further enhanced the dataset by merging TabEL with 2.6 million tables from the TableL [60] dataset that were extracted from 1.5M Common Crawl web pages using the DWTC framework [40]. The TableL dataset covers mostly five major topics: finance, environment, health, politics,

---

[1] http://www.Webdatacommons.org/Webtables/

and sports. These datasets have generally been used to train machine learning based STI approaches [16].

Zhang et al. [153] proposed datasets[2] for evaluating entity linking in Web tables, as well as table header classification and relation annotation. The datasets contain 16,000+ annotated relational tables that can be used for many studies related to Web tables. In particular, it proposes the **IMDB** (movie) and **Musicbrainz** (music) datasets with cell entities and column headers annotated using Freebase topics.

The **SemTab** competition (Semantic Web Challenge on Tabular Data to Knowledge Graph Matching[3]) colocated with the International Semantic Web Conference from 2019 to 2022 provides the biggest datasets. This competition has attracted nearly 50 participant teams over the three years. The SemTab 2019 datasets [63] use DBpedia as the target KG, while SemTab 2020 [64] uses Wikidata and SemTab 2021 [36] uses both DBpedia and Wikidata in addition to Schema.org for the last round. The competition consists of four rounds in 2019 and 2020 and three rounds in 2021 and 2022. The data sources vary depending on the rounds. SemTab 2019 Round 1 is a small number of high-quality tables extracted from T2Dv2. Tables from SemTab 2019 Round 2 are extracted from Wikipedia. Except GitTables [55] and BiodivTab [4], the rounds from SemTab contain a large number of artificially generated tables annotated for a target KG using SPARQL queries with the introduction of some noise such as misspellings.

The synthetic tables datasets can be used to test the scalability of a system given their size and the large number of lookup candidates that can be returned. Nevertheless, these datasets are not extremely challenging for the semantic interpretation approaches as they contain synthetically generated noise. This explains the very high accuracy of the top-performing approaches (F1 score up to $0,99$ for some tasks [64]). There is also room for improvement in incorporating all real-world challenges in future editions of the challenge, for example, tables with cells containing multiple entities. To increase the difficulty, SemTab 2020 introduces during round 4 the so-called **Tough Tables** dataset [35]. Tough Tables is composed of tabular data simulating various difficulties: a large number of rows to evaluate the systems performance, non-Web tables and artificially added misspellings and ambiguities.

In SemTab 2021 and 2022, the organizers bring new challenges with, in particular, the BiodivTab [4] dataset which contains 50 manually annotated tables from real-world biological datasets. BiodivTab also contains artificially generated noises, abbreviations and complex data formats. A subset of the GitTables dataset [55] has also been used in the last round of SemTab 2021. This dataset is a collection of CSV tables from GitHub which are annotated with DBpedia and Schema.org.

---

[2]https://github.com/ziqizhang/data/tree/master/Webtable%20entity%20linking
[3]http://www.cs.ox.ac.uk/isg/challenges/sem-tab/

3.5   Semantic Table Interpretation Approaches

Table 3.2: STI systems are classified into three families. We only consider the annotation tasks declared by the authors and when they have related evaluations.

Approches		Annotation Tasks					Table Elements					KG	Data Source	Published Year
Class	Algorithm	CEA	CTA	CPA	R2I	TA	$\mathcal{T}_{i*}$	$\mathcal{T}_{*j}$	$\mathcal{T}_{0*}$	$\mathcal{T}_{**}$	$\mathcal{T}_{out}$			
Heuristic — Lookup based	Venetis et al. [127]		√	√				√				Custom	Custom Web Tables	2011
	Wang et al. [130]		√				√	√	√			Probase	Custom Wikipedia Tables	2012
	Deng et al. [37]		√					√				FreeBase, YAGO	Custom Wikipedia Tables	2013
	Sekhavat et al. [109]			√				√				DBpedia	Custom Web Tables	2014
	TabEL [16]	√					√	√				YAGO	Limaye	2015
	ADOG [94]	√	√	√			√	√				DBpedia	SemTab 2019	2019
	Tabularisi [122]	√	√	√				√				DBpedia	T2D, VizNet	2019
	C^2 [66]		√					√	√	√		DBpedia, Wikidata	Limaye, ISWC2017, SemTab 2019, T2D, Semantification, Custom Data	2020
	Magic [113]	√	√	√				√	√			DBpedia, Wikidata	SemTab 2021	2021
	Alobaid et al. [6]		√						√			DBpedia	SemTab 2021, T2D	2022
Heuristic — Iterative	Zwicklbauer et al. [156]		√					√				DBpedia	Custom Wikipedia Tables	2013
	T2K [104]			√	√	√	√	√				DBpedia	T2D	2015
	TableMiner+ [153]			√	√	√	√	√	√		√	Freebase	Limaye, IMDB, MusicBrainz	2017
	LOD4ALL [79]	√	√	√			√	√				DBpedia	SemTab 2019	2019
	CSV2KG [114]	√	√	√			√	√				DBpedia	SemTab 2019	2019
	MTab [86, 89, 90]	√	√	√			√	√	√			DBpedia, Wikidata	SemTab 2019-2021	2019
	LinkingPark [26]	√	√	√			√	√				Wikidata	SemTab 2020	2019
	DAGOBAH-SL [57, 58, 59]	√	√	√			√	√				DBpedia,Wikidata	SemTab 2020-2022	2019
	MantisTable [29, 30, 31]	√	√	√	√		√	√				DBpedia, Wikidata	SemTab 2019-2021	2019
	Kepler-aSI [8, 9]	√	√	√	√		√	√				DBpedia, Wikidata	SemTab 2021-2022	2020
	JenTab [1, 2, 3]	√	√	√			√	√				DBpedia, Wikidata	SemTab 2020-2022	2020
	KGCODE-Tab [71]	√	√	√			√	√				DBpedia, Wikidata	SemTab 2022	2022
Feature engineering based	Limaye et al. [72]	√	√	√			√	√	√			YAGO	Limaye	2010
	Mulwad et al. [80, 81]	√	√	√			√	√	√			Wikitology	Limaye	2010
	SemanticTyper [98]		√					√				DBpedia	Museum	2015
	DSL [78]		√					√				DBpedia	City, Museum, Weather, Custom Soccer	2016
	Neumaier et al. [84]		√					√				DBpedia	Government Data Portal	2016
	NUMER [65]		√					√	√			DBpedia	NumDB	2018
Deep learning based — KG modelling	Vasilis et al. [41]	√					√	√				Wikidata	Limaye, T2D, Wikipedia	2017
	Biswas et al. [17]				√						√	DBpedia	Custom Wikipedia inforbox	2018
	DAGOBAH-Em [24]	√	√				√	√	√			DBpedia, Wikidata	SemTab 2019	2019
Deep learning based — Table modelling	Sherlock [56]		√					√				DBpedia	T2D, VizNet	2019
	Sato [144]		√					√	√			DBpedia	VizNet	2019
	ColNet [25]		√				√	√				DBpedia	Limaye, T2Dv2	2019
	Guo et al. [49]		√					√			√	DBpedia	T2Dv2	2020
	Zhang et al. [150]	√		√			√	√				DBpedia	T2Dv2	2020
	TURL [38]	√	√	√			√	√	√	√	√	DBpedia	WikiGS, WikiTable, T2D	2020
	TCN [129]		√	√			√	√		√	√	-	Custom Web Tables, WikiTable [38]	2021
	DUDUO [115]		√	√			√	√		√		-	WikiTable, VizNet	2021
	Singh et al. [112]			√			√	√	√		√	DBpedia	T2Dv2	2021
	Zhou et al [154]		√					√		√		DBpedia	Custom Wikipedia Tables	2021

In this chapter, we review the notable approaches from the literature. Among the five tasks introduced in the previous section, the literature mostly focuses on CTA, CEA, and CPA. We propose to classify STI systems according to three representative paradigms of their intrinsic methodology: (1) heuristic methods (Section 3.5.1) which mostly rely on heuristic techniques such as entity matching, Term Frequency–Inverse Document Frequency (TF-IDF), majority voting, or simple probabilistic frameworks to predict a target; (2) feature-engineering based methods (Section 3.5.2) which require a feature engineering process to extract statistical and lexical features from the table that are then used to train Machine Learning models ; (3) Deep Learning based methods (Section 3.5.3) that leverage a large number of tables and neural networks to learn deep and contextualised representations of elements of the table, requiring little feature engineering. More details on this classification are shown in Table 3.2 including representative algorithms, target tasks, table elements used, reference KG, and year of publication[4].

Then, we discuss these methods from different perspectives: the pros/cons of heuristic-based methods versus Machine Learning based methods, the importance of table elements and the KG structure for improving the accuracy, and the trade-off between efficiency and accuracy (Section 3.6). Finally, we describe the current performances of STI systems on selected datasets in Section 3.5.4.

3.5.1 Heuristic-Based Approaches

The heuristic class gathers diverse approaches which are often considered as baseline STI approaches. The core of each system is algorithmically straightforward and does not require much effort in feature engineering or learning. Indeed, the STI tasks are carried out using heuristic techniques such as string similarity measures [79, 94, 130], majority voting [156], TF-IDF [94, 122] or probabilistic frameworks [86]. The context of the table, including the header, the title, and the neighbouring cells [59, 130] is also taken into account but not thoroughly. We further identify two subclasses of heuristic approaches: lookup based approaches (Section 3.5.1) and iterative approaches (Section 3.5.1).

---

[4]In Table 3.2, we use the following notions: "R2I" indicates the task "Row-to-instance"; 'TA' indicates the task "Topic annotation"; "$\mathcal{T}_{i*}$" indicates that, when labeling a cell, information from the same row is used; "$\mathcal{T}_{*j}$" indicates that the approach leverages information from the target column (CEA, CTA) or columns (CPA); "$\mathcal{T}_{0*}$" means that the approach has a special treatment on the headers of the table; "$\mathcal{T}_{**}$" indicates that the approach considers information from all tables elements, including inter-columns influence and training the model with the whole table; "$\mathcal{T}_{out}$" indicates that the approach not only uses the target table itself for annotation but also considers metadata, including other tables associated with the target table and the text near the original target table.

Lookup Based Approaches

Approaches from this paradigm work with an initial candidate entity set determined by a lookup service. After generating candidates through lookup, these methods score the candidates using different metrics on table elements (e.g. cells, type of columns, etc.).

Venetis et al. [127] introduce a model for extracting the column type and the relationships between the key column and other columns. To increase knowledge coverage and avoid issues related to KG incompleteness, the authors present an isA database to carry out the CTA and a relation database to carry out the CPA. The isA database is built by using concept extraction techniques on 100 million English documents that contain the pattern "$C[such\ as|including]e[and|,|.].$". They generate the relation database with the help of the TextRunner open extraction system [11]. The authors demonstrated that a hybrid model leveraging a Bayes rule and majority voting has the best performance. The Bayes rule measures the global relevance between cell values and column type labels in tables. The authors conclude that using a target knowledge base (YAGO) leads to higher precision. However, leveraging the isA database can significantly improve the coverage and allow to obtain more meaningful labels for complex or non-explicit table cells.

Wang et al. [130] focus on the table headers to better identify the concept associated with a given column. The approach uses a header detection module that leverages Probase [137] querying and rule-based filtering. In the absence of headers, a custom concept is employed with Probase queries by measuring the type occurrence of the column cells. The table interpretation is executed by studying the cells-header compatibility and entity-values compatibility. Through experiments on a search engine, the approach demonstrates that headers can help understand the columns of a table.

Deng et al. [37] focus on the production of top-k candidates for CTA. The authors first build a Directed Acyclic Graph mapping the entity labels from YAGO and Freebase within a type hierarchy tree. Then, they leverage a distributed system to make the process scalable and efficient without losing precision and accuracy for annotations. Specifically, a two-stage MapReduce system is built. (1) Multiple signatures for each cell mention and entity label are generated to support the cell-label fuzzy matching. For example, "Shar" is one of the signatures for the cell mention "Shark Night 3D". All candidate types according to the Directed Acyclic Graph are then aggregated with subordinative entities of each column. (2) The occurrence of each type is counted in order to select the top candidate type. To accelerate the computation, candidate types are aggregated into disjoint groups.

Sekhavat et al. [109] leverage NELL [118], a Web text corpus, and natural language patterns (PATTY [83]) to extract relations (CPA). The target KG is YAGO, in which only 23 relations are considered as possible annotation for a pair of columns. These are also relations extracted

by PATTY from Wikipedia pages. Each relation r is represented by a set of textual patterns $p_1, ..., p_k$ provided by PATTY. Note that a pattern p_i can be associated with more than one relation. Semantic mentions in a pair of columns are linked to KG entities (YAGO) with exact matching. Two columns are connected via a relation r if each pair of entities (e_1, e_2) in the same row belongs to this relation. To determine whether (e_1, r, e_2) is a valid triple, textual contexts related to both e_1, e_2 are extracted from NELL and are mapped to a list of patterns $p_1, ..., p_k$ in PATTY. The problem then comes down to computing the posterior probability of r given evidences $p_1, ..., p_k$: $Pr(r|p_1, ..., p_k)$. A Bayesian framework is used to compute this posterior.

TabEL [16] aims to provide an extensible framework. After a pre-processing, the approach first generates candidates for each cell using YAGO and ranks them according to their string similarity with the cell and their popularity. Every candidate participates in the calculation of the annotation. In the joint inference module, an undirected probabilistic graphical model is extracted to capture entity-context (elements from the same table) co-occurrence. These co-occurrence factors are updated with the connectivity between candidates, resulting in the final CEA annotations.

ADOG [94] considers scores combined with string similarities, frequencies of properties, and the normalized Elasticsearch score for each match from DBpedia for the CEA task. The system weights these scores with the IDF score of types. To be able to compute the Levenshtein distance and TF-IDF, ADOG uses ArangoDB[5] to load DBpedia and index its components. The frequency of classes and properties is then used to obtain the CTA and CPA results.

Tabularisi [122] adapts TF-IDF statistics to rank the CEA candidates in a given column. A candidate entity is represented by a binary feature vector in which each feature is an indicator (1 if present, else 0) of a property used to describe the entity (e.g. $instanceOf$). Different features have different expressiveness. They are thus weighted using TF-IDF. Specifically, the "Term Frequency" of a feature is the number of cells whose first candidate entity has that feature, and the "Document Frequency" is the total number of occurrences of that feature in all candidate entities in all cells. The score of a candidate entity is a weighted combination of its TF-IDF score, Levenshtein similarity and word similarity. The weights are either set or learned using a two-layer neural network. The CTA is performed by a top-down brute-force search in the KG class hierarchy tree. The system also sends the most frequent relation among columns with SPARQL queries of cell mention pairs.

C^2 [66] aims to handle the CTA task with the help of nine datasets. C^2 classifies columns into string entity columns, number columns, and mix-type columns (e.g. emails, dates). The system relies on decision trees, which leverage the pattern of the cell mentions or numerical interval for splitting the branches, to annotate numeric columns and mix-type columns. For

---

[5]https://www.arangodb.com/

entity columns, voting from the concepts extracted from DBpedia or Wikidata for each entity cell is used. C^2 also considers the type co-occurrence within a table to adjust the annotation.

Magic [113] adopts the approach of generating comparison matrices (called INK embeddings) to speed up the computational efficiency. INK embeddings are representations of the attributes and values of an entity or the table context of a cell mention. The complete comparison matrix is generated by fusing multiple candidates. The system outputs CEA annotations by measuring the compatibility between the INK embeddings of the KG and the table. The INK embeddings of entities from the same column are collected to carry out CPA and CTA. For the annotation, the system focuses on the key column: they do the lookup (via public endpoints) for each cell in the key column, then use its neighbourhood to find the candidates for the neighbouring cells in the same row (they do not perform the lookup on the whole table due to limitations of public API usage). Misspellings might however be a challenge for Magic and it cannot detect synonyms of attributes. However, INK embeddings improve computational efficiency and provide a way to implement column wised similarity.

Alobaid et al. [6] handles CTA with a strong focus on the trade-off between type coverage and type specificity after generating type candidates via entities-wised queries on cell mentions. The type coverage is built on a weighted type hierarchy index inside a column, and the type specificity is associated with a distance to the root. The authors test different balance settings between these two factors using the T2D and SemTab datasets.

Iterative Approaches

Iterative approaches are usually built on top of a lookup system, with an additional multi-task disambiguation step for re-ranking candidate entities. The iterative disambiguation techniques, as described in Section 3.3, play a significant role in the improvement of the model performance.

Zwicklbauer et al. [156] pioneered iterative majority vote strategies. The idea is based on the majority voting of annotation candidates of the cells for the CTA task. The system generates these cell candidates using a search-based disambiguation method [157]. Since majority voting plays an essential role in the system, this approach is sensitive to the number of table rows. The extreme case is that the annotation precision can be less than 0.1 with single row tables.

T2K [104] annotates Web tables by mapping their columns to DBpedia properties, and their rows to DBpedia entities, associating the whole table with a DBpedia class. A key column's position for each table is firstly detected by a preprocessing step. T2K transforms row-to-instance and table topic tasks into CEA and CTA on the key column. The initial entity mapping is derived from a lexicographical comparison between the labels used in the table and those

of the entities described in DBpedia with Jaccard, Levenshtein, and deviation similarities. An aggregation of these similarities is then used to choose the initial CTA annotation. The system adopts an iterative process between a CEA matching module and a CPA matching until the output is stable. The system achieves promising results on CEA and topic annotation.

TableMiner [152] and the following work **TableMiner+** [153], first use a lexical expression to extract a primitive type for each column. Similarly to T2K, TableMiner sees row-to-instance and table topic as CEA and CTA on the subject column. The system identifies a subject column for each table considering evidence collected from the original webpages. During the annotation phase, the authors iteratively label records corresponding to the subject column and their attribute columns using information from the HTML context of the tables to create a richer representation of cells and columns. TableMiner+ uses partial matching during the CTA annotation. The authors claim that partial matching is efficient and that eight rows are enough for supporting the annotation. A loop is used between CEA annotation and CTA annotation until the results remain stable. This system leverages the Freebase KG and was evaluated on the IMDB and MusicBrainz specific datasets, as well as on Limaye.

LOD4ALL [79] is initialized by building an RDF store database and a score DB database containing candidate types with scores reflecting the level of specificity generated by Okapi BM25 [106]. The system uses a similar approach as [156] for the candidate generation module, which uses a combination of Elasticsearch's score and the SimString's score to select the top 100 candidates. The CTA leverages Okapi BM25 type scores and the type coverage on the table. The CEA and CPA are calculated after CTA type filtering. The system is targeting DBpedia as ontology and has participated in the SemTab 2019 challenge. A similar pipeline is used by **CSV2KG** [114]. However, CSV2KG considers applying a threshold on the normalized entropy of the two highest type counts from the annotated candidates parent types list to decide the level of granularity of the CTA.

MTab [86] employs four different lookup services. This approach analyses signals from server's lookup ranking, header context analyses, SpaCy type prediction[6], Duckling type prediction[7] and value similarities. Each signal is transformed into a normalized probability score. The system aggregates selected probabilities with learnable weights according to the associated task. In addition, the authors also used EmbNum [88] to help to produce annotations on numerical columns. Column types and column pair relations are computed based on entity scores. Entities, types and relations are iteratively calculated two times to disambiguate CEA, CTA, and CPA annotations with inter-tasks relatedness. In a more recent work, MTab4Wikidata [89, 90] adapts fuzzy matching and "two cells search" to enhance the support of misspelling and ambiguities in table content. The system won the first prize in both SemTab 2019 and SemTab

---

[6]https://spacy.io/
[7]https://github.com/facebook/duckling

2020 challenges.

LinkingPark [26] leverages the Wikidata MediaWiki API to generate cell candidate entity lists. It also adapts a fine-grained Elasticsearch index to rank those candidates. This system firstly adopts a cascaded pipeline to generate candidate entities. Then the system disambiguates each cell through an iterative coarse-to-fine algorithm by considering the CPA annotation results. The system finally generates the CTA annotations from the disambiguated cell annotations. The authors also claim that Wikidata's type ontology is noisy which makes it difficult to assign types during CTA annotation.

DAGOBAH-SL [59] calculates a score for each cell entity. This score combines the Levenstein similarity and the context similarity between the entity's neighbouring nodes and the row context of the target cell. The authors collect triples containing cell candidates between the two columns and calculate the sum of the weights from the corresponding cell candidates for each relation. The output of CPA is the relationship with the highest sum. The system then leverages the CPA results to perform CEA disambiguation. As for CTA, in addition to the majority vote based on CEA, DAGOBAH-SL also leverages the distance to the root concept and the ranking of each Wikidata entity's class to select the most accurate type. In more recent works, DAGOBAH-SL [57, 58] enhances the system with CTA disambiguation and look-up generation. The entity context made with multiple-hop neighbouring entities is also taken into account in calculating the scores. The system won the first prize in accuracy in the SemTab 2021 and SemTab 2022 challenges.

MantisTable [30, 32] pipeline starts with classifying each column into three types: Named Entity column, Literal column, and Subject column. The candidate generation of this approach is based on SPARQL queries which extract all candidates containing the cell mentions. Then, the system handles the CEA using row-wise compatibility analysis and CPA using majority voting. For the CTA task, the authors list all candidate types in addition to their number of occurrences in the table (row coverage). After filtering with a threshold, the rest of the type candidates are transformed into a graph according to the ontology hierarchy. Type scores are then updated with the distance to the root. In the end, the highest score represents the most accurate and specific annotation. The recent version of **MantisTable SE** [29] optimizes the system by updating the scoring function, accessing the LamAPI[8] API (instead of using a SPARQL endpoint) and adding a final disambiguation step. MantisTable also supports row-to-instance by applying CEA on a subject column detected in preprocessing. The enhanced version of MantisTable, named **s-elBat** [31], proposes a revision step after generating the CEA, CTA, and CPA annotations. This system recheck the confidence of each annotation, only some mentions are considered for an additional annotation attempt to optimise computational efficiency. A similar pipeline has been adopted by Kepler-aSI [8,9] where they leverage Wikidata

---

[8]https://bitbucket.org/disco-unimib/lamapi

or DBpedia end-point.

JenTab [1, 2, 3] starts from analysing the row context and column context of a table. This system wraps each computational unit into independent modules so that they can be recalled easily and repeatedly. Those modules include row information processing, disambiguation using CTA output, etc. The system leverages different module combinations for supporting CEA, CTA and CPA tasks. The evaluation shows that JenTab has an excellent performance on synthetic datasets. The authors also investigate the implications of considering multi-hop links in type hierarchy relations. The result shows that considering two hops has a small probability of improvement, while multi hops lead to a significant decrease of the accuracy.

KGCODE-Tab [71] first distincts table columns into subject columns and attribute columns. Later, for each cell value, the system uses Bing[9] to search the cell mentions and obtains the result page in HTML pages for checking and correcting the spelling of each word. The cell candidates are extracted from MediaWiki Action API[10] and DBpedia Lookup[11]. Later, the authors design different functions for computing the Cell-Entity Annotation (CEA), Column-Type Annotation (CTA), and Columns-Property Annotation (CPA) for subject columns and attribute columns. The system won the first prize in accuracy in the SemTab 2022 challenge.

3.5.2 Feature Engineering Based Approaches

This family of methods extracts statistical and lexical features (such as distribution of numerical values, occurrence of cell mentions, textual similarity, etc.) from the table rows and columns and uses them with machine learning models. Typical algorithms used for STI include SVM [81], Random Forest [78] and K-Nearest Neighbor [84] for example. A labelled dataset is required for the training. The amount and the quality of training data, and consequently the quality of input features, have a significant impact on the model performance, as discussed in [78]. In addition, we observe that ML methods target the CTA task more than other tasks, as columns can provide more statistical features than other annotation targets.

Limaye et al. [72] introduce one of the first works on STI. The approach computes the TF-IDF cosine similarity between a cell mention and an entity label and the compatibility between the cell type and the column type to execute the CEA task. CTA task depends on TF-IDF cosine similarity between column header and each entity's type label. The CPA annotation depends on the compatibility between the relation and column pairs. All these features are weighted through a machine learning framework.

Mulwad et al. [81] leverage majority voting on the type of cells candidates for the CTA annota-

---

[9]https://www.bing.com/search
[10]https://www.wikidata.org/w/api.php
[11]https://lookup.dbpedia.org/

tion. The PageRank algorithm weights each cell's type from the ontology during the CTA. For CEA annotation, the system collects entity features from entity PageRank, string similarities, entity index score and entity page length to generate a vector representation of each entity. An SVM classifier is then built upon these features to make predictions. This system supports both DBpedia and Wikitology.

SemanticTyper [98] processes each column of the table independently for CTA. The system first distinguishes between columns of numbers and columns of strings by voting on the types of cells in each column given a predefined threshold. Strings columns are trained upon cosine similarities on TF-IDF, considering a column as a document. To annotate numerical columns, the authors use a variety of distribution representation methods. In both cases, they adapt the training data that consist of a set of semantic labels associated with samples of data values. The prediction aims to find the most similar candidate by comparing the distance between the query column and each sample set in the training data corresponding to a distinct semantic label. The chosen distance metric depends on the column type. The training dataset was extracted from vertical domain datasets such as museums and cities, and the authors associate columns from these datasets with DBpedia classes. The main limitation of this work resides in not taking into account relationships between columns.

DSL [78] build their approach for CTA on datasets from four different domains: city, weather, museum, and soccer. Labels are partially manually added to these datasets. DSL leverages features including string similarity and number distribution between chosen labelled datasets and the rest of the data during the prediction. The difference with SemanticTyper is that the distribution is also available for string columns in this approach. The system learns the weights between these features through two supervised learning algorithms: logistic regression and random forest. The evaluation shows that logistic regression achieves better results. The authors claim that the incorrect predictions come from the top decomposition point of the decision tree.

Neumaier et al. [84] focus on CTA labelling for numerical columns. Their work is not limited to predicting a unique label but rather expands the scope of labelling to its surrounding information. For example, instead of labelling "height", this system will label it as "the height of an athlete playing basketball in the NBA". To do so, the authors constructed a background KG based on DBpedia. This background knowledge base is extracted as a hierarchical structure divided into multiple multi-level groups to provide context. Each node in the hierarchy represents a type or a predicate and provides statistical information (maximum, minimum, or distribution) of the relative number set as features. The authors use these features and KNN to make predictions. The authors also explore the system's performance at different hierarchy levels in the background KG built on DBpedia and Open Data. They pointed out that DBpedia has still limitations in terms of coverage and freshness compared with other open

datasets. For example, the Austrian Open Data Portal has tables generated by weather stations every 15 minutes. However, DBpedia typically has numeric values only for "current" or "latest". Another limitation of this work is that the size of numerical columns and the popularity of numerical KG properties may influence the accuracy. Hence, **NUMER** [65] proposes to link subject cells with KG entities first, and then only extract the linked properties for enabling the column-wise number distribution study.

3.5.3 Deep Learning Based Approaches

Deep Learning has achieved many successes in various domains thanks to the availability of huge amounts of data and powerful computing resources. It has attracted more and more attention from the STI community over the past few years. We identify two main directions for the application of deep learning in the STI domain: KG modelling (Section 3.5.3) and table modelling (Section 3.5.3).

KG Representation Learning

This direction focuses on the entity level in which models learn embedding representations for entities of a table cell instead of the cell itself. Specifically, KG embedding techniques (e.g. TransE [21], TransH [134]) are used to encode the entities and their relationships into a vector space. STI models rely on the intuition that the entities in the same column should exhibit semantic similarities. Hence, they should be close to each other in the embedding space w.r.t. cosine similarity distance [41] or Euclidean distance [24].

Vasilis et al. [41] provide different methods. One of the proposed systems assumes that the correct CEA candidates in a column should be semantically close. From this assumption, a weighted correlation subgraph in which each node represents a CEA candidate is built. The edges are weighted by the cosine similarity between two related nodes. The best candidates are the ones whose accumulated weights over all incoming and outcoming edges are the highest. In addition, a hybrid system that combines the correlation subgraph method and an ontology matching system is also introduced and achieves a significant improvement in the end.

Biswas et al. [17] focus on topic annotation for Wikipedia infoboxes by leveraging metadata from the Wikipedia page. The annotation ignores the infobox content since the infobox information is often incomplete, incorrect and missing. Wikipedia page section headers and abstract are extracted as the source of information and are featured by Word2Vec embeddings for each word. Note that named entities present in the abstract are also transformed into RDF2Vec [101] vectors with DBpedia pre-trained embeddings. The global representation vector (called document embeddings) of an infobox is the concatenation of Word2vec and

RDF2vec vectors. Two classifiers (Random Forest and CNN) are trained on top of the document embeddings on 150,000 tables with 30 preset types. The evaluation shows that CNN outperforms the Random Forest classifier.

DAGOBAH-Embedding [24] hypothesizes that all entities in the same column of the table should be close to each other in the embedding vector space. Consequently, the correct candidates are assumed to belong to a few clusters. The K-means clustering is performed using TransE's pre-trained embedding to cluster the candidate entities. The good clusters with high coverage are retained by a weighted voting strategy. Both CEA and CTA are selected from the chosen clusters. Experimental results prove that this approach has successfully improved the accuracy of the CTA task. However, the system is also misled by incorrect candidates in the selected clusters during the CEA task.

Table Representation Learning

This direction deals directly with the textual content of the table as well as intra-table and inter-table interactions. The contextualized representation of basic elements of the table (i.e. cell, column) is learned by using deep neural networks [25, 56] or language models like BERT [38, 115, 154].

Sherlock [56] learns to perform the CTA task using 1,588 features extracted from a single column of a given relational table. The features are divided into four categories: character-wise statistics (e.g. frequency of the character "c"), column statistics (e.g. mean, std of numerical values), word embedding, and paragraph embedding. Except for column statistics features, other features are compressed into a fixed-size embedding using a subnetwork. A two-fully connected layer network is trained on both the embedding features and column statistics features to predict a column type annotation among 75 types inherited from the T2Dv2 dataset. The evaluation shows good results on various column types, including Dates and Industry. However, it is less sensitive to the purely numerical values or values appearing in multiple classes. Facing the potential missing information in single-column annotation, **Sato** [144] extends Sherlock by considering the whole table context. The table topic embeddings with LDA features modelled by an additional subnetwork and the column-pairs-wise dependency modelled by a CRF layer are also studied.

ColNet [25] predicts the column type (CTA) using only intra-column contextual information. Specifically, all the cell mentions of a column are split and concatenated into a single word sequence. This word sequence is converted into an embedding vector using word embeddings models like word2vec. This embedding is later input into a CNN model. It is worth noting that ColNet predicts the type of each column independently, and thus ignores the inter-column contextual information. To generate a training dataset, ColNet makes use of a KG to collect

candidate classes given the cells of the column. For each candidate class c, N different sets of h entities belonging to c in KG are retrieved and the associated word embeddings are stacked into a matrix $\mathcal{M}$. This forms N training samples $\{\mathcal{M}, c\}$ for class c. To alleviate the computational complexity, the candidate classes that appear rarely are not modelled and h rows of the column are randomly selected to predict a type annotation. The type prediction of ColNet is then refined by the matching entities of inner cells through majority voting. Similar work includes **Guo et al.** [49] where the authors also introduce the use of BiGRU-Attention rather than CNN and support multi-column annotation using linear-CRF (linear-chain conditional random field) on the entire network table and an undirected graph model that directly models the conditional probability.

Zhang et al. [150] address the table-to-KG matching including CEA and CPA tasks. Additionally, in the pipeline, with the help of a table-to-KG matching step, their tool performs a novel entity discovery task. A classification model is based on syntactic similarity (e.g. edit distance, Jaccard distance) and semantic similarity (deep semantic matching method DRMM [47]) between a table mention and corresponding candidate entities to determine whether the mention is linkable. If yes, at most one entity is predicted for this mention. Another classifier is then built on the cell annotations (CEA), exploiting column-wised features such as naive features (e.g. length of header), label similarity between header and properties of cell entities, value similarity between literal values (e.g. numerical, time) in the table and literal values of cell entities for the column property matching (CPA).

TURL [38] pioneers the application of pre-trained language models such as BERT in the STI domain. It provides a universal contextualized representation for each table element (i.e. caption, header, content cells) which can be fine-tuned and applied in various downstream tasks such as CEA, CTA, CPA, or table augmentation. The table augmentation task mainly involves enriching the semantics of the table by extending it with new columns (attributes). The model employs a Transformer-based encoder [126] to capture the information from table elements. To this goal, the input table is first serialized into a sequence of caption tokens, title tokens, header tokens, and row-by-row cells. A cell consists of its content (mention) and a candidate entity representing it in a KG. The sequence of tokens is then converted into embeddings using word embeddings for textual tokens and KG embeddings for entity tokens. To reduce the redundancy in the fully-connected attention learning and better draw the inter-column and intra-column, inter-row and intra-row, column-row interaction, the conventional attention layer is masked by a so-called visibility matrix which allows only a portion of table elements to participate in the modelling of a specific element. For example, cells in the same row or the same column can interact with each other. Apart from the BERT's Masked Language Model objective, TURL introduces an additional Masked Entity Recovery objective to reinforce the learning of factual knowledge embedded in the table and represented by KG entities. The model is trained on 570K relational Wikipedia tables.

Singh et al. [112] introduces a method based on BERT for relation extraction from tables (CPA). Only two-column tables are studied in this work. The table is tokenized and transformed into linearized rows and linearized column headers that are passed through a pre-trained BERT encoder to obtain two vector representations for table content and table header. A fully-connected layer takes as input these two vector representations and predicts scores over all candidate relations in the two-column table. The model is trained on synthetic tables generated automatically from a KG. However, synthetic tables lack metadata such as column headers and captions. According to the authors, such metadata is important for the relation extraction task. Hence, they propose a novel method which generates synthetic tables associated with metadata (context of table contents, meaningful column headers).

TCN [129] not only exploits intra-table contextual information but also inter-table contextual information for the two tasks CTA and CPA. According to the authors, the global context of a table can be complemented by discovering its implicit connections with other semantically related tables. Such inter-table connections can come from overlapping cell contents, consistent schemas or similar table topics between two tables. The embedding representations of a specific table cell and the table topic are jointly learned leveraging intra-table contexts (i.e. other cells in the same column or the same row, the table topic) as well as inter-table contexts (i.e. the cells sharing the same value, the columns with a similar header and the topic from other related tables). All of these contexts are fused into the embedding through the attention mechanism. As in DODUO model [115], the CTA and CPA tasks are trained in a supervised manner on two specific objective functions dedicated to column type prediction and column pair relation prediction, respectively. In addition, in the case where a large annotated training dataset is not available, TCN switches to transfer learning in which the ultimate cell embedding is fined-tuned with a BERT-like unsupervised pre-training. The evaluation shows that the inter-table contextual information contributes positively to the model's performance. However, the utilisation of inter-table context remains challenging since it requires prior knowledge about tables' schemas which are generally diverse.

DODUO [115] learns to jointly annotate the column type and column pair relation through multi-task learning. Similarly to other Transformer-based models, the key idea of DODUO is to incorporate table contexts (intra-column and inter-column contexts) into the prediction of a single column type or a single column pair relation using a table-wise attention mechanism. The model serializes the input table column by column into a sequence of tokens in which each token represents either a column header or a column cell. A special [CLS] token is appended at the beginning of each column to distinguish two different columns. This token is also considered as the embedding representation of the column itself. During the inference phase, two different output layers perform two different tasks: one takes a single column representation (i.e. the hidden vector of the [CLS] token) as input and produces a semantic type for this column accordingly, one takes a pair of column representations and predicts a

relation between them.

Zhou et al. [154] focus on the column type detection (CTA) task. This work leverages the Star-Transformer model [48] to learn a vector representation for each column taking the inter-column interactions into account. The input embedding of a column is initialized as a concatenation of semantic features which are word embeddings averaged over cell contents and statistical features which are adopted from Sato's [144] Sherlock model. In the context of limited training tabular data and weak order-dependence of table columns, the Star-Transformer is preferable to the Transformer as it reduces the computational complexity by replacing the fully-connected attention with a sparser one.

3.5.4 Evaluation

Traditionally, STI systems are evaluated using the information retrieval metrics: accuracy, recall, and F1 scores [24, 38, 63, 78, 153]. Among the various tasks, CTA is a special one since a given type and its parents can be all correct when determining the category of an entity or of a group of entities. For example, Paris can be equally typed as a capital or more generally as a city, which are not in conflict with each other. Considering only one of these types, such as capital, as the only correct answer would make it difficult to assess systems that can only predict the parent class. The SemTab 2019 challenge has defined the AH (Equation (3.1)) and AP (Equation (3.2)) scores for this purpose, where PerfectA indicates that the predictions made by an STI system exactly match the type declared in the ground truth and OkayA refers to annotations corresponding to one of the parent types. This evaluation method is also adopted by [25]. The SemTab Challenge evaluates systems using the AIcrowd evaluator[12] and STILTool [33].

$$AH = \frac{\#\text{PerfectA} + 0.5 \times \#\text{OkayA} - \#\text{WrongA}}{\#\text{TargetColumns}} \tag{3.1}$$

$$AP = \frac{\#\text{PerfectA}}{\#\text{AnnotatedColumns}} \tag{3.2}$$

SemTab 2020 and 2021 further enhance the CTA evaluation metrics by introducing the correctness score $cscore$ (Equation (3.3)) for correctly positioning the annotated type in the hierarchy tree where $d(\alpha)$ indicates the distance between the annotation α and the ground truth.

---

[12]https://github.com/sem-tab-challenge/aicrowd-evaluator

$$cscore(\alpha) = \begin{cases} 0.8^{d(\alpha)}, & \text{if } \alpha \text{ is an ancestor of the GT,} \\ 0.7^{d(\alpha)}, & \text{if } \alpha \text{ is a descendant of the GT,} \\ 0, & \text{otherwise;} \end{cases} \qquad (3.3)$$

Given the $cscore$, approximated Precision (AP), Recall (AR), and F1-score (AF1) for the CTA evaluation are then defined as follows:

$$AP = \frac{\sum cscore(\alpha)}{\#\text{Annotations}}, \ AR = \frac{\sum cscore(\alpha)}{\#\text{Targets}} \qquad (3.4)$$

$$AF1 = \frac{2 \times AP \times AR}{AP + AR} \qquad (3.5)$$

Table 3.3: Top-3 systems for each dataset and their corresponding F1 score unless otherwise stated in the footnote.

Dataset		CEA / Row-to-instance			CTA[a] / Topic annotation			CPA		
Limaye[b]		TabEL[c]	TabEAno (Mtab) [87]	T2K ++	T2K ++	Guo et al	MantisTable	Mulwad et al.	T2K ++	TableMiner+
		0.894	0.88	0.87	0.88	0.852	0.84	0.89	0.80	0.76
T2D		TabEAno	Zhang et al.	Kruit et al.	ColNet[d]	Alobaid et al. [6]	MantisTable	T2K ++	Singh et al.	MantisTable
		0.91	0.90	0.89	0.976	0.96	0.95	0.91	0.71	0.51
SemTab 2019	R2	MTab	CSV2KG	Tabularisi	MTab	CSV2KG	Tabularisi	CSV2KG	IDLab	Tabularisi
		0.911	0.883	0.826	1.414	1.376	1.099	0.881	0.877	0.790
	R3	MTab	CSV2KG	ADOG	MTab	CSV2KG	Tabularisi	MTab	CSV2KG	Tabularisi
		0.970	0.962	0.912	1.956	1.864	1.702	0.844	0.841	0.827
	R4	MTab	MantisTable	CSV2KG	MTab	CSV2KG	Tabularisi	MTab	CSV2KG	Tabularisi
		0.983	0.973	0.907	2.012	1.846	1.716	0.832	0.830	0.823
SemTab 2020	R1	MTab	LinkingPark	MantisTable	JenTab	LinkingPark	MTab	MTab	LinkingPark	JenTab
		0.987	0.987	0.982	0.962	0.926	0.885	0.971	0.967	0.963
	R2	MTab	DAGOBAH	LinkingPark	LinkingPark	MTab	DAGOBAH	MTab	LinkingPark	DAGOBAH
		0.995	0.993	0.993	0.984	0.984	0.983	0.997	0.993	0.992
	R3	MTab	LinkingPark	DAGOBAH	LinkingPark	MTab	DAGOBAH	MTab	DAGOBAH	bbw [110]
		0.991	0.986	0.985	0.978	0.976	0.974	0.995	0.993	0.989
	R4	MTab	LinkingPark	DAGOBAH	MTab	bbw	DAGOBAH	MTab	bbw	DAGOBAH
		0.993	0.984	0.985	0.981	0.98	0.972	0.997	0.995	0.995
	2T	MTab	bbw	DAGOBAH	DAGOBAH	MTab	LinkingPark	-	-	-
		0.907	0.863	0.830	0.743	0.728	0.686	-	-	-
SemTab 2021	R1 (DBpedia)	DAGOBAH	GBMTab	JenTab	JenTab	DAGOBAH	Magic	-	-	-
		0.945	0.692	0.607	0.46	0.422	0.159	-	-	-
	R1 (WikiData)	DAGOBAH	MTab	AMALGAM [7]	DAGOBAH	MTab	JenTab	-	-	-
		0.923	0.907	0.658	0.832	0.728	0.697	-	-	-
	R2-Hard	MTab	DAGOBAH	MantisTable	MTab	DAGOBAH	MantisTable	MTab	JenTab	DAGOBAH
		0.985	0.975	0.968	0.977	0.976	0.955	0.997	0.996	0.996
	R2-Bio	DAGOBAH	MTab	MantisTable	MTab	Magic	DAGOBAH	MTab	DAGOBAH	JenTab
		0.970	0.964	0.93	0.956	0.916	0.916	0.947	0.899	0.899
	R3-Biodiv	JenTab	MTab	DAGOBAH	KEPLER-aSI [8]	DAGOBAH	MTab	-	-	-
		0.602	0.522	0.496	0.593	0.391	0.123	-	-	-
	R3-Hard	DAGOBAH	MTab	MantisTable	DAGOBAH	MTab	MantisTable	MTab	JenTab	DAGOBAH
		0.974	0.968	0.959	0.99	0.984	0.965	0.993	0.992	0.991
	R3-Git (DBp)	-	-	-	DAGOBAH	KEPLER-aSI	MantisTable	-	-	-
		-	-	-	0.07	0.041	0.037	-	-	-
	R3-Git (Sch)	-	-	-	MantisTable	DAGOBAH	-	-	-	-
		-	-	-	0.205	0.183	-	-	-	-
SemTab 2022	R1	DAGOBAH	s-elBat	Kepler-aSI	DAGOBAH	JenTab	s-elBat	DAGOBAH	s-elBat	JenTab
		0.975	0.957	0.944	0.954	0.945	0.945	0.984	0.983	0.975
	R2-Hard (WD)	DAGOBAH	KGCODE-Tab	s-elBat	KGCODE-Tab	DAGOBAH	Kepler-aSI	DAGOBAH	s-elBat	KGCODE-Tab
		0.904	0.856	0.925	0.968	0.960	0.881	0.931	0.931	0.916
	R2-2T (WD)	DAGOBAH	s-elBat	KGCODE-Tab	KGCODE-Tab	DAGOBAH	Kepler-aSI	-	-	-
		0.945	0.937	0.905	0.543	0.409	0.369	-	-	-
	R2-2T (DBp)	DAGOBAH	KGCODE-Tab	s-elBat	KGCODE-Tab	s-elBat	TSOTSA	-	-	-
		0.926	0.827	0.789	0.480	0.373	0.342	-	-	-
	R3-BioDiv	KGCODE-Tab	TSOTSA	DAGOBAH	KGCODE-Tab	TSOTSA	Kepler-aSI	-	-	-
		0.91	0.76	0.736	0.87	0.79	0.73	-	-	-
	R3-Git(Sch)	-	-	-	KGCODE-Tab	s-elBat	TSOTSA	-	-	-
		-	-	-	0.66	0.65	0.48	-	-	-

[a]For SemTab 2019, we consider the AH score, while for SemTab 2020 and 2021, we consider the AF1 score

[b]We consider only one of the Limaye subsets named Manual

[c]TabEL only reports about the accuracy of the system and not the F1 score

[d]ColNet was evaluated on T2Dv2. Thus, we consider only the result from 237 PK columns, which is almost the 233 tables from T2D

Table 3.3 gives the performance of the top three systems with the highest F1, AP, or AF1 scores for the CEA, CTA and CPA tasks on the datasets commonly used by the community. We observe a considerable diversity in the evaluation process across studies. This diversity is reflected in: i) the original version of the Limaye dataset had some errors that were corrected by the TabEL team. However, these corrections are not used by STI systems; ii) the TabEL and C^2 teams only report on accuracy scores, thus lacking an evaluation of the recall to compare with other systems; iii) ColNet has further enhanced the T2D dataset while other approaches do not consider these enhancements; iv) [66] provides aggregated results, and it is hard to get the performance details per dataset, and v) the performance of a system with the same dataset in different articles can largely vary. For example, the difference in accuracy between ColNet's T2D dataset between [25] and [66] is about 60%, probably because [66] has not used all of ColNet's settings. This diversity makes it difficult to compare the performance of different STI systems and stresses the importance of challenges such as SemTab in the STI community.

We adopted the following strategy to produce Table 3.3: i) we only consider datasets that have been used for evaluation more than three times. In addition, the datasets used solely for training purposes (e.g. TabEL [16] and Viznet [54]) have not been considered; ii) if the performance of a system on the same dataset has been reported multiple times, we only consider the accuracy from the original paper; iii) we prioritize comparing F1 scores except for CTAs in SemTab 2019, which take into account the AP score. If F1 scores are not provided, we use precision for the comparison; iv) for the SemTab challenge, we used the final results presented in the papers published by the participants rather than the results achieved during the time frame of the competition.

Based on our classification of STI approaches along three paradigms, we observe that heuristic systems appear in the top three systems for all datasets and all tasks. In particular, none of the feature engineering systems or deep learning systems reached the top three in the entity matching tasks (CEA and Row-to-instance). We believe that one of the main reason is that unlike CTA or CPA, which can extract features from columns or column pairs (all column cells can provide features such as entity embeddings, string length, or distribution), features that can be used to annotate individual cells or single rows are relatively rare. As a result, it limits the performance of such systems. Furthermore, the performance of a learning-based classifier is related to the number of candidates used. Annotating an entity means that a classifier should be trained to serve millions of candidates, which makes the task more difficult. From this point of view, rare feature engineering systems or deep learning systems position themselves for the tasks of CEA and row-to-instance. Ideal STI systems are therefore likely to be hybrid systems combining the best of heuristic-based and deep learning based methods.

We further group the selected datasets according to their provenance. These datasets are either synthetic tables automatically generated (e.g. SemTab datasets) or tables collected

from the Web (e.g. Limaye and T2D). We observe that heuristic-based systems that follow an iterative approach such as MTab, DAGOBAH, LinkingPark and JenTab, achieve very strong performances on large synthetic datasets such as SemTab. These systems generally use the entire target KG in order to get a very good coverage. Leveraging inter-tasks process for iterative disambiguation further optimizes the performance of the system. As these datasets are synthetically generated from a KG, string matching based approaches have also more advantages since this matching step is generally very reliable and will not face challenges with KG incompleteness issues. Systems that rely on statistical learning such as ColNet or Guo et al. [49], on the other hand, perform well on smaller real-world datasets, like Limaye and T2D. It may be because smaller datasets provide a limited set of candidate entities/types/relationships. Short candidate lists make training more accessible, more efficient and more accurate. In addition, heuristic approaches are highly based on the closed world assumption, where KG incompleteness is always a big issue. Deep learning and feature engineering are more robust on this point since they are not highly dependent on the completeness of the KG. That may be one of the reasons explaining that learning-based methods such as ColNet or Guo et al. [49] have been able to become one of the top three systems for real-world datasets like Limaye and T2D.

3.6   Discussion

We have grouped the many STI approaches proposed so far into three families or paradigms. In this section, we further analyse these methods alongside different dimensions: the pros and cons of matching methods versus learning methods, the trend towards deep learning methods, the importance of table elements, the trade-off between efficiency and accuracy, and the influence of the target KG structure for improving the accuracy (but at the risk of adding noise).

Matching vs Learning

We observe that STI approaches rely on matching (a KG entity with a cell mention) and learning. Matching is key in heuristic-based approaches while feature engineering and deep learning based methods rely on representation learning of the input table. These can also be combined: the matching strategy is employed by learning-based models as a post-processing step where the annotations learned from neural networks are refined using mention-matching entities (DAGOBAH-Embedding [24] and ColNet [25] are two examples).

From our observation, matching highly relies on the compatibility between the target table and the target KG. Consequently, it may be challenged by incompleteness in the table, knowledge shifting of KG and incompatibility between the table and the KG. Matching methods are less

robust to noise than the learning methods. On the other hand, learning methods need large training datasets which are not always easy to collect or generate. Some learning approaches limit the number of target candidates to alleviate the lack of training data. [38,56,78,98,115,129] predict the CTA within a predefined set of around one hundred types. ColNet [25] deals with the data shortfall using data augmentation. The model is trained on data generated automatically from a KG. However, it takes hours to do a single annotation. Another challenge for learning approaches is the size of target tables. To perform the CTA, [56,78,98] rely on the statistical features computed from the table (e.g. distribution of number, length of string for each cell, etc.). These features are not statistically stable if the number of samples (i.e. table cells) is low. Finally, we also discovered that learning approaches do not consider thoroughly the hierarchy of types possibly used in a KG impacting the type specificity returned by the CTA task [78,98,115].

Rise of Deep Learning

After 2017, deep learning techniques emerged in the STI field and have attracted research focus. Compared to feature engineering approaches, complex neuronal networks allow the system to process tabular features more efficiently as the feature engineering step is sometimes difficult and time-consuming to maintain. For example, Sherlock [56] is based on 1,588 column-wised features. To mitigate this issue, an end-to-end learning framework is preferable and is more and more employed, for example, KG modelling with KG embedding techniques (e.g. Vasilis et al. [41]) and table modelling with BERT-like models (e.g. TCN [129]). However, we observe that table modelling approaches using language models always target class annotation (CTA) or relation annotation (CPA) tasks. The entity annotation task (CEA) still lacks dedicated work and has a lot of room for improvement. At the time of conducting this survey, TURL may be the only one that handles the CEA task. Moreover, many systems [115,129,154] try to simplify the table representation to a collection of unordered lists for columns or rows, ignoring their index and other structural information. TURL [38] proposes a visibility matrix, as an attention matrix, to describe the connections between table elements (e.g. cells in the same columns, cells in the same rows, etc.). We argue that this design is only applicable to relational tables whereas the model is trained on a dataset containing all table types. This limitation still requires more effort to cover more complex scenarios.

Coverage of Table Elements

Annotation models use different table elements that are analysed in depth in Section 3.3. We observe that more and more table elements are considered in recent approaches. This phenomenon is characteristic of learning-based models. [56,78,84,98] primarily concentrate on extracting features from a single column. With the rise of the deep learning and attention

mechanism, [38, 115, 129, 144] start to pay more attention to other complex table elements. A typical example is TURL [38], in which the authors consider all of the six table elements discussed in Section 3.3. However, we can not compare the superiority of an approach only by the number of elements it uses. The search for the right number of elements taken into account to increase the accuracy without being subject to noise remains an open challenge.

Effectiveness vs Efficiency

Annotation systems usually deal with a trade-off between effectiveness and efficiency. TableMiner+ [153] introduces partial matching in which the CTA calculation relies on only eight table rows in order to improve the performance. This strategy indeed makes the systems faster but degrades the accuracy. For example, considering the annotation of a column containing ["Joe Biden", "Donald Trump", "Barack Obama", "Abe Shinzo"], applying partial matching on the first three column cells will output "American presidents" as the type of this column, while the correct answer is more likely to be "politicians" since "Abe Shinzo" is not an American president but a Japanese prime minister. Systems whose annotation pipeline includes a candidate generation step will heavily depend on the entity lookup service used. However, public lookup endpoints impose several limitations on their usage and it may take more time to obtain a candidate set with a desirable coverage of the target table. Furthermore, some systems (e.g. [86]) are not suitable for real-time applications due to the heavy computational requirements of their intrinsic algorithm. In addition, in scenarios where there is not enough data for training, learning-based models take advantage of transfer learning. While it helps to save time and resources, the system accuracy may be degraded if the fine-tuning is not carefully performed.

Public KGs vs Custom KGs

Many approaches to annotate tables rely on encyclopedic KGs such as Wikidata and DBpedia. Those KGs provide rich and high-quality information helping the annotation become more effective. However, more information also leads to more ambiguity, and KGs are usually incomplete. Knowledge base shifting is also a challenge for approaches based on public KGs. We observe that some approaches [38, 56, 78, 98, 115, 129] only treat the target KG as a dictionary of concepts but not a knowledge network, which means the relationships and hierarchy of concepts have not been used. The extreme case is that some attention-based models [115] directly abandon KGs and use only concept names. Knowing how to properly inject a KG structure into a statistical model is an open challenge. Some works build their custom KG to increase the coverage. For example, [127] built an isA database using Web documents which contains three times more types than Freebase.

Chapter 4

Heuristic-Based Semantic Table Interpretation

The problem of semantic interpretation of tabular data is a growing topic in the scientific community spanning multiple research communities. This is also a primary concern in industry since there is a growing desire to extract dormant knowledge from the internal repositories to feed enterprise knowledge graphs. For interpreting the content of a table, the critical ingredient is to capture some form of context. The context can be immediate neighboring cells and the table structure (rows, columns, headers) or some external sources such as encyclopedic or common sense knowledge. The heuristics-based approach tends to rely solely on the context that one can find in the table. The heuristic class gathers diverse approaches, often considered baseline STI approaches. The core of each system is algorithmically straightforward and does not require much effort in feature engineering or learning, as introduced in Section 3.5.1.

In this chapter, we introduce our efforts in developing a heuristic-based STI system, named DAGOBAH SL. It is a mature system for performing STI that has participated in the yearly SemTab challenge series since 2020. DAGOBAH SL produces fine-grained annotations for cells (CEA), columns (CTA), and relationships between columns (CPA) given different reference knowledge graphs. The system is available via an API for developers as well as via a user-friendly Web interface that offers functionalities for visualizing annotations, and enriching tabular data from the knowledge graph (e.g. adding columns and filling in missing values).

We first present our lookup mechanism for generating the candidate cell entities in Section 4.1. In Section 4.2, we describe the first prototype of our heuristic annotation system, named DAGOBAH SL, and evaluate it with the participation of SemTab 2020. We further introduce our optimizations of DAGOBAH SL with the Multi-hop Relations scoring and Soft Context Scoring mechanism in Section 4.3. Then, we introduce the industrialisation of DAGOBAH-SL in Section 4.4 and conclude in Section 4.5.

4.1 DAGOBAH Lookup

It is not reasonable to run the entire annotation algorithm for all entities from the target knowledge graph. Supposing a system that annotates a cell with an entity from Wikidata, if this system calculates the compatibility score with the table context for all the millions of entities in Wikidata, the execution time will be very high. Hence, the annotation process should be launched within a limited range of candidates. Selecting the candidates for annotation is referred to as the "Candidate Selection" steps in Section 3.3.

In this section, we aim to introduce the cell entity lookup methods that have been adopted by DAGOBAH systems. The DAGOBAH entity lookup service, named DAGOBAH Lookup, provides a fuzzy search engine given a KG alias list for each entity. DAGOBAH Lookup aims to generate a limited number of candidate entities taking a cell as input. We first introduce the lookup method in Section 4.1.1, the deployment of the lookup system in Section 4.1.2, and then report the performance in Section 4.1.3.

4.1.1 Lookup Mechanism

Given a target cell[1] e_m contained in a table, we aim at retrieving a set of relevant candidates $\mathscr{E}_c$ from a KG, *i.e.* entities whose labels or aliases are similar to the text of the cell. We assume that for each target cell e_m, there is not more than one matching entity, w.r.t, we did not consider the case of multi-value tables. We employ two strategies to evaluate the similarity between a cell and the candidate entities: an exact match using a regex similarity, and a threshold-based match using the Levenstein ratio. We take the maximum value between all comparisons made across entity labels and aliases.

Regex similarity Candidate labels or aliases should include all the words of e_m in any order. This is particularly useful to match the full name of a person since the first and last names could appear in any order, *e.g.* "Elon Musk" versus "Musk Elon". A full edit ratio on these two multiple tokens strings would yield a relatively low score of 0.44.

Levenshtein ratio This ratio is computed between the candidate labels and aliases, and the text of the cell. We empirically fix the threshold to 0.65 to ensure that a cell has at least one candidate and we then retain the top 50 candidates for each cell.

---

[1] In the context of the SemTab challenge series, the target cells are provided. In general, DAGOBAH has a module that enables inferring a primitive type for any arbitrary cell so that lookups are only triggered on cells that have entities as values.

4.1.2   Lookup Platform

We aim to handle the cases where the table corpora are significantly large with thousands of tables and cells to annotate, such as tables used in the context of the SemTab series. The pre-processing (introduced in Section 2.4) step helps to identify table columns eligible for entity lookup based on their primitive types[2]. Given a cell e_m in such a column, the entity lookup step retrieves a set of relevant candidate entities $\mathscr{E}_c(e_m)$ from a target KG. The lookup service of DAGOBAH currently supports two KGs for which indexes have been built: Wikidata and DBpedia.

Wikidata entities. The lookup service collects items and properties with their labels and aliases in all languages. To increase lookup coverage, aliases of each entity are enriched with the values of 11 additional properties such as P2561 (name), P1705 (native label), or P742 (pseudonym).

DBpedia entities. The lookup service collects English resources with their labels in all languages. To increase lookup coverage, labels are enriched with the values of 25 alias properties such as abbreviation, birthName, or originalTitle. In addition, labels and aliases of all redirected entities are also included.

The public Wikidata/DBpedia API is not an ideal lookup service due to restrictions on the number of concurrent connections, result set sizes, and query time. To avoid these limitations, we built our own lookup service using the Wikidata Toolkit[3], DBpedia dump[4], and a Spark-based big data platform. Specifically, from the initial raw Wikidata dump, we use the Wikidata Toolkit to filter out the unnecessary triples or entities such as Form, Lexeme, MediaInfo, Sense and Statement, so that we only retain entities that are items identified by QID and property identified by PID, associated with their labels and aliases in all languages. For DBpedia, we download the dump and index all triples from the documents. We then store the filtered dump in a Hadoop Distributed File System (HDFS) and perform the entity lookup via the Spark framework. We continuously update the dump for concluding the new entities and facts. Table 4.1 presents the Wikidata dump used in this service during the experiment for the evaluation with the participation of SemTab 2020.

4.1.3   Evaluation

In order to evaluate the performance of the system, we participated to the SemTab 2020 challenge. Table 4.2 provides the statistics of this dataset.

---

[2]Since target cells are given in the SemTab challenge, this feature is not used in our experiments.

[3]https://github.com/Wikidata/Wikidata-Toolkit

[4]https://downloads.dbpedia.org/2016-10/core/

Wikidata version	# triples	# entities	# predicates
July 2020	490M	86M	7,800
Circa 2017	100M	50M	8,200

Table 4.1: Overview of the Wikidata KG used in the challenge. A 2017 version was used in Round 1. The July 2020 version was used in Round 2, 3 and 4.

	Round 1	Round 2	Round 3	Round 4
# Tables	34K	12K	63K	22K
# Cells to annotate	985K	283K	768K	1.66M
# Unique cells to annotate	264K	138K	379K	531K
Average cell length	20	21	20	21

Table 4.2: Overview of the SemTab 2020 table corpus in each round. Several cells may contain the same text. Such cells are only counted once in the line "# Unique cells to annotate". The length of cells counts the number of characters.

We provide in Table 4.3 the complete lookup time for the Round 1, 2, and 3 (line "Current Time"). To construct our lookup service, we considered a pure Python Levenshtein module which has very poor performance (generally, 50 to 100 times slower than the more efficient Cython Levenshtein library[5]). As a result, this lookup service is not optimized, taking about 268h ($\approx$ 11 days) for the lookups of Round 3. In a future system, the Cython Levenshtein library can be used to significantly reduce the execution time. We roughly estimate the potential lookup time of such a future system (line "Ideal Time") by inducing the time of Cython Levenshtein usage from the pure Python Levenshtein's time.

	Round 1	Round 2	Round 3	Round 4
Current time	44	64	268	102
Ideal time	44	36	96	-

Table 4.3: Spark Lookup Time (in hours) for Round 1, 2, and 3 using 150 machines and for Round 4 using 5 clusters, each of 100 machines. "Current time" is measured using the pure Python Levenshtein module. "Ideal time" is the expected lookup time in a future system using the Cython Levenshtein library.

4.2 DAGOBAH-SL Scoring System

In this section, we introduce the DAGOBAH SL (Semantic Lookup) system for STI. DAGOBAH SL provides an end-to-end process that annotates relational tables with constituents of a KG such as Wikidata. Similar to DAGOBAH Lookup, we assume that for each target cell e_m, there is

---

[5]https://pypi.org/project/python-Levenshtein/

not more than one matching entity. Its workflow consists of the four sequential steps depicted in Figure 4.1. Given a relational table as input, the pre-processing determines table metadata and annotation targets (introduced in Section 2.4). The entity lookup module then collects candidates from the KG for each target cell of the table (introduced in Section 4.1). In this section, we explain the following steps: i) the pre-scoring module evaluates each candidate to determine a confidence score (Section 4.2.1). ii) next, we introduce the Columns-Property Annotation (CPA) annotation (Section 4.2.2). iii) Cell-Entity Annotation (CEA) is performed in order to compute the final entity scores taking CPA into account (Section 4.2.3). iv) Then, Column-Type Annotation (CTA) (Section 4.2.4) is carried out with majority vote based on CEA results. Finally, we report the result with the evaluation of the SemTab 2020 challenge (Section 4.2.5).
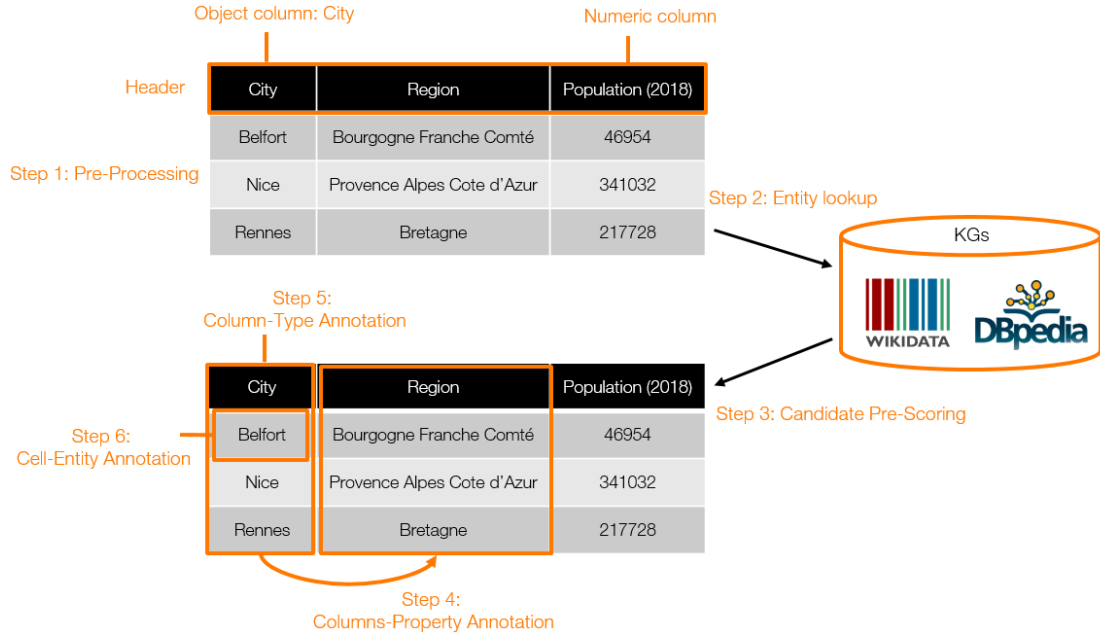
Figure 4.1: Overview of the DAGOBAH annotation workflow.

4.2.1 Pre-scoring Mechanism

The pre-scoring step aims to assign a preliminary confidence score to each candidate $e_c \in \mathscr{E}_c$ generated after the lookup step. In the first version of DAGOBAH-SL, named DAGOBAH-SL 2020, the confidence score function (Equation (4.1)) leverages the semantic contextual relationships and literal similarity between the candidate e_c and the cell e_m.

$$Sc(e_c, e_m) = Sc_{context}(\mathscr{N}_{table}(e_m), \mathscr{N}_{graph}(e_c)) * Sc_{sim}(e_m, e_c)^x \qquad (4.1)$$

67

$Sc(e_m, e_c)$ is the product of a context factor $Sc_{context}(\mathcal{N}_{table}(e_m), \mathcal{N}_{graph}(e_c))$ and a literal factor $Sc_{sim}(e_m, e_c)$. $Sc_{sim}(e_m, e_c)$ provides the highest Levenshtein ratio between the cell and the label set of the candidate. This label set is composed of the labels and aliases of the entity (*e.g.* "France", "La France" for Q142). $x \in \mathbb{N}^+$ allows defining the importance of the textual similarity acting as an amplification factor. We empirically observed that 5 was an appropriate amplification factor for x for this challenge.

The context score is calculated by Equation (4.2):

$$Sc_{context}(\mathcal{N}_{table}(e_m), \mathcal{N}_{graph}(e_c)) = \begin{cases} \overline{\mathcal{N}_s} \text{ if } \overline{\mathcal{N}_s} \geq 0.1 \\ 0.0001 \text{ otherwise} \end{cases} \tag{4.2}$$

$\mathcal{N}_{table}(e_m)$ is the set of neighboring cells in the same row as e_m. $\mathcal{N}_{graph}(e_c)$ is the set of neighboring elements of e_c in the KG[6]. Neighboring literals are directly added to $\mathcal{N}_{graph}(e_c)$ whereas, for neighboring entities, their labels and aliases are added. $\mathcal{N}_s$ is a set which contains the best neighborhood matching score ns_i for each neighboring cell $n_i \in \mathcal{N}_{table}(e_m)$ and all neighboring literals or nodes in $\mathcal{N}_{graph}(e_c)$. For each neighboring cell n_i, the neighborhood matching score $ns_i \in \mathcal{N}_s$ is generated as follow:

- If n_i is a string, then ns_i is the highest Levenshtein similarity value.

- If n_i is a number, then ns_i is given by Equation (4.3):

$$ns_i = \max_{n_g \in \mathcal{N}_{graph}(e_c)} 1 - \frac{|n_i - n_g|}{|n_i| + |n_g|} \tag{4.3}$$

- If n_i is a date-time value, and if a matching exists between the cell and a neighboring element, then we set ns_i to 1, otherwise $ns_i = 0$.

- If the previous steps gives a ns_i value lower than 0.85, then the system resets ns_i to 0.0001.

For example, the confidence score $Sc(e_m, e_c)$ of the candidate e_c "Q1574185" with the cell e_m "University College Cork" is equal to 1 since "Q1574185" has this exact label and all neighboring cells share information with neighboring elements of "Q1574185".

4.2.2 Columns-Property Annotation (CPA)

The CPA task involves finding a semantic relation between a pair of ordered columns {head, tail}. In other words, we try to figure out the most suitable relation among the ones connecting a

---

[6]Neighboring elements are object (resp. subject) of triples whose subject (resp. object) is e_c.

candidate entity in $\mathscr{E}_{head}$ for the head column to a candidate entity in $\mathscr{E}_{tail}$ for the tail column. We employ a simple majority voting strategy which relies on the occurrence and accumulated confidence score of the relation r that we define as:

- Occurrence(r) = #(e_{head}, e_{tail}),

- AccumulatedConfidenceScore(r) = $\sum Sc(e_{head}) * Sc(e_{tail})$

such that $e_{head} \in \mathscr{E}_{head}$, $e_{tail} \in \mathscr{E}_{tail}$, and $\langle e_{head}, r, e_{tail} \rangle \in KG$.

In contrast to the relation's head which is always an entity, its tail can be an entity, a textual value, a numerical value, or a date-time value. Literal values may be noisy (*e.g.* "370.1069999997" may represent the integer value "369", "1845-01-01" may correspond to "1845/01/01" or "1845" in the KG). Due to the different natures of each tail type, we consider different matching metrics to verify whether a triple $\langle e_{head}, r, e_{tail} \rangle$ exists in the KG:

- For entity IDs, score(id_1, id_2) = 1 if the two ids are exactly the same, otherwise 0.

- For numerical values, score$(num_1, num_2) = 1 - \dfrac{|num_1 - num_2|}{|num_1| + |num_2|}$

- For string values, score$(text_1, text_2)$ = Levenshtein$(text_1, text_2)$

- For date-time values, we compare many variants of date-time values (*e.g.* from the initial value "2020", we could have "2020-01-01" or "2020/01/01"). Two date-time values are matched, *i.e.*, score$(date_1, date_2)$ = 1 if one is a variant of the other.

From the criteria above, the relation with the highest occurrence is considered the target relation. If there are several relations having this highest occurrence, we select the one with the highest score. For example, consider the table provided in Fig. 4.1. The two relations P131 ("*located in the administrative territorial entity*") and P159 ("*headquarters location*") can relate the first (Col 0) and the third (Col 2) columns. Specifically, Q1574185-"University College Cork" (resp. Q245247-"King's College London") is located (P131) in Q36647-"Cork" (resp. Q202059-"London Borough of Lambert"). Cork is also the headquarter (P159) of University College Cork. Therefore, the occurrence count of P131 is 2, and 1 for P159. We conclude that P131 is the CPA for the column pair {Col0, Col2}.

4.2.3 Cell-Entity Annotation (CEA)

The CEA task aims to annotate the table cells with entities from the KG. Taking the CPA into account. In DAGOBAH-SL 2020, for each candidate entity in a given row, we update its score computed in the pre-scoring step (Section 4.2.1) by adding a constant score 1 (resp. 0) if the

CPA relation can (resp. cannot) connect this candidate to the counterpart in the remaining column. The output entity is the one with the highest score.

For example, for the cell "Cork" in Fig. 4.1, two of the candidate entities are Q36647 ("Cork city in Munster, Ireland") and Q162475 ("Cork County, Ireland"). They both have the same score output from the pre-scoring step (Section 4.2.1). Given we have determined the relation between "Col 0" and "Col 2" to be P131 and that the candidate Q36647 is linked to the cell "University College Cork" via this relation, we increase the score of Q36647 by 1. Q36647 then becomes the candidate with the highest score, and is chosen as the CEA output for the "Cork" cell.

In the edition 2021, we also consider the result from CTA (Section 4.2.4). Indeed, the pre-scoring of a candidate entity e_c only considers its local information, *i.e.*, the row it belongs to. Column type output by CTA and column pair relations output by CPA allow to consider table's global information. Hence, the final score $Sc(e_c, e_m)$ of a candidate entity e_c is computed as follows:

$$Sc(e_c, e_m) = \frac{(PSc(e_c, e_m) + \alpha \times score_{CTA} + \beta \times \overline{score_{CPA}})}{1 + \alpha + \beta} \tag{4.4}$$

If e_c belongs to the type output by CTA for its column, then $score_{CTA}$ is equal to the score of this type, otherwise it is set to 0. In $\overline{score_{CPA}}$, we average the scores of the relations output by CPA for column pairs involving the column of e_c. For each relation, if e_c belongs to its domain or its range (depending on the relation orientation), then we consider the score of this relation, else it is set to 0. To strengthen (resp. weaken) a frequent (resp. unusual) CTA/CPA in the update of $Sc(e_c, e_m)$, a coefficient α (resp. β) is employed and defined as $\frac{occurrence(CTA)}{2}$ (resp. $\frac{occurrence(CPA)}{2}$). Note that the occurrence of CTA/CPA is divided by 2 to prioritize the pre-scoring $PSc(e_c, e_m)$.

4.2.4 Column-Type Annotation (CTA)

After getting the output of CEA, a majority voting strategy is adapted to identify the most precise type for target columns. This process is illustrated in Fig. 4.2 for the first column ("Col 0") of the example table depicted in Fig. 4.1.

The CTA annotation begins with a type enrichment step. Let $\mathcal{T}_j$ be the set of candidate types for the cell j from a target column. $\mathcal{T}_j$ is represented by a hierarchical tree with 3 levels of types. We consider the types related by the P31 predicate ("*instance of*") to the predicted CEA entity as level 0. Their parent types linked through the P279 predicate ("*subclass of*") are assigned to level 1 and their ancestors to level 2. The system gives priority to the direct types. The rank of a candidate type is another useful factor for the CTA decision. According to
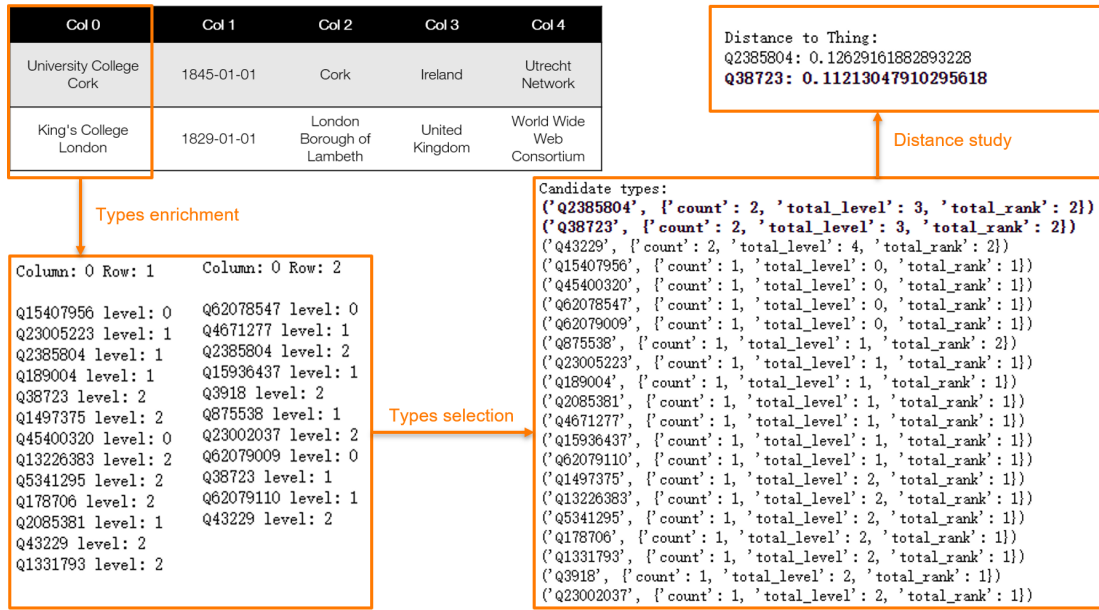
Figure 4.2: CTA annotation structure

Wikidata's mechanism for annotating multiple values for a statement, a type for an entity may have three ranks[7]. We represent those ranks as priority numbers: PREFERRED-2, NORMAL-1 and DEPRECATED-0.

The second step consists in a preliminary selection of types. We compute the frequency, accumulated level, and accumulated rank for all candidate types of a target column, *i.e.* all types appearing in at least one $\mathcal{T}_j$ of the column. We then select the types with the highest occurrence, the lowest accumulated level, and the highest accumulated rank at the same time. In case of ties, we give priority to the occurrence, then to the accumulated level, and finally to the accumulated rank. In the example of Fig. 4.2, Q2385804 and Q38723 are chosen at this step.

The final step consists in computing the distance between the chosen candidate types and the entity Thing (Q35120) in order to select the most specific type. We first consider the mutual inheritance relationship between the remaining candidate types. When such a relationship exists, we select the most specific type. If no inheritance relationship can be found, we compute the shortest distance to Thing. If there is still a tie among some candidate types, the system randomly selects one of them as CTA output.

---

[7]https://www.wikidata.org/wiki/Help:Ranking

		CTA			CEA			CPA		
		F1	P	C	F1	P	C	F1	P	C
Round 1 DAGOBAH SL	0.834	0.854	95.3%	0.922	0.944	95.3%	0.914	0.962	90.6%	
(Synthetic Tables)	MTab	0.926	0.926	-	0.987	0.988	-	0.971	0.971	-
Round 2	DAGOBAH SL	0.983	0.983	99.9%	0.993	0.993	99.9%	0.992	0.994	99.5%
(Synthetic Tables)	MTab	0.984	0.984	-	0.995	0.995	-	0.997	0.997	-
Round 3	DAGOBAH SL	0.974	0.974	99.9%	0.985	0.985	99.9%	0.993	0.994	99.9%
(Synthetic Tables)	MTab	0.976	0.976	-	0.991	0.992	-	0.995	0.995	-
Round 4	DAGOBAH SL	0.972	0.972	-	0.984	0.985	-	0.995	0.995	-
(Synthetic Tables)	MTab	0.981	0.982	-	0.993	0.993	-	0.997	0.997	-
Round 4	DAGOBAH SL	0.743	0.745	-	0.830	0.819	-	-	-	-
(Tough Tables)	MTab	0.728	0.73	-	0.907	0.907	-	-	-	-

Table 4.4: Results of the DAGOBAH system in Rounds 1, 2, 3, and 4 of the SemTab 2020 challenge. "F1" stands for F1-score, "P" stands for Precision, and "C" stands for Coverage. The coverage is the ratio between the number of annotations proposed by the system and the number of targets to annotate. We also report on the score of MTab as this is the challenge winner.

4.2.5 Evaluation

We used DAGOBAH SL during the SemTab 2020 challenge. In this venue, DAGOBAH SL leverages mutual influences between annotations allowing our team to obtain very competitive results on all tasks of the SemTab 2020 challenge.

Table 4.4 provides the annotation scores (F1-score, precision, and coverage) of our system for the three tasks in Rounds 1, 2, 3, and 4 of the challenge. It should be noted that during the Round 1, we built the lookup service based on an old version of Wikidata (from 2017) that contains fewer entities (50 millions vs. 86 millions in 2020 version), which has resulted in incomplete lookups, and thereby hindered the annotation results as discussed in Section 4.5. Note that the datasets used in Rounds 1, 2, and 3 are automatically generated in a synthetic way by adding some artificial noise (Synthetic Tables). Interestingly, in Round 4, beside a synthetic dataset, a novel corpus (Tough Tables [35]) is introduced and consists of a set of high quality manually-curated tables with complex patterns of cells (*e.g.*, ambiguous names, typos). The results clearly show that a simple, yet optimised, model can achieve very good performance on the SemTab 2020 synthetic corpus. This is partly explained by the fact that a row in any tables from this corpus is fully represented in the Wikidata Knowledge Graph (KG). As a consequence, a good lookup service with high coverage and a well-tuned matching strategy are enough to produce very competitive results.

We observe that one difficulty concerns the dynamic and evolving nature of KGs such as Wikidata. During Round 1, we used an outdated KG version. Hence, the lookup service was not able to retrieve relevant candidates for ≈ 5% of the target cells, leading to a significant drop

in performance, compared to the leader MTab. Challenge organizers should either consider distributing the reference KG to use alongside the tables to annotate, or consider tables with evolving annotations along the time, anticipating that new items or even properties may be added in the KG.

Regarding the SemTab 2020 challenge itself, two levers can be used to increase its usefulness to the scientific community. One of the issues to consider when addressing the problem of annotating tabular datasets is the dynamic nature of the data. In SemTab Challenge, tabular data sources are frozen and made available at the beginning of each round, thus addressing the issue of the evolution of tabular data over time. However, the knowledge graphs used for annotation (DBpedia for SemTab 2019 and Wikidata for SemTab 2020) also evolve, affecting the result produced by the annotation algorithms. In order to allow an unbiased evaluation of the approaches, it seems important that a dump of the knowledge graph to be used be provided by the organizers at the beginning of the challenge to ensure the same evaluation between systems.

The other main limitation of Rounds 1, 2, and 3 of SemTab 2020 resides in the nature of the data to annotate. Indeed, tables are synthetically generated from the Wikidata knowledge graph. Consequently, the proposed tables are relatively clean and lookup operations can easily match cells in the tables to entities in Wikidata even if noising techniques are introduced. In real-world applications, tabular data can contain complex values (*e.g.,* cells containing lists of entities using various separators, alternative entity names), artifacts (*e.g.,* data encoding problems, formatting errors, input errors, missing values), and more complex layouts (*e.g.,* merged rows/columns, multi-line headers, horizontal/vertical/matrix tables) making table manipulation and annotation much more complicated. In order to produce more robust, generic, and intelligent annotation systems, it seems important that evaluation corpora take these challenges into account in the future. An example can be found in the Tough Tables corpus from Round 4 which contains tables manually scraped from the Web. We observe a remarkable degradation in performance of CTA and CEA tasks given the complexity and ambiguity of this corpus.

4.3 Multi-Hop Relations and Soft Context Scoring

In this section, we present the scoring improvements based on DAGOBAH SL system introduced in Section 4.2. This enhancement provides a better representation and disambiguation of entities in exploiting their contexts more efficiently in the KG and an improved and flexible entity scoring that leverages both local information and global table information. The improvement introduces a new global scoring function which includes two new components: Multi-Hop Relation for exploiting the deeper context of knowledge graph entities (Section 4.3.1) and a Soft Scoring Mechanism for taking the cell position in the table into

account (Section 4.3.2). Finally, we report the result with the evaluation of the SemTab 2021 challenge (Section 4.3.3).

In the second edition of DAGOBAH SL, the new optimised confidence score function (Equation (4.5)) is set as following:

$$PSc(e_c, e_m) = Sc_{context}(\mathcal{N}_{graph}(e_c), \mathcal{N}_{table}(e_m)) \times e^{\gamma(Sc_{sim}(e_c, e_m) - 1)} \tag{4.5}$$

This new pre-score function is the product of a context factor and a literal factor $Sc_{sim}(e_c, e_m)$. The latter returns the highest Levenshtein-based matching ratio between the cell and the label and aliases of the candidate. Aliases are penalized with a ratio weighted by 0.9 since we consider labels to be more important. The amplification factor $\gamma \in \mathbb{N}^+$ determines the importance of the textual similarity. We empirically observed that 2 was an appropriate amplification factor for the SemTab challenge.

The improvements mainly concern the context factor defined as follows:

$$Sc_{context}(\mathcal{N}_{graph}(e_c), \mathcal{N}_{table}(e_m)) = \frac{\sum_i w_i \times sn_i}{\sum_i w_i} \tag{4.6}$$

where $\mathcal{N}_{table}(e_m)$ is the set of neighboring cells in the same row as e_m and $\mathcal{N}_{graph}(e_c)$ is the set of neighboring nodes of e_c in the KG[8]. For each neighboring cell $n_i \in \mathcal{N}_{table}(e_m)$, sn_i is its neighborhood matching score w.r.t. $\mathcal{N}_{graph}(e_c)$.

The new function setting solves two issues related to the context score calculation with respect to DAGOBAH SL introduced in Section 4.2:

Expensive evaluation. Each sn_i was evaluated by iterating over all context nodes in $\mathcal{N}_{graph}(e_c)$ to find the best matching node. Hence, a performance bottleneck arose when scoring a generic entity with millions of edges in the KG. For example, let's consider the cell "Belfort" in Figure 4.1 and the Wikidata candidate entity Q171545. To check whether the neighboring cell "Bourgogne Franche Comté" is in the context of Q171545, we performed a comparison with each of the $\sim$ 1,000 nodes in $\mathcal{N}_{graph}(Q171545)$, including Q142 (France), Q3371185 (Paul Faivre), etc. (Figure 4.3a).

One-hop graph contexts. $\mathcal{N}_{graph}(e_c)$ consisted of nodes only one hop away from e_c. Consequently, a neighboring cell $n_i \in \mathcal{N}_{table}(e_m)$ matching with a node two hops away from e_c was not considered in the context of e_c. For example, given the one-hop context of Q171545 (Belfort) in Figure 4.3a, we wrongly considered that Bourgogne Franche Comté had no relation with Belfort whereas it is the region of Territoire de Belfort (French department) whose capital is Belfort.

---

[8]Neighboring nodes are connected to e_c via predicate paths in the KG, regardless of the predicate direction.

It improves both the efficiency and the expressiveness of the context score by avoiding the exhaustive scoring and exploiting more expressive contexts for an entity via two-hop predicate paths.
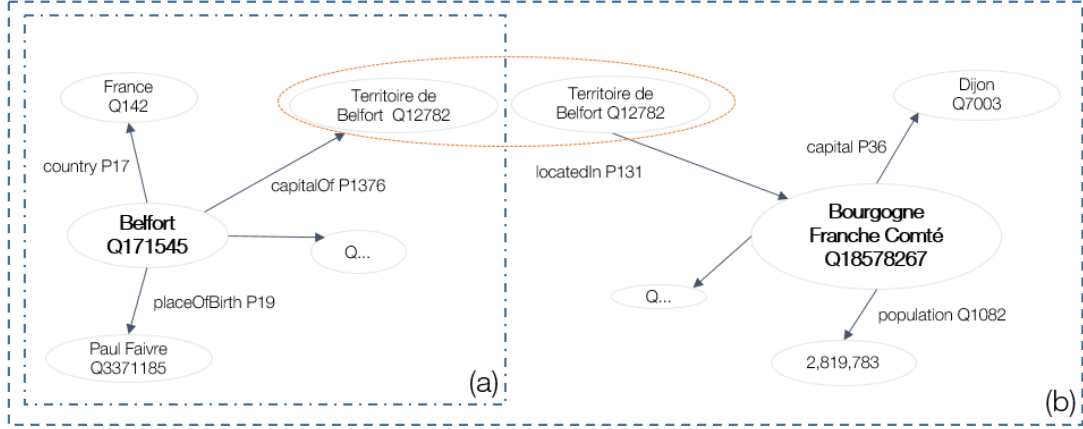

Figure 4.3: Graph context of entity Q171545 (Belfort) in Wikidata. (a) One-hop graph context of Q171545. (b) Graph context is expanded by sub-graph intersection.

4.3.1 Exploiting the Knowledge Graph Context with Multi-Hop Relations

The neighborhood matching score sn_i in Equation (4.6) indicates whether a neighboring cell n_i of e_m matches with a neighboring node of e_c. Loosely speaking, computing sn_i comes down to searching a candidate entity for n_i in $\mathcal{N}_{graph}(e_c)$ and to assessing its similarity. In our running example, Q18578267 is a candidate for cell "Bourgogne Franche Comté" in the two-hop context $\mathcal{N}_{graph}(Q171545)$ (Figure 4.3b). From this observation, we propose the following way to efficiently compute sn_i. The entity lookup step (Section 4.1) outputs candidate entities $\mathcal{E}_c(e_m)$ for a target cell e_m but also candidate entities $\mathcal{E}_c(n_i)$ for its neighboring cells n_i. Hence, we check if a candidate entity $e_i \in \mathcal{E}_c(n_i)$ is in $\mathcal{N}_{graph}(e_c)$. In that case, sn_i is simply calculated by comparing the labels of the neighboring cell n_i and the matching node e_i, which avoids additional comparisons with other nodes in $\mathcal{N}_{graph}(e_c)$.

To check if $e_i \in \mathcal{E}_c(n_i)$ is in $\mathcal{N}_{graph}(e_c)$, we actually check if e_i is connected to e_c by a predicate path in the KG. We chose to compute such predicate paths since they are useful in the soft context scoring. To efficiently find predicate paths between e_c and e_i, we extract the one-hop subgraphs $\mathcal{G}_{e_c}$ and $\mathcal{G}_{e_i}$ around e_c and e_i. If an intermediate node v is present in both $\mathcal{G}_{e_c}$ and $\mathcal{G}_{e_i}$, the paths pointing to v in the two sub-graphs are concatenated. In our running example, we find the following predicate path: Belfort $\xrightarrow{\text{capitalOf}}$ Territoire de Belfort $\xrightarrow{\text{locatedIn}}$ Bourgogne Franche Comté. Since we only consider one-hop subgraphs, paths can have a maximum length of two hops. This approach allows to enrich the information about an entity by including not only direct neighbors but also indirect neighbors two hops away. Such

enhanced graph contexts increase the chance for a neighboring cell $n_i \in \mathcal{N}_{table}(e_m)$ to match, and thus make the context score more precise. We argue that, in the context of STI with Wikidata, paths longer than 2 hops are often noisy and meaningless, and thus can have a negative impact on the context score.
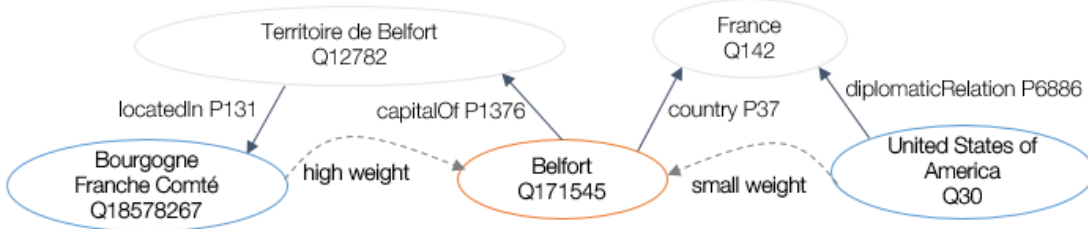

Figure 4.4: Neighboring nodes of Belfort (Q171545) contribute differently to its information content.

Hence, neighboring nodes of the candidate entity e_c in $\mathcal{N}_{graph}(e_c)$ may provide different information content as some neighbors can be "semantically closer" to e_c than others. To illustrate, consider the 2-hop context of Q171545 (Belfort) depicted in Figure 4.4. Q18578267 (Bourgogne Franche Comté) is more relevant than Q30 (United States of America) since the path Belfort $\xrightarrow{capitalOf}$ Territoire de Belfort $\xrightarrow{locatedIn}$ Bourgogne Franche Comté is more informative than Belfort $\xrightarrow{country}$ France $\xleftarrow{diplomaticRelation}$ United States of America. To quantify this, we rely on the so-called truth value $\tau(e_i)$ [27] of a neighboring node e_i, which can be seen as the discriminative capacity of the associated path $\tau(e_c \xrightarrow{p_1} v \xrightarrow{p_2} e_i)$, and is defined as follows:

$$\tau(e_i) = \tau(e_c \xrightarrow{p_1} v \xrightarrow{p_2} e_i) = \frac{1}{1 + log(g(v))} \tag{4.7}$$

where $g(v)$ is the generality of the intermediate node v, i.e., the number of its incoming and outcoming edges in the KG. Note that direct neighbors (or 1-hop paths) always get the highest truth value 1.0.

4.3.2 Soft Scoring

In Equation (4.6), neighborhood matching scores sn_i are weighted to compute the ultimate score of an entity. Indeed, each neighboring cell $n_i \in \mathcal{N}_{table}(e_m)$ contributes differently to the annotation of the target cell e_m with a weight w_i defined in Equation (4.8):

$$w_i = \frac{\overbrace{se_i}^{(4.8a)}}{\underbrace{\sqrt{d(col_i) + 1}}_{(4.8b)}} \times \overbrace{cnt(col_i)}^{(4.8c)} \times \overbrace{\tau(e_i)}^{(4.8d)}. \tag{4.8}$$

(4.8a) Cells containing entities should be more important than cells containing literals (*e.g.,*

date, measurement with/without unit, number) since there is a lack of literal disambiguation methods (*e.g.*, date-time normalization, unit detection/normalization/conversion). That is why, we set se_i to 1.0 if the neighboring cell n_i contains an entity, and to 0.15 if n_i contains a literal.

(4.8b) A neighboring cell on the left side of the table has more chance to be a meaningful context for the target cell. Hence, $d(col_i)$ is the distance between column col_i and the first object column of the table.

(4.8c) Cells n_i from a neighboring column highly connected to the target column should have a greater weight in the context. Hence, we take into account the connectivity $cnt(col_i)$ of the neighboring column w.r.t. the target column, defined as the highest occurrence of a relation potentially found between the two columns.

(4.8d) The Multi-hops Relations depicted in Equation 4.7.

4.3.3   Evaluation

To evaluate one-hop and two-hop graph contexts as well as the soft context scoring described in this section, we consider these four experimental settings:

Setting 1   The context score of an entity is computed using only its one-hop neighboring graph. Weights w_i do not follow Equation (4.8) but are set to 1.0 for entities and 0.15 for literals.

Setting 2   The context score of an entity is computed using its two-hop neighboring graph. Weights w_i do not follow Equation (4.8) but are set to 1.0 for 1-hop neighbors, 0.25 for 2-hop neighbors, and 0.15 for literals.

Setting 3   The context score of an entity is computed using its two-hop neighboring graph. Weights w_i follow Equation (4.8). This setting allows to assess if richer contexts and stricter scoring lead to better annotation.

Setting 4   This setting restricts Setting 3 to 1-hop and unidirectional 2-hop predicate paths in graph contexts. This allows to evaluate the impact of bidirectional paths which are often less informative or noisy but may be helpful in some cases.

We provide an experimental evaluation of the four aforementioned settings in Table 4.5. It should be noted that DAGOBAH is continuously improved. Hence, results of this evaluation are based on the current version of DAGOBAH but we also report results submitted to the SemTab challenge in gray cells for comparison. To validate the modifications proposed in Sections 4.1 and 4.2.1, we include the scores of the DAGOBAH 2020 system on tables annotated

with Wikidata in Round 1. Submission Settings {1,2,3,4}* are similar to Settings {1,2,3,4} but slightly differ in scores and weights initialization. This does not change CEA scores but impacts CTA performances. Indeed, CTA is highly sensitive to entity scores and taxonomy weights to select the most fine-grained type among the many possible (direct and ancestor) types of entities.

That is why CTA scores are different between Evaluation and Submitted. Indeed, our strategy for most fine-grained type selection is very sensitive to entity score and taxonomy weight. A small change in score may not have large impact on CEA (confirmed by the similarity in CEA evaluation and submitted scores), but can result a different fine-grained type knowing that an entity has many possible types (directs, ancestors).

We report CEA, CTA, CPA (if applicable) results for Round 1 WDTable, DBPTale datasets and Round 2 synthetic HardTable, BioTable datasets and Round 3 BioDivTable datasets. In our opinion, WDTa and HardTable are, respectively, the most difficult and easiest datasets of Rounds 1 and 2.

DAGOBAH achieves a high performance on synthetic datasets (Round 2) whereas high-quality manually-curated datasets with complex table patterns are more difficult to annotate (Rounds 1 and 3). In HardTable, no gain is brought by using a richer graph context or a more flexible scoring. This can be explained since tables are almost fully represented in the target KG and columns can be disambiguated from their contents. On the contrary, BioTable provides remarkable ambiguities with content overlaps between columns that hinder their disambiguation (*e.g.*, column "Gene" can be mistaken with column "Protein"). Hence, annotation seems to benefit from richer graph contexts. In BioDivTable, Setting 4 obtains the lowest scores whereas Setting 1 is comparable to Setting 3. We suppose that unidirectional 2-hop predicate paths may be noisy or not correctly considered, leading to the lowest score of Setting 4.

In general, Settings 2, 3, and 4 are more precise for CEA than Setting 1. Hence, 2-hop graph contexts bring useful information. The better performance of Settings 3 and 4 compared with Setting 2 shows the effectiveness of soft context scoring. We notice that Setting 3 achieves similar performances to Setting 4 which can be interpreted as follows. First, unidirectional paths (*i.e.*, $\xrightarrow{p_1}\xrightarrow{p_2}$ and $\xleftarrow{p_1}\xleftarrow{p_2}$) bring enough information and allow to obtain equal results compared with considering both unidirectional and bidirectional paths. Second, the influence of noisy bidirectional paths (*e.g.*, Belfort $\xrightarrow{country}$ France $\xleftarrow{diplomaticRelation}$ United States of America) is limited by the soft context scoring which prevents a degradation in the annotation quality. This allows useful bidirectional paths to contribute positively in the entity score. It can be observed that CTA and CPA performances are not as high as expected in most datasets despite CEA good performances. The development of better strategies for type and relation selection will be the subject of future works.

Table 4.5: Comparison of experimental settings and performance of the DAGOBAH system in Rounds 1, 2, and 3 of the SemTab 2021 challenge (in gray cells). "F1" stands for F1-score, "P" stands for Precision. Best results between settings are in bold.

Dataset	System Setting	CTA		CEA		CPA	
		F1	P	F1	P	F1	P
Round 1 – WDTable	Setting 1	**0.793**	**0.793**	0.913	0.913	-	-
	Setting 2	0.790	0.790	0.923	0.923	-	-
	Setting 3	0.783	0.783	**0.926**	**0.926**	-	-
	Setting 4	0.783	0.783	0.924	0.924	-	-
	DAGOBAH SL Basic	0.743	0.743	0.830	0.841	-	-
	Setting 2*	0.832	0.832	0.923	0.923	-	-
Round 1 – DBPTable	Setting 1	0.25	0.25	0.935	0.935	-	-
	Setting 2	0.27	0.27	0.946	0.946	-	-
	Setting 3	**0.274**	**0.274**	**0.947**	**0.947**	-	-
	Setting 4	**0.274**	**0.274**	**0.947**	**0.947**	-	-
	Setting 2*	0.422	0.424	0.945	0.946	-	-
Round 2 – BioTable	Setting 1	0.874	0.874	0.882	0.882	0.898	**0.901**
	Setting 2	0.911	0.911	0.916	0.916	**0.899**	0.899
	Setting 3	0.915	0.915	0.950	0.951	**0.899**	0.899
	Setting 4	**0.916**	**0.916**	**0.970**	**0.970**	**0.899**	0.899
	Setting 4*	0.916	0.916	0.970	0.970	0.899	0.899
Round 2 – HardTable	Setting 1	**0.968**	**0.969**	0.975	**0.976**	**0.996**	**0.997**
	Setting 2	**0.968**	**0.969**	**0.976**	**0.976**	**0.996**	**0.997**
	Setting 3	**0.968**	**0.969**	**0.976**	**0.976**	**0.996**	**0.997**
	Setting 4	**0.968**	0.968	**0.976**	**0.976**	**0.996**	**0.997**
	Setting 3*	0.976	0.976	0.975	0.976	0.996	0.996
Round 3 – BioDivTable	Setting 1	0.338	0.339	0.619	0.64	-	-
	Setting 2	0.335	0.335	0.60	0.62	-	-
	Setting 3	**0.344**	**0.345**	**0.62**	**0.641**	-	-
	Setting 4	0.343	0.343	0.475	0.491	-	-
	Setting 4*	0.381	0.382	0.496	0.497	-	-
Round 3 – HardTable	Setting 3*	0.99	0.99	0.974	0.974	0.991	0.995
Round 3 – GitTables DBP	Pre-processing + Mapping	0.07	0.117	-	-		-
Round 3 – GitTables SCH	Pre-processing + Mapping	0.183	0.185	-	-	-	-

Note that for BioDivTab and GitTables datasets, we adapted the DAGOBAH algorithms presented in this paper. Indeed, for BioDivTab, primitive types output by the pre-processing step were used to discriminate object and literal columns. A column contains literal values if its primitive type is numerical, date-time, unit, or miscellaneous. Otherwise, it is considered as an object column containing entity mentions that are passed to the lookup module. For GitTables, primitives types are manually mapped to Schema.org and DBpedia Ontology classes.

However, it is obvious that considering multi-hop relationships in the calculation of confidence weights will greatly increase the computational cost, which is the problem we are currently facing. In the future, we expect to use a blocking strategy to optimise the performance. One possible solution is that when computing a neighbor with distance 2, we do not calculate its soft scoring at first but its string distance instead. Only when its string similarity is higher than a certain threshold, we consider this neighbor in our soft-scoring calculation. At the same time, the scoring computation of each neighboring node is independent. Hence the use of parallelized distributed computing systems is one of the future directions to increase the speed of our operation.

4.4 Industrialisation

In this section, we briefly introduce our efforts on the industrialisation of the DAGOBAH project in Orange. Two tools built with the Orange teams are presented, which include a RESTful API of table annotation named DAGOBAH API (Section 4.4.1) and a Web based user interface named DAGOBAH UI (Section 4.4.2).

4.4.1 DAGOBAH-API

In this Section, we introduce DAGOBAH API. This RESTful API is deployed on the Orange Developer portal[9]. It provides pre-processing and annotation services for tables, as well as lookup services to disambiguate mentions and retrieve corresponding Wikidata or DBpedia entities. We also report the performance of current API service with stress tests in Section 4.4.1. This API is accessible to all company's R&D teams and business units upon request. We plan on extending the API access to external users in the future.

In DAGOBAH API, we provide 3 services:

- Entity Lookup API: a lookup service aims to find relevant entity candidates of an input label from a reference Knowledge Graph.

- Table Preprocessing API: a preprocessing system for tabular data. It involves reading

---

[9]https://developer.orange.com/apis/table-annotation

tables from files and extracting metadata from tables (orientation, header, key column detection, and column primitive typing recognition).

- Table Annotation API: a semantic annotation system/algorithm for tabular data. Its goal is to automatically understand the table by matching its elements with concepts, and relations of Knowledge Graphs, such as Wikidata, DBpedia or enterprise KG. The CEA and CTA tasks are activated for the entity column identified by Preprocessing module.

In the back-end of the API server, we decompose the features of the API into different services, including Spacy-based Named-entity recognition (NER), annotation, lookup, and preprocessing, etc. Each service is made of docker images deployed in a 64-bit Linux machine with 8 CPUs and 64Gb of RAM.

Stress test of DAGOBAH API

We drive a stress test for the current API service. We first evaluate the loading capacity of the API server. One table describing housing prices[10] with 20 million rows and an extended table[11] from Tough Table [35] are selected for this test. We launch a grid search by increasing 1,000 rows per run by querying large tables to the API with different row numbers until it returns an error. For the housing price table, we find that the limitation for Lookup is 34,000 rows and the annotation limitation is 22,000 rows. For the Tough table dataset, the lookup limitation is 30,000 rows and the annotation limitation is 23,000 rows. The returned errors are always out-of-memory issues.

For simulating multiple users, we also launched the same query from different docker endpoints with one selected table ("Y85GTOSS.csv" from Tough Table) with 4,000 rows. We observed a high probability of receiving a timeout stop once four users using the API at the same time[12].

In DAGOBAH API, we provide a hyper-parameter named K. This hyper-parameter affects the number of candidate entities generated by the lookup services. For a better understanding of the influence of K on the final precision and running time, we launched a grid search on Limaye dataset with a value of K from 50 to 400. We report the running time and the precision in Figure 4.5. We observe that, in general, increasing K brings a negative effect on both precision of the annotation and the total running time. However, when K reaches 200, we surprisingly see the precision and running time are un-normally and largely optimised. After multiple double-checks. We suspect that this may be due to the Lookup system that we are using switching from a method to another when K is around 200, or from a bug in the system.

---

[10]https://www.kaggle.com/datasets/hm-land-registry/uk-housing-prices-paid?resource=download
[11]Table 3C3INQLM.csv
[12]We have tested 4 times where 2 times ended with success and we receive the errors for the rest
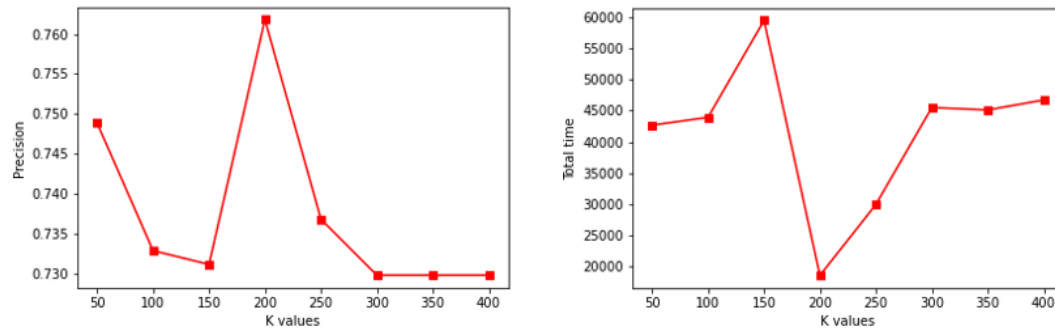
Figure 4.5: The evaluation of DAGOBAH API on Limaye with different K numbers: a) precision; b) running time

4.4.2 DAGOBAH-UI

In this Section, we introduce DAGOBAH UI, a user-friendly Web interface that enables to visualize, validate, or manipulate results of STI methods such as the DAGOBAH API. Through an interactive demonstration on real world datasets, we illustrate how such a UI can ease the adoption and mass usage of STI techniques by end-users. DAGOBAH UI is a Web user interface that makes use of the RESTful API exposing the DAGOBAH SL system [58] and the Wikidata KG [128]. The DAGOBAH-SL system annotates tabular data with elements of KGs such as Wikidata or DBpedia. This system was empirically evaluated during the three editions of the SemTab challenge [64] and has shown competitive performance with a 1st prize in the Accuracy track of the 2021 edition. Tabular data can be uploaded from the local file system or from the most commonly used benchmarks such as the SemTab datasets [64] and T2Dv2 [68].

Several tools can be related to DAGOBAH UI. For example, MantisTable is a Web application allowing to import and manage tables as well as trigger a specific semantic interpretation method [30]. MTab is a Web interface that allows to upload a table and display the resulting semantic annotations provided by the MTab tool [90]. However, DAGOBAH-UI provides additional features such as the ability to enrich a given table with information coming from the KG. OpenRefine[13] is a powerful tool for transforming and extending tabular data with external data. Its reconciliation functionalities are close to the CEA task in the STI process. However, while an end-user has to manually instruct OpenRefine how to annotate specific columns via ad-hoc rules, the semantic interpretation process is fully automatic in DAGOBAH UI.

The preprocessing toolbox of DAGOBAH UI allows table cleaning (encoding problems, cell misalignment, etc.) as well as extracting information about the topology of the table (orientation, header, etc.) which are crucial for the annotation process. The results of this process
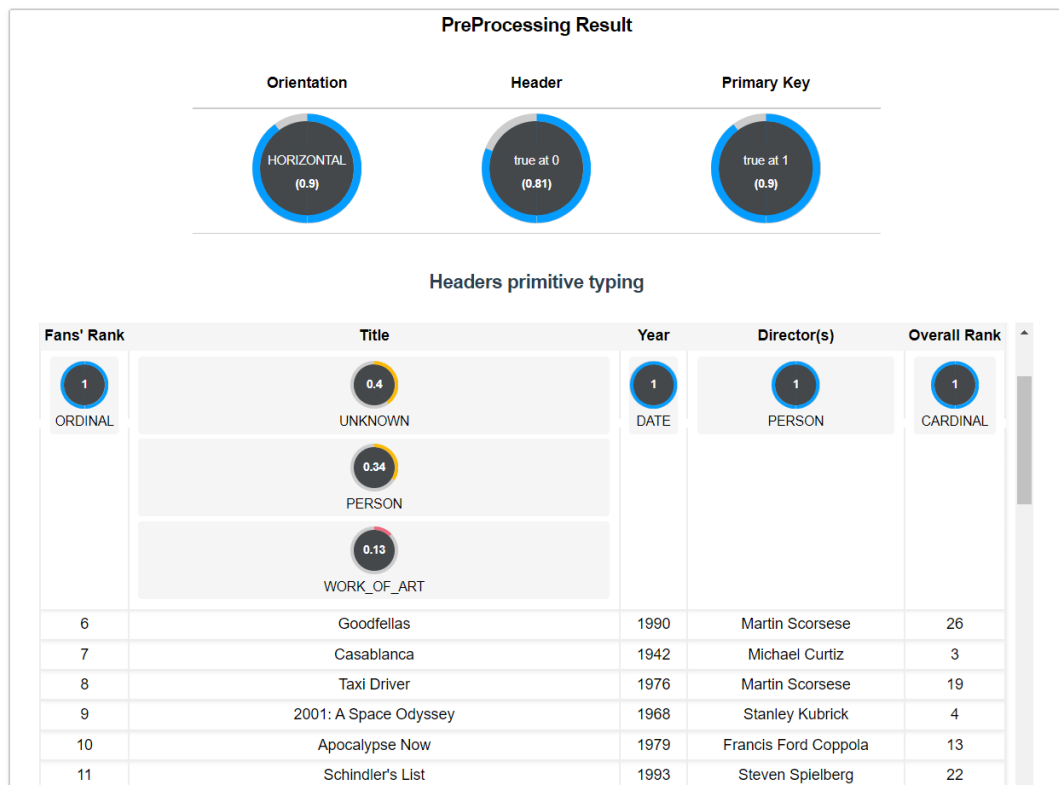
---

[13]https://openrefine.org/

Figure 4.6: DAGOBAH UI depicting the preprocessing on the T2D tables with the associated confidence scores.

on Table *77694908_0_6083291340991074532.json* (from T2D dataset [105]) are presented in Figure 4.6.

The user can then launch the semantic annotation process. The results of this process on Table *77694908_0_6083291340991074532.csv* (from SemTab 2019 dataset [63]) are presented in Figure 4.7.a. This consists of carrying out three tasks. CEA aims at associating a cell of the table with an entity of the KG. In DAGOBAH UI, the CEA annotation results are presented together with the original mentions. For example, "Star Wars" has been annotated with the Wikidata entity Q177738 (Star Wars: Episode IV - A New Hope). CTA aims to map each column with an entity type. In the user interface, these annotations are presented in the upper part of the table (in the headers). In the example, the system has annotated the "Title" column with the entity Q11424 (film). CPA seeks to associate pairs of table columns with a property of the KG. The relationships found are symbolised by links at the top of the table. When the user clicks on a link, the associated Wikidata property is displayed. In the example, the relationship P577 (publication date) has been identified between the columns "Title" and "Year". Figure 4.7.b shows the annotation results for another Web table about Movies generated partly from Wikidata. This example illustrates the power of the semantic elevation produced

by the annotation process. The system found that "Star Wars" in the SemTab table and "m/star_wars" in the Movie Table actually denote the same entity: Q177738 (Star Wars: Episode IV - A New Hope). This data reconciliation capability is particularly interesting for use cases involving heterogeneous datasets.
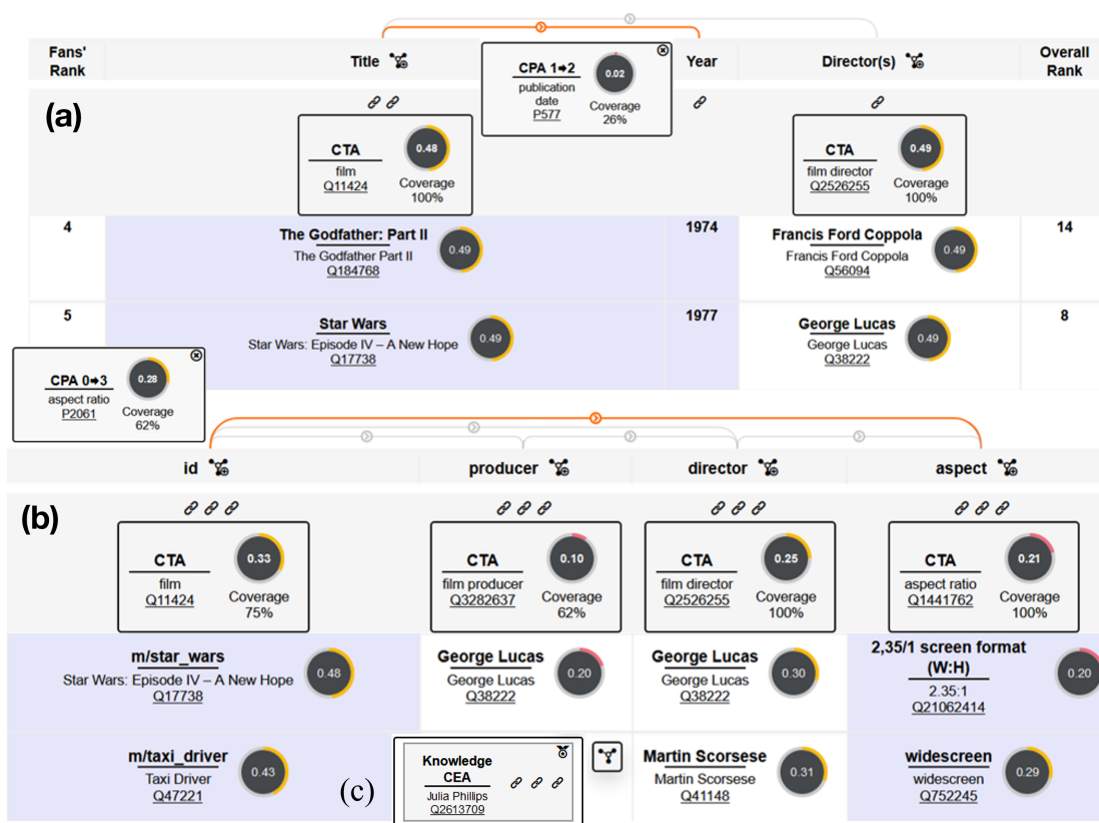

Figure 4.7: DAGOBAH UI depicting the semantic annotations on the SemTab and Movie tables with the associated confidence scores.

DAGOBAH-UI also allows users to enable downstream tasks following an STI annotation, such as filling missing cells and adding an additional column to the table, which is also known as data imputation and schema augmentation tasks. We will further introduce our effort in Chapter 6.

Our future work about DAGOBAH-UI includes the development of new features around KG enrichment from tables. As discussed in this section, tables can benefit from KGs through cell completion and table expansion with new columns. Conversely, tables are also a great source of dormant knowledge that can be leveraged to enrich open KGs. To this aim, DAGOBAH UI will enable to export of the annotations as RDF triples to complement KGs.

4.5 Discussion

In this chapter, we introduce the system DAGOBAH SL. DAGOBAH SL is a heuristic table annotation system that aims to handle CEA, CTA, and CPA tasks using Wikidata, DBpedia, and Schemas.org as the background knowledge graph. We further explain recent score function optimisations of DAGOBAH SL with new mechanism including the Multi-hop Relation scoring and Soft Context scoring. DAGOBAH SL has been evaluated through SemTab challenge series, and it has achieved state-of-the-art performance. The performance of the optimisations is reported with the SemTab 2022 dataset comparing to DAGOBAH SL base system. We also present the applied API for DAGOBAH SL with its performance, and a user interface named DAGOBAH UI.

We argue that DAGOBAH SL focuses only on the overlapped information between the table context and the given KG. Still, DAGOBAH SL somehow ignores the common themes of the target table during the disambiguation. Specifically, if multiple book titles appear in the same table, they are more likely to have something in common, such as coming from the same series or having the same author. However, DAGOBAH SL can not reveal such a topic from the table. This issue is related to the very nature of the data. Hence, the high performance of DAGOBAH SL for tables that are artificially and synthetically generated may not reflect what is actually found in the wild. More specially, tables often serve a specific purpose for the creator, and the attributes are selected accordingly. For example, one might want to use a table for presenting all books within the topic of Star Wars, but not all entities from the type of literary work (Q7725634). At the same time, the creator of this table might also want to focus on the publication dates without other attributes of books (e.g., the authors) in the table to emphasize that the Star Wars series are continuously updating[14]. A heuristic approach that leverages the information overlapping between the table and KG may not take advantage of the limited context of tables. Also, tables can be grouped into the collection with a common theme, but at the moment, DAGOBAH SL annotates tables very independently as if they were no notion of a collection. This context may typically not be made explicit in corpora but could be detected using topic modeling algorithms adapted to tables. This limitation of DAGOBAH SL points out the direction of optimising STI disambiguation.

---

[14]This example is actually modeled in the file 'file405599 0 cols1 rows23.csv' from the Limaye dataset

Chapter 5

Embeddings-Based Semantic Table Interpretation

For interpreting the content of a table, the key ingredient is to capture some form of context. The context can be immediate neighboring cells and the table structure as what has been discussed in Chapter 4, but also the relatedness from the components of the tables. The heuristics-based approaches from the previous chapter tend to rely solely on the context found in the table. In contrast, the embeddings-based approaches will leverage external sources, such as language models, including graph embeddings techniques, on tables or encyclopedic KGs that will provide additional information about components of the table (e.g. cells, title, columns). Also, leveraging the graph embeddings could be beneficial for digging more hidden themes of the target table or the semantic similarities between elements from the chosen table.

In this chapter, we present our experiments about using graph embeddings techniques for optimizing given STI tasks. We first introduce the concept of graph embeddings in Section 5.1. We then illustrate how to use clustering methods for STI in Section 5.2. Next, a CEA disambiguation plug-in tool named Radar Station is presented in Section 5.3. We finally disuss the results in Section 5.4.

5.1 Graph Embeddings

KG embeddings methods embed the elements (e.g., entities and relations) from a given KG into a given definite vector space and trained for achieving specific tasks (e.g. link prediction). They are algorithms often used as a pre-processing step of a graph with the goal of turning a graph into a computationally digestible format. KG embeddings facilitated various downstream tasks such as link prediction [131], recommendation [73], and clustering [119] as they transformed the symbolic knowledge presentation into mathematical vector representations.

Given the complex structure and large scale of some particular KGs (e.g. Wikidata, DBpedia),

effectively representing its structure and the underlying semantics is a crucial task [132]. This research direction is also known as knowledge representation learning [62]. Seen the attention that has been attracted by the KG embeddings, various methods have been proposed, in following we list following representative models:

- **TransE** [21] is a popular model that considers both entities and relations from the same vector space. The training intends to adjust the three vectors from a given triple (h, r, t) to the synchronized state until $h + r \approx t$. TransE provides an efficient method at scale and has been implemented in various downstream tasks. However, it also faces challenges from symmetry/one-to-many/many-to-one/many-to-many relations that TransE can not handle.

- **RotatE** [117] considers the relation as a rotational degree between heads and tails in the complex plane. It introduces a loss function based on $h \circ r \approx t$ for simulating the relation translation where $\circ$ denotes the elementwise Hadamard production. The knowledge representation is shown in a complex space that allows the model to capture more underlining semantics such as symmetry relations.

- **RESCAL** [91, 92] is based on a bilinear scoring function $h^T M_r t$. RESCAL aims to leverage a so-called three-way rank-r factorization on top of KG relation slice tensor. The shortcoming of RESCAL is that it requires many model parameters; hence, it is hard to apply on large KGs and quickly overfits as the rank grows.

- **DistMult** [140] is based on a bilinear scoring function $h^T diag(r) t$, where it is similar with RESCAL that M_r from RESCAL is restricted to diagonal matrices. However, because $h^T diag(r) t = t^T diag(r) h$, this model considers all relations as symmetric relations.

- **ComplEx** [124] also can be seen as a constrained variant of RESCAL [61] and extension of Dismult that leverages fewer relation dimensions inside a complex space. The ComplEx score is defined as $Re(h^T diag(r) \bar{t})$ and this function is not asymmetric anymore.

Table 5.1 illustrates the inference abilities of our chosen models [117]. In this table, the header "symmetry" indicates that the model could present the rule that if "(Bob, married, Mary)", then "(Mary, married, Bob)". "Antisymmetry" relation presents the rule that if "(Bob, is_the_father, Sam)", then the prediction "(Sam, is_the_father, Bob)" is false. "Inverse" depicts inverse relations, e.g. hypernym and hyponym. Some relations decomposed into multiple relations, and it is known as "composition", e.g. my mother's husband is my father.

These representative models could be classified into translational distance models and semantic matching models following the survey of [132]. Where translational distance models study the geometric distance between entities inside the vector space such as TransE or RotatE.

Table 5.1: The pattern modeling and inference abilities of presented models

Model	Score Function	Symmetry	Anti-symmetry	Inversion	Composition
TransE	$h + r \approx t$	✗	✓	✗	✗
RotatE	$h \circ r \approx t$	✓	✓	✓	✓
RESCAL	$h^T M_r t$	✓	✗	✗	✗
Dismult	$h^T diag(r) t$	✓	✗	✗	✗
ComplEx	$Re(h^T diag(r) \bar{t})$	✓	✓	✓	✗

Semantic matching models measure the similarity between entities and relations during the training with help of a neuron network, RESCAL, DisyMult, ComplEx are Semantic matching methods.

5.2 Using Clustering for Semantic Table Interpretation

This section presents DAGOBAH Embedding, a system that uses KG embeddings for disambiguating the annotations. It takes advantage of embeddings (sets of graph entities vectors) pre-trained and open-sourced. For our first experiment presented in this section, we chose to use the OpenKE [51] pre-trained TransE embeddings from the University of Tsingua since it is one of the richest and easiest to use at the time. The intuition behind this approach is that entities in the same column should be closed in the embedding space as they share semantic characteristics, and thus could form coherent clusters. We build our system based on the lookup introduced in Section 4.1.

5.2.1 System Description

The intuition of this approach is that the correct entities from the same table column should be close to each other in the embedding space. Hence, they should be in the same cluster after we use the clustering algorithm. Once we could successfully select the right clusters, we could largely reduce the noise during the interpretation and optimize the running time performance.

From this intuition, the first step consists in determining a group of candidates for each column of the input table by performing look-up operations (as described in Section 4.1) to extract the candidate entities and associated embeddings. The output of this phase is a set of vectors in the vector space representing the candidates potentially associated with the elements of our column (the columns are processed one after the other). The idea is then to launch a clustering algorithm in order to extract the probable candidates by making the hypothesis that the entities of the same column are semantically close.
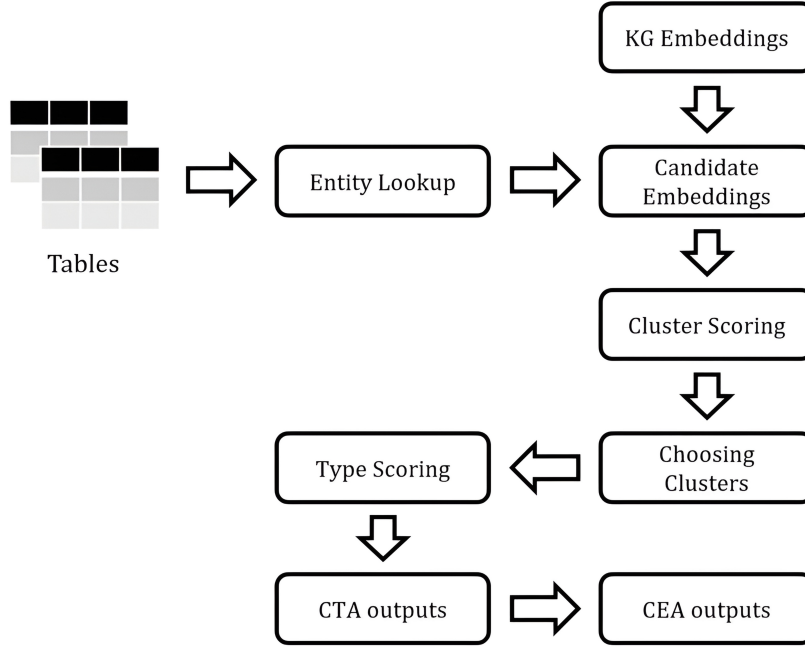
Figure 5.1: Illustration of DAGOBAH-Embeddings.

The challenge is to have most of the expected candidates for a given column in one of the clusters. A grid search strategy measuring precision of correct candidates clustering with different algorithms and K values was implemented to determine the best algorithm (K-means with hyperparameter K equals to number of lookup candidates divided by number of rows)(step 3 in Figure 5.1). In order to select the relevant cluster, a scoring algorithm has been defined (step 4 in Figure 5.1). Considering that, ideally, all good results are in the same cluster, this means that, conversely, one of the clusters contains a large number of rows of the target column. The cluster selection rule reflects this fact by selecting the cluster with the best coverage of the column under study. The clusters with the highest rows coverage (i.e. number of rows having at least one candidate in the cluster) are selected. Then, a confidence score is associated to each candidate within these clusters (Equation 5.1).

$$S_c(i) = item_{conf}(i) * item_{sim}(i)^x \qquad (5.1)$$

where $item_{conf}(i)$ is the co-occurrence score given by Equation 5.2 and $x \in \mathbb{N}^+$ allows to give more importance to the textual similarity.

$$item_{conf} = FE + 0.5 * FH \qquad (5.2)$$

where FE and FH are defined in Equation 5.3.

$$FE = \frac{e+1}{N_C} \quad , \quad FH = \frac{h+1}{N_C}$$ (5.3)

where e = number of entities in other columns matching with Wikidata properties values of the candidate; h = number of headers in other columns matching with Wikidata properties labels of the candidate; N_C = number of table columns. The normalized confidence score for a given cluster n is then computed using:

$$S_k(n) = \frac{\sum_{i \in n} S_c(i)}{len(n)}$$ (5.4)

where $len(n)$ is the number of elements within cluster n.

From all candidates in the selected clusters, a counting for every existing type is computed, each type inheriting the confidence score of its corresponding candidate (step 5 in Figure 5.1). All types with a score higher than a threshold ($Max(score) * 0.75$) is selected. Thus, the output type is the one having the highest specificity within the DBpedia hierarchy (using subclasses count and distance to $owl : Thing$). Finally, the candidates of each cell resulting from the lookup operations are examined according to the selected type (step 6 in Figure 5.1). The following score is computed for each lookup entity i belonging to cluster n:

$$S_e(i) = R_t * (0.2 * S_k(n) + 0.5 * S_c(i))$$ (5.5)

where R_t = 1.5 if the entity belongs to the type T produced in CTA, 1.2 if it belongs to a parent of T and else 1. From a given row candidate, the output entity is the one with the highest score. This system is described in Figure 5.1.

5.2.2 Determining the Number of Clusters and Choosing Clustering Algorithm

Choosing a suitable clustering algorithm and a suitable cluster number is always a crucial task. In our experiment, three algorithms are selected for further sample tests, which include :

- **K-means** [76] is a vector quantization method from signal processing commonly used for cluster analysis in data mining. K-means classification partitions n observations into k groups in which each observation belongs to the group with the smallest average distance, serving as the prototype for the group.

- **BIRCH** [151] algorithm uses a tree structure to facilitate rapid clustering, a numerical structure similar to the balanced B+ tree, generally referred to as the Clustering Feature Tree. Each node of the tree is composed of several Clustering Features. The BIRCH algorithm is more suitable for situations where the amount of data is large and the

number of K categories is also large. This algorithm has the advantage of being very efficient. Moreover, only one iteration of the data is needed to calculate the whole hierarchy.

- **Spectral Clustering** [85] is a widely used clustering algorithm. Compared to the traditional K-Means algorithm, spectral clustering adapts more easily to different data distributions, and the clustering accuracy is also excellent. It is a more efficient algorithm than K-means because less computation is required and its implementation is quite simple. Spectral clustering is an algorithm derived from graph theory. The main idea is to treat all data as points in space, which can be connected by edges. The weights of the edges depending on the distance between the points. Counterintuitively, the closer the points are, the higher the weight, and vice versa. The algorithm then looks for cuts in the graph in order to generate sub-graphs in which the weights between nodes are as high as possible and the weights between sub-graphs are as low as possible. The output clusters of the algorithm are the clusters thus generated.
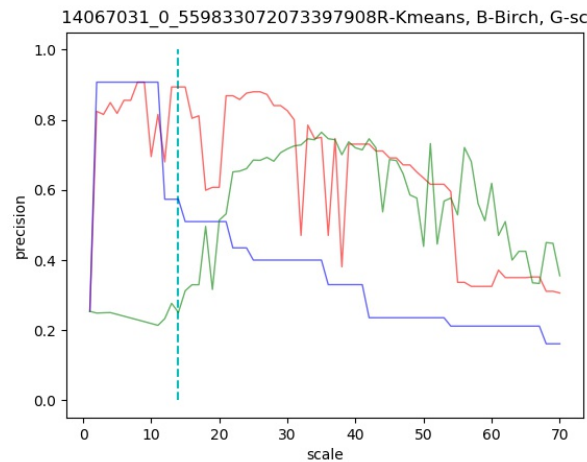

Figure 5.2: Results of the clustering evaluation for table 34041816_1_4749054164534706977. Red: K-means; Blue: BIRCH; Green: Spectral Clustering; Dotted line: scale = p.

It is difficult to choose the right algorithm and the right number of clusters (K value) since we might have different K values for different tables and different columns. To better observe how the K value affects our performance of the system regarding different clustering algorithms, we launched a grid-search procedure on 15 sampled tables. The idea of this process is to run the three clustering algorithms with a number of clusters fixed at a value between 0 and five times p on the sampled table from T2D dataset, where p is an index equal to the number of look-ups divided by twice the number of rows of the table to be annotated. We aim to discover a suitable function to determine the K value.

The visualisation of results of the grid-search on two sampled tables of the corpus are given
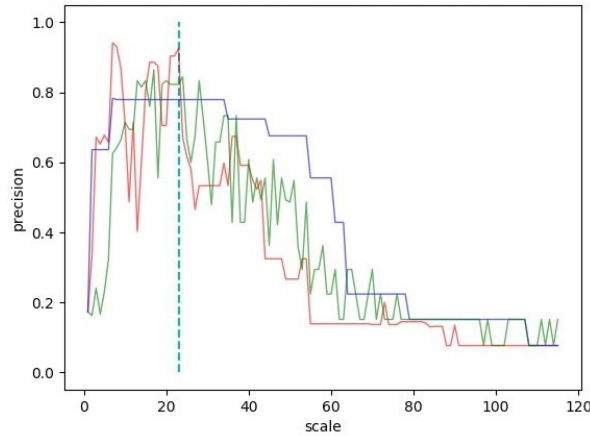
Figure 5.3: Results of the clustering evaluation for table 54719588_0_8417197176086756912. Red: K-means; Blue: BIRCH; Green: Spectral Clustering; Dotted line: scale = p

Table 5.2: Results of the baseline and DAGOBAH embedding approaches for the first rounds of the challenge

	Task	CTA		CEA		CPA	
	Criteria	Prim. Score	Sec. Score	F1 Score	Precision	F1 Score	Precision
Round 1	Baseline	0.479	0.242	**0.883**	**0.892**	0.415	0.347
	DAGOBAH Embedding	**1.212**	**0.336**	0.841	0.853	-	-

in Figure 5.2 and 5.3, and the 13 other tables are shown in the Appendix A. The ordinate axis indicates the F1 score (modified by multiplying the recall by two in order to make the clustering more permissive and to accept a maximum of candidates in the clusters, at the expense of precision). The x-axis indicates the expected number of clusters (value of the hyperparameter). The three algorithms, K-Means, Birch, and spectral clustering, are respectively represented in red, blue, and green. Also we proposed a K value function so-called p-index, where it equals the number of the total candidates divided by the number of the row from the target column. In the graph, we have represented the p-index (the number of lookup candidates divided by the number of lines of the target table) with a blue vertical bar. Following this sampled research, we empirically found that the K-Means algorithm obtained better results since it is more stable, and its peak point is highly correlated with the p-index than the two other algorithms in our chosen sampled tables. On the whole corpus of evaluation, we noticed that this index gave, in the majority of the cases, the most adapted number of clusters.

5.2.3   Evaluation

We first compare the result of the clustering algorithm with a baseline which only depends on majority voting after a lookup research without any further operations on the SemTab 2019 challenge Round 1, we show the result of this evaluation in Table 5.2.
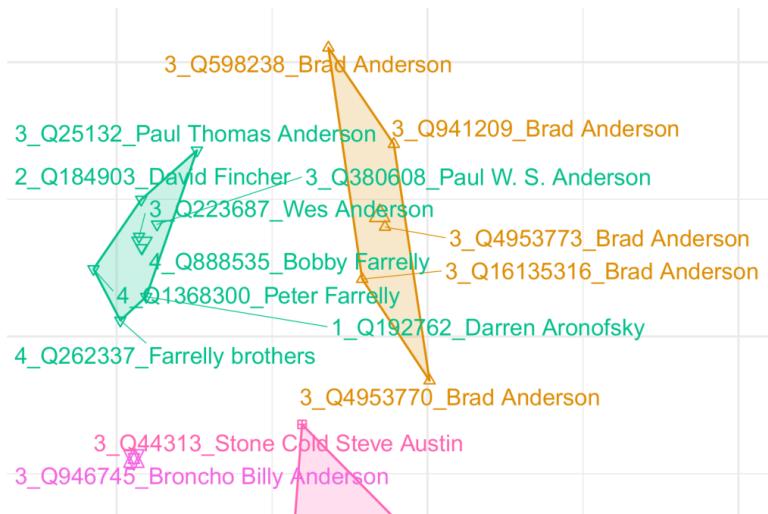
Figure 5.4: Result of K-means clustering applied to Wikidata embedding

The CEA's results of the baseline are satisfactory, but the baseline has difficulty in producing the CTA results expected by the evaluator. In this selected baseline, the predicted type was often too generic or too specific. In addition, the baseline showed two important limitations: a high dependency on lookup services (over which DAGOBAH has little control) and difficulties in correctly setting up algorithms (in particular finding the right compromise between specificity and representativeness of types in the case of CTA). Concerning the CPA, a simple lookup technique on the header was used during round 1 explaining the low accuracy.

To show the contribution of the embedding approach, an evaluation was carried out on the corpus. CEA performances are slightly poorer because of basic lookup strategies used (compared to the fully-optimised lookups used in the baseline) and the absence of expected candidates in the selected clusters. But the embedding approach proves to be highly proficient to determine the type of a column which is the core of more reliable annotations. In addition, the results are particularly interesting in cases where string mentions in the original table are incomplete. In table 54719588_0_8417197176086756912 for instance, one column references movie directors only by their family names (in that case, our baseline performance was poor). Doing K-means clustering on a subset of this table (four rows) performs pretty well even with very few data, as illustrated in Figure 5.4 (2 clusters shown among 12). Indeed, the green cluster gives the expected candidates, even if disambiguation still has to be done for some cells (using S_e).

We also compare this method with DAGOBAH-SL and MTab, two state-of-the-art heuristic STI methods, during the first round of the SemTab 2020 as illustrated in Table 5.3. We notice that DAGOBAH SL and MTab perform much better than DAGOBAH Embedding. The drop in the scores is arguably derived from the fact that DAGOBAH Embedding retains only a limited

Table 5.3: Results of the Embeddings system in Rounds 1 of the SemTab 2020 challenge. "F1" stands for F1-score, "P" stands for Precision, and "C" stands for Coverage. The coverage is the ratio between the number of annotations proposed by the system and the number of targets to annotate. We also report on the score of MTab as this is the challenge winner.

		CTA			CEA			CPA		
		F1	P	C	F1	P	C	F1	P	C
Round 1	DAGOBAH Embedding	0.779	0.803	95.3%	0.776	0.843	91%	0.809	0.958	73%
	DAGOBAH SL	0.834	0.854	95.3%	0.922	0.944	95.3%	0.914	0.962	90.6%
(Synthetic Tables)	MTab	0.926	0.926	-	0.987	0.988	-	0.971	0.971	-

number of clusters for the entity disambiguation, which does not always involve all the good candidates given the imperfection of the current clustering algorithm.

5.3 Radar Station

Cell-entity annotation [63] (CEA) is one of the fundamental tasks for STI. This task is often performed by retrieving and scoring possible entity candidates (from a target KG) to disambiguate a cell value. Next, the result is used as input for performing Column-Type Annotation (CTA) and Columns-Property Annotation (CPA) [1, 26, 58, 89]. However, associating a mention contained in a cell with an entity in a KG is a complex task requiring the resolution of several issues including handling properly the syntactic heterogeneity of mentions (e.g. the Wikidata entity "France" (Q142) may be referenced in a table by mentions like "The Republic of France" or "FRA"), the polysemy of terms (e.g. "Apple" can refer to a fruit or a company), and the diversity and complexity of table formats and layouts (e.g. matrices, relational table with hidden subjects, etc.).

Numerous approaches have been proposed for handling these issues. Among these methods, heuristic-based iterative approaches [1, 26, 58, 89] aim to leverage the column types and the inter-column relationships aggregated by voting strategies for disambiguating cell annotations. They have demonstrated to be the methods reaching the best performance in the SemTab challenge series [36, 63, 64]. However, one drawback of these strategies is related to error propagation. Often, such systems propagate the entity annotation error in the target column. Furthermore, they also often ignore other column-wised semantic similarities: for example, books appearing in the same column may share the same topic.

To address these limitations, we propose a new hybrid disambiguation system called Radar Station[1] that takes advantage of both an iterative disambiguation pipeline and semantic disambiguation using graph embedding similarities. Radar Station takes as input CEA annotations and associated confidence scores that quantify the level of certainty associated with each

---

[1]We made this system open-sourced in GitHub: https://github.com/Orange-OpenSource/radar-station

result. Our approach uses an ambiguity detection module that detects cases where the cell annotation is potentially wrong due to error propagation. In the following steps, the use of graph embeddings allows Radar Station to potentially fix the wrong annotations by taking into account semantic proximities (e.g. geometric proximity of entities representing books) that are not directly encoded and captured in the sole content of table columns. We evaluate Radar Station using several graph embedding models belonging to different families on Web tables as well as on synthetic datasets, and we provide a thorough analysis of the performance among the graph embeddings models and the datasets.

5.3.1   Motivation

STI covers five main tasks as introduced in Section 3.2. In these tasks, Radar Station aims to improve the CEA disambiguation. Thus, this section reviews the current state-of-the-art methods for the CEA task on relational tables. We classify them into three groups: heuristic-based approaches, iterative disambiguation, and graph embeddings approaches, and we discuss their strengths and limitations [74].

Heuristic-Based Approaches

Starting from a basic lookup service that generates target candidates for a given cell mention, heuristic-based approaches leverage diverse methods interpreting the table context to filter unreliable candidates and to produce a final annotation. Based on heuristic candidate generation and string similarities measures, [72] is one of the first works on STI. It constructs a graph-based algorithm that exploits learnable features from column context, row context, and relation context to construct a confidence function for each candidate for annotating a cell. TabEL [16] introduces a hybrid system that leverages probabilities to build a graphical model for representing the interactions between cells, columns, and headers. ADOG [94] generates features from string similarities, frequencies of properties, and the normalized Elasticsearch score. Then, these features are calibrated with the candidate's TF-IDF score according to entities' types in the same column. Our approach takes as input a list of CEA candidate annotations together with their scores (generated by such an existing CEA annotation tool), and detects the presence of potential ambiguities in order to select the right candidate from this closed set.

Candidate Disambiguation

Adding a disambiguation process on top of a heuristic-based approach can significantly improve the performance of an annotation system. Iterative processing is one of the most commonly-used methods for improving pre-annotated results. The iteration loop aims to

collect the results of several annotation tasks, mutually improving the compatibility between annotations (e.g. taking into account the type of a column produced by the CTA to choose the right CEA candidates), and increasing the scores of candidates that would not have been chosen in the first place. For example, [156] uses a loop that exploits the CTA annotation of a given column to select candidate cells that feature that type and then redefines a new CTA annotation for the column by exploiting the entities selected. Regarding the CEA disambiguation, we identified two classes of iterative systems. First, T2K [104] and TableMiner+ [153] introduce a loop in the pipeline that ends when the result becomes stable. The other iterative systems [1, 26, 58, 89] provide a predefined pipeline with sequential modules (e.g. the pipeline of LinkingPark [26] is composed of a CEA pre-scoring, then a CPA step, and finally the use of the CPA annotations to generate the final CEA annotations).

Radar Station uses the output scores of an existing STI system. It currently supports the DAGOBAH-SL [58, 59], MTab [89] and BBW [110] systems which have all competed during the SemTab Challenge series [36, 64] and are selected as baseline systems during the evaluation of Radar Station. These systems use string similarity in the scoring system and leverage the table's global information carried out by the CTA and CPA annotations to generate more precise CEA annotations. For cell annotations, they evaluate whether a candidate entity $e_c \in \mathcal{E}_c(e_m)$ retrieved from the KG is a good representation of the corresponding table cell e_m by incorporating the table context of e_m and KG context of e_c in the score of e_c. Although these approaches show great performance for datasets like BioTable and HardTable from SemTab, they still have limitations as described in Section 5.3.1. First, the use of a unique column type or columns pairs relationship potentially propagates type (resp. relation) annotation error through cell annotations. Second, leveraging only entities' type (resp. relations) result does not allow to take into account more attributes and properties in the disambiguation process. For example, a column type may not bring necessary information such as a person's nationality, building localization, or object ownership for disambiguating entities. Facing these challenges, Radar Station first activates an ambiguity detection module that detects cases where the cell annotation is potentially wrong. Meanwhile, it considers entities' embeddings to leverage more similarity measures inside a given column.

Usage of Graph Embeddings

Methods applying graph embeddings for STI focus on entity-level in which the models learn embedding representations for entities of a table cell instead of the cell itself. Specifically, KG embedding techniques are used to encode the entities and their relationships into a vector space. STI approaches using deep learning models are based on the intuition that the entities in the same column should exhibit semantic similarities. Hence, they should be close to each other in the embedding space w.r.t. a cosine similarity distance [41] or a Euclidean distance [24].

Vasilis et al. [41] provide different methods. One of them assumes that the correct CEA candidates in a column should be semantically close. From this assumption, a weighted correlation subgraph in which a node represents a CEA candidate is constructed. The edges are weighted by the cosine similarity between two related nodes. The best candidates are the ones whose accumulated weights over all incoming and outcoming edges are the highest. In addition, a hybrid system combining a correlation subgraph method and an ontology matching system, is also introduced, which considerably improves the final result. Yasamin et al. [44] further enhance this approach by taking the header of the table into account for ontology matching and giving more weights to unique cell candidates when calculating embeddings Page-Rank. Our previous work, DAGOBAH-Embeddings [24] (Illustrated in Section 5.2), follows the same assumption that all entities in the same column of the table should be close to each other in the embeddings space. Consequently, the correct candidates are assumed to belong to a few clusters. They apply a K-means clustering using TransE pre-trained KG embeddings to cluster the entity candidates. The good clusters with high coverage are selected by a weighted voting strategy. Experimental results prove that they have successfully improved the accuracy of the CTA task. However, the system is also misled by incorrect candidates during the CEA task when the correct candidates are not in selected clusters. TURL [38] leverages the BERT model for STI and table augmentation with the help of a visibility matrix for capturing table structure. Although TURL introduces entity embeddings as one of the inputs to its model to assign information to entities, the entity embeddings do not embed properties about the entities in the graph, such as the fact that neighbouring nodes are missing in them.

The contributions of our approach are as follows. First, we use embeddings only during the disambiguation step to benefit from both the iterative disambiguation and the embeddings disambiguation. Second, we provide a new scoring mechanism that takes into account the scores generated by CEA approaches and the distance between the entities in the embedding space.

5.3.2 System Description

Radar Station is not a standalone annotation system. It is built on top of a given annotation system and resolves ambiguities detected in the annotated results. We choose to use DAGOBAH-SL [108] as the base annotation system to illustrate the process of Radar Station. We motivate the need for Radar Station observing that pure string-based matching and iterative scoring methods are limited in situations where: i) the target KG is incomplete; ii) the matching mechanism failed; iii) the CTA or CPA disambiguation can not provide enough information in very ambiguous cases (e.g. candidates belonging to the same type or no property identified). These situations also cover cases with limited row numbers that can annotate a unique column type (resp. unique columns relationship) by majority voting. For example,

voting for a common type given only the two cell mentions "Apple" and "Blackberry" may lead to randomly select the company or the fruit.
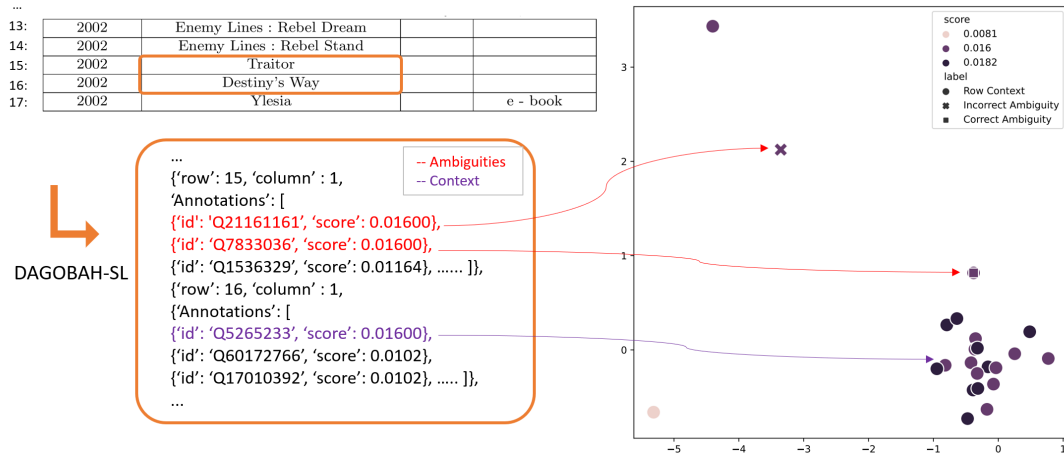

Figure 5.5: Illustration of Radar Station with DAGOBAH-SL results. The plot is generated with RotatE embeddings after dimension reduction by T-SNE.

Figure 5.5 provides an example where DAGOBAH-SL is not able to handle properly an ambiguous case: two potential candidates with the same score for the cell "Traitor". The ambiguity comes from an unsuccessful matching between the column "2002" with literal information "30 July 2002" of candidate "Q7833036" for entity scoring and CPA disambiguation. CTA disambiguation does not work in this case since these two candidates are books with the type "literary work" ("Q7725634"). "Q21161161" is a science fiction novel and "Q7833036"is an anti-wars romance novel.

We do not aim at improving the performance of the system by relying on clever string-matching methods. Instead, we expect to find more semantic similarities using the full column as context with the help of the scores generated from the row context. In this example, one could identify that the correct entity is "Q7833036" since the topic of this table is the science fiction series "The New Jedi Order" from Star Wars. This relationship is missing in the table cells, but it still could be beneficial for the disambiguation steps. Radar Station aims to leverage graph embeddings to dig similarities alongside the entity types and common relationships inside the tables. The architecture of Radar Station is illustrated in Figure 5.6 and the modules are described in the following sections. Table 5.4 summarizes the notation used in the Radar Station approach.

Input Data Structure

Before running Radar Station, the information required by the system includes the index of the cell in the table (row number and column number), and information about all candidates
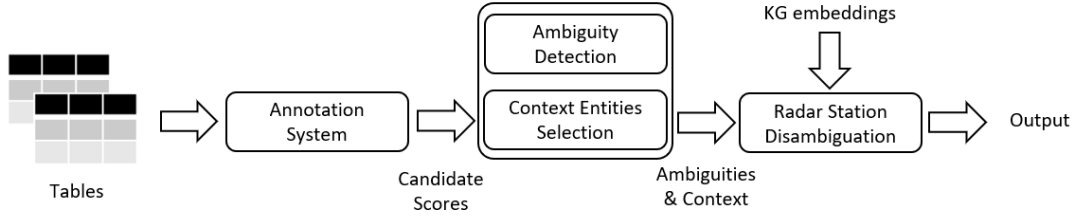
Figure 5.6: Overview of the Radar Station pipeline.

Table 5.4: Summary of the notation used to define Radar Station

Notation	Description
$\mathscr{C}$	The collection of cells from the target column
$\mathscr{E}_c$	The collection of the context entities representing the column $\mathscr{C}$
$\mathscr{A}m_{c_i}$	The collection of the ambiguous entities extracted from the cell c_i
$\mathscr{S}c(e)$	The initial score for the candidate e generated from the previous annotation system, in our case, DAGOBAH-SL
$\mathscr{E}m(e)$	The embedding of a given entity e
$\mathscr{E}_k$	The collection of K nearest context entities of an ambiguous candidate $am \in \mathscr{A}m_{c_i}$ for the target column $\mathscr{C}$, $\mathscr{E}_k \in \mathscr{E}_c$

for each cell without filtering the candidates. This information includes an identification of each candidate and their confidence score. The confidence score evaluates how compatible a candidate is with the context information given by the table (e.g. row values, column type, columns-pair relations).

Context Entities Selection

The row context has already been interpreted by DAGOBAH-SL and is used to compute the confidence score of each candidate. The first step of Radar Station is to build a column-wised context to support the disambiguation process. We collect entities with the highest confidence score from all cells of a given column $\mathscr{C}$ as the context entities set. In case of ambiguity, that is, multiple candidates (n candidates) sharing the same highest score, we collect all of them, and the score is divided by n. Other candidates are not taken into account to maximize the trust for "sure" annotation from DAGOBAH-SL (e.g., only one candidate with the highest score) and to avoid noise inside this column. For example, for the row 15 in Figure 5.5, both "Q7833036" and "Q21161161" are collected into the context set with a score "0.008" (0.016/2), and for row 16, only "Q5265233" is collected with a score "0.016". The collected context entities set for the column $\mathscr{C}$ are noted as $\mathscr{E}_c$.

Ambiguity Detection

Radar Station detects ambiguous cells that are worthy to be disambiguated given a tolerance t. Intuitively, t enables to relax the constraints one wants to have in looking up candidates potentially matching a cell mention. Once a candidate's score is larger than $t * Max(scores)$, it is selected as one of the "top candidates". For example, if we set $t = 1$, "Q7833036" and "Q21161161" for row 15 of Figure 5.5 will be among the top candidates. If we relax the tolerance t to 0.7, "Q1536329" will also be considered as a top candidate. We denote "Ambiguities" as Am for the case that the size of the top candidates is greater than or equal to two. Radar Station is activated in this case and it will annotate the cell with one of the candidates from the ambiguities. When there is no ambiguity inside a cell, we directly output the single top candidate as the annotation.

Radar Station Disambiguation

Algorithm 1 Radar Station disambiguation algorithm

Require: Cell index $\mathscr{C}$ and ambiguities for each cell $\mathscr{A}m_{c_i}$, $c_i \in \mathscr{C}$ where the collected context entities (or senders) of the target column is $\mathscr{E}_c$.
  Candidate scores from the annotation system $\{Sc(e_i)\}$, $e_i \in \mathscr{E}_c$.
  Candidate embeddings $\{Em(e_i)\}$, $e_i \in \mathscr{E}_c$.
Ensure: Entity annotation selected by Radar Station.
 1: build a KD-tree with all candidates' embeddings $\{Em(e_i)\}$
 2: $K \leftarrow min(|\mathscr{E}_f|, 20)$
 3: **for** each cell c_i from $\mathscr{C}$ **do**
 4:   **if** there is an ambiguity in $\mathscr{A}m_{c_i}$ **then**
 5:     $\mathscr{E}_c \leftarrow$ filter entities from the same cell in $\mathscr{E}_c$
 6:     **for** each ambiguous entity am_i in $\mathscr{A}m_{c_i}$ **do**
 7:       find the K nearest candidates of the ambiguous entity $\mathscr{E}_k$ in the KD-tree by ignoring candidates from the same cell.
 8:       $RadarScore_{am_i} \leftarrow 0$
 9:       **for** each neighboring entity $e_j \in \mathscr{E}_k$ **do**
10:         $RadarScore_{am_i} \leftarrow RadarScore_{am_i} + \frac{Sc(e_j)}{distance(am_i, e_j)}$
11:       **end for**
12:       $RadarScore_{am_i} \leftarrow \frac{RadarScore_{am_i}}{K}$
13:       $g(am_i) \leftarrow \alpha RadarScore_{am_i} + Sc(e_j)$
14:     **end for**
15:     the annotation is the ambiguous entity with the highest $g(am_i)$
16:   **end if**
17: **end for**

In our approach, we leverage KG embeddings to uncover the entities' co-relationship from a table to improve the disambiguation step. The principle of the Radar Station approach is inspired by radar station signal emissions. The receiving signal power of a signal station

depends on both the initial power strength from the sending station and the distance between the sender and the receiver. That is, the receiving signal will be stronger when the initial power from the sender is stronger, and this receiving signal strength will decrease as the distance increases. In our approach, we treat each context entity from the same column as a signal sender, and receivers are the ambiguities to be resolved. One ambiguous candidate captures signals from multiple neighbouring context entities (i.e. senders) and the sum of the receiving signals is the confidence score of the candidate. The disambiguation pseudo-code is presented in Algorithm 1.

We consider only the K nearest context entities to computer the final score of an annotation in order to avoid noise and to optimize the performance. The system first constructs a K-Dimensional tree (KD tree) of all context entities for each column, and then calls this KD tree to drop the K nearest context entities during the prediction (lines 1-2). We set that the maximum K value is 20 (line 2). We set the initial sender power strength with the confidence score generated by DAGOBAH-SL. One ambiguous candidate am_i detects K received signals from the surrounding senders $e_j \in \mathscr{E}_c$ (or context candidates) to generate the confidence score $f(am_i)$ with the Function 5.6 (line 3-12), where $Sc(e_j)$ denotes DAGOBAH-SL scores of the sender e_j and $distance(am_i, e_j)$ denotes the Euclidean distance between the sender e_j and the receiver am_i.

In detail, for a target am_i, we collect its top-K nearest neighbors from $\mathscr{E}_c$, where each context entity c_j belongs to $\mathscr{E}_c$. We have each context entity's scores $Sc(c_j)$ and the distance with the target candidate $distance(am_i, c_j)$. We then apply the Function 5.6. Like this, we could generate a confidence score for each of those two target candidates. We divide each of their context entities' confidence score by the distance between those two candidates and then calculate the sum to compare.

$$f(am_i) = \frac{1}{K} \sum_{j<K} \left(\frac{Sc(e_j)}{distance(am_i, e_j)} \right) \tag{5.6}$$

The final result $g(am_i)$ for an ambiguous entity is the combination of Radar Station score $f(am_i)$ and the initial DAGOBAH-SL confidence score $Sc(am_i)$ introduced in Function 5.7 (line 13).

$$g(am_i) = \alpha f(am_i) + Sc(am_i) \tag{5.7}$$

Our initial experiments showed that the average distance in the embedding space between the target ambiguity and its top K nodes is approximately 1. According to the Function 5.6, we know that $f(am_i)$ and $S_c(am_i)$ are roughly in the same order of magnitude. Since we expect to disambiguate candidates with a tolerance between 0.7 and 1, we need the value of the discrepancy caused by $f(am_i)$ to be roughly within $S_c(am_i) * 0.3$. We originally set α to 0.3

and we tested the following α values (0,3, 0.2, 0.1, 0.05, and 0.01). We empirically observed that 0.05 gives the best results.

5.3.3 Implementation

In our experiments, we consider Wikidata as the target KG. We first rely on the DAGOBAH-SL system to lookup for candidates for each entity cell. We only consider the top 100 candidates according to the string similarity on entity label and aliases. We evaluate the result on four different gold standard datasets: T2D [104], Limaye [72], Tough Tables version 2 [35] and ShortTables.

Knowledge Graph Embeddings

Pre-trained KG embeddings can provide additional information for table understanding beyond the table context. Entities inside the same table column should be somehow co-related, which means they may share the same entity type, similar topics, or even attributes. In order to have the most suitable embeddings given the latest version of Wikidata, we use the PyTorch-BigGraph framework [70] for training embeddings. The triples used for the training are collected from a Wikidata dump published in May 2021[2]. Before the training, the triples with literal values and Wikimedia disambiguation page entities (e.g. "Q1151870") are filtered out. The selection of the final embeddings is made given our empirical evaluation of Radar Station after fine-tuning the hyper-parameters. We consider two representative translational distance models (TransE [21] and RotatE [117]) and two semantic matching models (DistMult [140] and ComplEx [124]) following the classification of [132].

Translational distance models study the geometric distance between entities inside the vector space. TransE [21] considers both entities and relations from the same vector space. The training intends to adjust the three vectors from a given triple (h, r, t) to the synchronized state until $h + r \approx t$. In Pytorch-BigGraph, we use the $\boxed{\text{translation}}$ operator for generating the TransE model. Unlike TransE's translation, RotatE [117] regards the relation as a rotational degree between heads and tails. It introduces a loss function based on $h \circ r \approx t$ for simulating the relation translation. We use the GraphVite's [155] pre-trained RotatE embeddings in our experiments.

Semantic matching models measure the similarity between entities and relations during the training. DistMult [140] is based on a bilinear scoring function $h^T M_r t$, where M_r is the relation matrix built on top of the entity. ComplEx [124] can be seen as a constrained variant of RESCAL [61] that leverages fewer relation dimensions inside a complex space. The ComplEx score is defined as $Re(h^T diag(r)\bar{t})$. In Pytorch-Biggraph training, we use the $\boxed{\text{diagonal}}$ operator

---

[2]https://archive.org/details/wikibase-wikidatawiki-20210521

Table 5.5: Gold standard datasets for evaluating STI approaches. The ambiguities are based on DAGOBAH-SL scores

Gold standard	#Tables	Avg. #Rows	Avg. #Col	#Entities	Ambiguities (t=1)	Ambiguities (t=0.9)
Limaye	437	37	2	5,143	181 (3.52%)	685 (13.31%)
T2D	762	157	5	18,589	2,322 (12.49%)	8,852 (47.62%)
2T_v2	180	1,080	5	661,297	30,686(4.64%)	86,739(13.11%)
ShortTables	2,237	2	5	4,474	1,422 (31.78%)	1,822 (40.72%)

for generating DistMult embeddings and iterations between $\boxed{\text{complex\_diagonal}}$ and $\boxed{\text{dot}}$ operators for ComplEx embeddings.

Datasets

We evaluate Radar Station on three popular gold standards: T2D[3], Limaye[4], and Tough Tables version 2[5]. The original T2D and Limaye datasets contain some annotation errors that we have corrected. As T2D and Limaye are gold standards based on DBpedia and Radar Station is a Wikidata-based annotation system, we translate the DBPedia entities given in the gold standards into Wikidata entities through the "Wikidata item" hyperlink from Wikipedia pages of DBpedia entities. We manually corrected this translation when it was failing. Since the number of entities in Wikidata is larger than the number of entities in DBpedia [100], the annotation based on Wikidata is also harder with more candidates to disambiguate. We publish the new resulting ground truth on Zenodo (see the supplementary material). ShortTables is a new dataset we built from T2D, in such a manner that each table only contains two rows. The aim of creating such a dataset is to simulate extreme cases where voting strategies lack electors (i.e. row entities) for a correct CTA (resp. CPA) annotation. The provenance of T2D and Limaye is Web tables. We also consider a synthetic dataset named Tough Tables version 2 (2T_2) to evaluate on more data types. We provide the statistics of these gold standard datasets in Table 5.5.

Evaluation

We evaluate Radar Station with these four datasets varying the embeddings and the tolerance threshold. A random selection of the highest scoring candidates is considered as our baseline and noted as the original system name. We show the overall result for t equals to 1, 0.95, and 0.9 based on DAGOBAH-SL scores on four datasets with different embeddings in Table 5.6 and the fine-tuned result based on DAGOBAH-SL, MTab and BBW with Limaye and T2D in Table 5.7.

---

[3]http://Webdatacommons.org/Webtables/goldstandardV2.html
[4]http://Websail-fe.cs.northwestern.edu/TabEL/
[5]https://zenodo.org/record/6211551

Table 5.6: Radar Station evaluation based on DAGOBAH-SL scores. AP: Ambiguity quality, PA: Precision inside ambiguities, GP, Global precision

t	Methods	Limaye			T2D			2T_v2			ShortTables		
		AQ	PA	GP	AQ	PA	GP	AQ	PA	GP	AQ	PA	GP
1	DAGOBAH-SL	0.647	0.168	0.853	0.308	0.053	0.785	0.067	0.023	0.870	0.672	0.194	0.654
	RS + TransE		0.630	0.870		0.294	0.813		0.041	0.871		0.355	0.673
	RS + RotatE		0.636	0.870		0.289	0.812		0.044	0.871		0.363	0.673
	RS + DistMult		0.391	0.861		0.163	0.798		0.034	0.870		0.229	0.658
	RS + ComplEx		0.57	0.869		0.171	0.798		0.036	0.870		0.235	0.659
0.95	DAGOBAH-SL	0.614	0.296	0.853	0.332	0.180	0.785	0.327	0.208	0.870	0.671	0.302	0.654
	RS + TransE		0.528	0.872		0.312	0.815		0.230	0.872		0.414	0.673
	RS + RotatE		0.542	0.873		0.312	0.815		0.235	0.872		0.418	0.674
	RS + DistMult		0.377	0.860		0.230	0.797		0.213	0.870		0.328	0.659
	RS + ComplEx		0.435	0.864		0.233	0.798		0.219	0.870		0.334	0.660
0.9	DAGOBAH-SL	0.653	0.432	0.853	0.336	0.241	0.785	0.500	0.300	0.870	0.714	0.414	0.654
	RS + TransE		0.570	0.872		0.323	0.815		0.313	0.872		0.532	0.684
	RS + RotatE		0.578	0.873		0.322	0.814		0.318	0.872		0.536	0.684
	RS + DistMult		0.475	0.860		0.274	0.797		0.303	0.870		0.466	0.668
	RS + ComplEx		0.494	0.862		0.275	0.798		0.306	0.870		0.471	0.669

5.3.4 Evaluation Settings

We aim to evaluate the performance of Radar Station on the ambiguity lists and how it can influence the global annotations. Thus, we use three indicators including Ambiguity Quality (AQ), Precision inside Ambiguities (PA), and Global Precision (GP). AQ (Equation 5.8) shows the quality of generated ambiguity list after the Ambiguity Detection step, that is, how many ambiguous cells contain a ground truth in its top candidates. It indicates the extreme precision that we could achieve in all ambiguous annotations, which is PA in Equation 5.9. GP (Equation 5.10) is the overall precision in all labelled cells considering annotations generated with or without Radar Station.

$$AP = \frac{\#Correct\ candidates\ in\ the\ candidate\ set\ of\ ambiguities}{\#\ Ambiguities} \tag{5.8}$$

$$PA = \frac{\#\ Correct\ ambiguity\ disambiguations}{\#\ Ambiguities} \tag{5.9}$$

$$GP = \frac{\#Correct\ annotations}{\#Total\ labels} \tag{5.10}$$

We also introduce the Cohen's Kappa [28] coefficient to evaluate the independence of the annotation from different embeddings models. Cohen's Kappa coefficient is used to analyze the consistency of two reviewers' ratings of the category items, commonly used in the comparison between new discrete data and the gold standard to detect whether the new data has a certain

Table 5.7: Gold standard datasets for evaluating STI approaches with RotatE embeddings. AP: Ambiguity quality, PA: Precision inside ambiguities, GP, Global precision

Dataset	System	t	AQ	Original output		Radar Station	
				PA	GP	PA	GP
Limaye	DAGOBAH-SL	0.9	0.653	0.432	0.853	0.578 (+0.146)	0.873 (+0.020)
	MTab	0.83	0.820	0.705	0.857	0.787 (+0.082)	0.875 (+0.018)
	BBW	0.65	0.587	0.359	0.563	0.507 (+0.148)	0.597 (+0.034)
T2D	DAGOBAH-SL	0.95	0.332	0.180	0.785	0.312 (+0.132)	0.815 (+0.030)
	MTab	0.71	0.385	0.295	0.837	0.346 (+0.051)	0.857 (+0.020)
	BBW	0.65	0.263	0.192	0.364	0.253 (+0.061)	0.382 (+0.018)

effect on accuracy. Cohen's Kappa coefficient is formulated as follows:

$$kappa = \frac{P_o - P_e}{1 - P_e} \tag{5.11}$$

where P_o is the proportion of two reviewers who are equal in this decision; P_e is the expected value of making the same decision when two reviewers make independent choices. When $kappa$ equals to 1, it means that the two datasets are the same.

5.3.5   Analysis

Overall result

We first observe from Table 5.6 that all the chosen embeddings contribute to a significant improvement for PA in the ambiguous cases with the chosen tolerance values and GP. We also notice that Radar Station brings more improvements to GP for the Limaye (Max. 0.02), T2D (Max. 0.03), and ShortTables (Max. 0.03) than for 2T_v2 (Max. 0.002). This drop for 2T_v2 is due to the distribution of the scores of the top candidates: i) as we can see, after relaxing the tolerance from 1 to 0.9, AQ for 2T_v2 has dramatically increased in comparison to the other datasets. Hence, there is no clear boundary between top candidates and bad candidates for the 2T_v2 dataset. That leads to a relatively lousy context embedding for the disambiguation. This scoring distribution is impacted by row number with DAGOBAH-SL mechanism, that is, the more rows we have, the more balanced the scoring would be; ii) the other reason is that 2T_v2 is a synthetic dataset generated with types from a KG. Thus, other column-wised semantic similarities are not obvious in this dataset. Hence, we recommend that future synthetic datasets should consider the inclusion of common themes from these tables to simulate other real-world use cases.

We introduce ShortTables for simulating the extreme cases where the very limited number of rows does not allow existing systems to generate correct CTA and CPA annotations. Bad CTA or CPA may propagate the error to the cell annotations. Thus, we expected to have
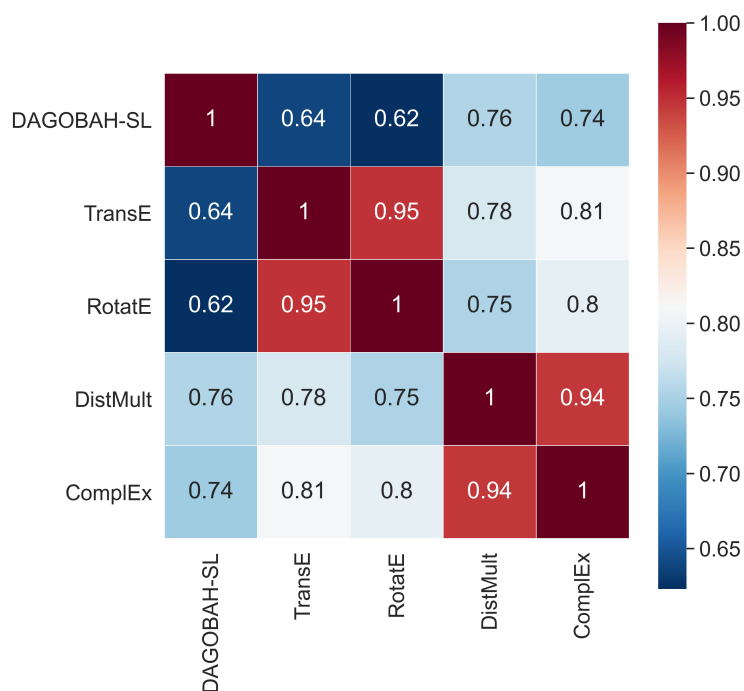
Figure 5.7: Illustration of the Kappa test between different outputs on all datasets, $t = 0.95$.

a more significant GP improvement for ShortTables compared to T2D. However, from our evaluation, the contribution of Radar Station is close in these two datasets (Max 0.03). We analyze that a small number of rows can decrease the quality of type annotation and more likely propagate error with type disambiguation: therefore, it provides more chances for semantic disambiguation. At the same time, the limited number of rows also limits the content of the context entity set that has been used for semantic disambiguation. We argue that these two effects cancel each other in this experiment. We have implemented Radar Station on two other systems and evaluated its performance with two Web table datasets. The result shown in Table 5.7 indicates that Radar Station benefits to all input annotation systems.

Analysis on embeddings

Regarding the two families of embeddings (TransE and RotatE are translational distance models, DistMult and ComplEx are semantic matching models), the GP for embeddings from the same family achieves similar results inside our trained embeddings. From the result of Cohen's kappa shown in Figure 5.7, we observe that the output is similar for embeddings from the same family. For example, the kappa value for TransE and RotatE is much higher than TransE with other outputs (same for DistMult and ComplEx). This similarity could also be seen in the precision shown in Table 5.6. We also observe that translational distance models are generally better than semantic matching models in our trained embeddings. That may be
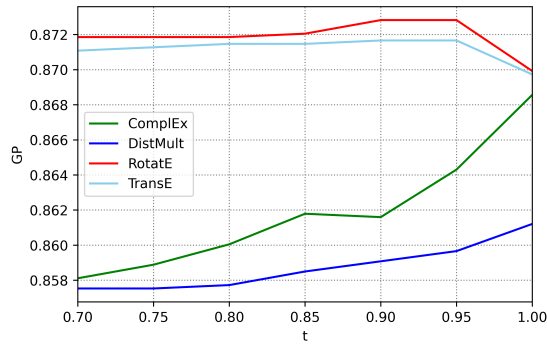
Figure 5.8: The GP evaluation on Limaye with t from 0.7 to 1 based on DAGOBAH-SL.
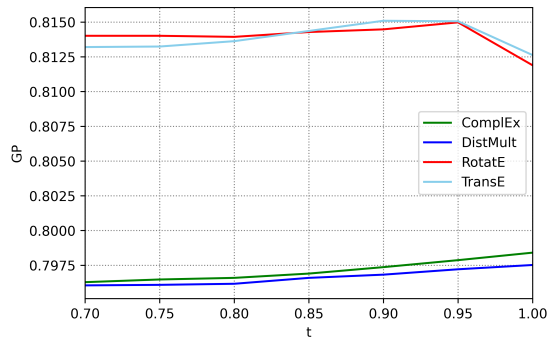

Figure 5.9: The GP evaluation on T2D with t from 0.7 to 1 based on DAGOBAH-SL.

because we leverage geometric distance inside Radar Station, which is compatible with the training strategy of translational distance models. That is, the translational models directly leverage the embedding distance in the latent space as the key of their loss function while the semantic matching models do not. Globally, RotatE embeddings outperform all other models for all datasets.

Tolerance

Relaxing the tolerance has for effect to include more candidate entities and thus has the potential to increase the probability that the correct candidate is in the candidate set. However, such an operation also puts more noise into the candidate list. In Figure 5.8 and 5.9, we illustrate how the tolerance t influences the performance of the system on Limaye and T2D. It shows that relaxing the tolerance with TransE and RotatE improves the quality of the annotation (performance peak at t=0.95 in Figure 5.8). In our observation, largely relaxing the tolerance may decrease the accuracy since more noise is included during the disambiguation. This is therefore a delicate tradeoff to generalize across datasets.

5.4   Discussion

In the chapter, we present the work of the clustering method that performs optimisation on CTA task but it also decrease the performance of CEA task. We also propose Radar Station, a novel approach for addressing the difficult and ambiguous cases leveraging on KG embeddings for uncovering the column wise-similarity. These approaches are evaluated on the SemTab challenge as well as two other popular gold standards. The evaluation shows that we have made a improvement in accuracy and performance based on three others top-tier systems, including DAGOBAH-SL.

However, we also found that they are still some limitations and areas for improvements:

- First, we only use information from the same column as disambiguation support data for Radar Station. But at the same time, the information brought by other columns of the table, headers, titles, and table metadata is not well interpreted for supporting the annotation.

- For the words, sentences, and characters in the table, we treat them as a simple input for mapping with table elements and KG by way of string matching. But the semantic meaning they have in themselves is somewhat ignored despite the KG.

- Our work is still limited to relational tables. Other table types remain uninterpretable.

To address these limitation, in future work, we expect to use language models to better integrate all the information in a table (including its structure and its textual content). We expect first to test existing variants of BERT for tables to explore their feasibility and propose better solutions. Making use of the automatically produced semantic annotations in applications is also an objective of this thesis. We will also focus on the correlation between or inside tables/KGs.

Chapter 6

Applications of Semantic Table Interpretation

The interpretation of tables from the previous sections aims to uncover the meanings of the data with respect to a semantic artifact, such as an ontology or a knowledge graph. One could leverage the extracted knowledge following STI tasks for the benefit of different use cases. Examples of these use cases include search engines [15, 23], question/answering systems [95, 116, 141], knowledge base enrichment [104, 139, 150] or dataset recommendation [148].

In this chapter, we focus on Table Augmentation tasks. The life cycle of tabular data and the coverage of background knowledge represented in open or enterprise KGs can vary. This situation generates differences between tables published by organisations and open or enterprise KGs that are continuously curated: information is missing, dimensions are eluded since they are deemed useless by the data producer, etc. However, in many use cases (e.g., dataset search, profiling and recommendation), the completeness and richness of the data have positive effects and are desirable qualities. Once annotated, tables can be enriched with additional elements from the supporting KG. The purpose of table augmentation is to extend an existing table with additional data. It includes filling the table with new rows and columns or finding missing cell values. These tasks are also known as data imputation and schema augmentation. One of the examples is that structured tabular data is widely used in the field of Machine Learning where the approaches leverage features from different columns to train a predictive model. Those Machine Learning models are mostly data-driven where the performance of the trained model always depends on whether the training datasets are clean or not. Augmenting a given table with new columns or updated cell values provides additional features to improve the quality of the input data.

Table augmentation aims to extend a given table using other sources as background knowledge. The background knowledge could come from other tables, a knowledge graph, or textual information from the training data of a given language model. Based on our observation from the literature, three different subtasks of table augmentation have been introduced so far, namely row population [107, 146], data imputation [139, 147], and schema augmentation [15,

139]. We prioritised data imputation and schema augmentation tasks because they were more important for our use cases in the industry.

Data imputation, also known as data completion and missing cell filling, refers to correcting or filling an empty cell of a given table. Figure 6.1.a illustrates an example of data completion where we fill the missing cell from the first row and second column with a specific year number (2021). Schema augmentation, known as column extension, extends a table with additional columns. [149] argues that this task roughly corresponds to the join operation in databases. Figure 6.1.b depicts schema augmentation of a table about movies by extending a column of movie directors.

(a)

Title	Year	Shooting place
The French Dispatch		Angoulême
Les Beaux Gosses	2009	Rennes
Nos Vies Heureuses	1999	Belfort

⇩

Title	Year	Shooting place
The French Dispatch	2021	Angoulême
Les Beaux Gosses	2009	Rennes
Nos Vies Heureuses	1999	Belfort

(b)

Title	Year	Shooting place
The French Dispatch	2021	Angoulême
Les Beaux Gosses	2009	Rennes
Nos Vies Heureuses	1999	Belfort

⇩

Title	Year	Shooting place	Director
The French Dispatch	2021	Angoulême	Wes Anderson
Les Beaux Gosses	2009	Rennes	Riad Sattouf
Nos Vies Heureuses	1999	Belfort	Jacques Maillot

Figure 6.1: Illustration of data imputation tasks: a) data imputation; b) schema augmentation.

Aiming at this target, in DAGOBAH UI [108], we introduce two semi-automatic system prototypes for table augmentation. DAGOBAH UI provides a list of candidates with entities (resp. relations) aiming at enriching an empty cell (resp. adding a column). Users can select their target elements to add from this candidate list. In this chapter, we first introduce the related work of table augmentation in Section 6.1. In Section 6.2, we illustrate the application of data imputation and in Section 6.3, we introduce our efforts around schema augmentation. We finally conclude and discuss in Section 6.4.

6.1 Related Work

DATA imputation fills an empty table cell. [5] proposed a hybrid model between a heuristic lookup-based system and a machine learning system to predict a missing cell of a relational

table. The system's training dataset is collected from Web tables. [147] considers both KG and used table corpora as the sourced data and proposes a heuristic system for combining the evidence from a different source when predicting an empty cell. They also accept multi-value output and enable the system to have a "None" output for the cases where the cell should be empty. TURL [38] leverages language models. It follows the same encoding steps as presented in Section 3.5.3. Where the authors argue that data imputation is similar to the NER task. Hence the system directly uses "[MASK]" to generate the missing cell value.

Schema augmentation corresponds to extending a given table with new columns. One possible way to achieve the goal leverages the given table corpus as the background. The Mannheim Search Joins Engine [69] operates by searching for tabular data describing entities in the local table, then picks relevant columns from the top-k candidate tables to merge. [15] targets column-matched tables with the local table and performs correlation mining to numeric columns of given Wikipedia tables. InfoGather [139] is a table augmentation framework based on PageRank for matching the local table against web tables. The context surrounding the tables is leveraged in a machine learning framework, where the similarity between two tables is captured via a set of features. The optimized version InfoGather+ [145] focuses on the number values. [146] proposes a Bayes-based system that outputs a ranked list for column labels based on a given table corpus. Like what has been applied in data imputation, TURL [38] uses "[MASK]" token over the headers to predict the label of the potential headers.

In DAGOBAH UI, we enrich the table with new columns and cell values relying on the processed table annotations generated with DAGOBAH SL. KG annotations tell the system about the entities, types, and relations of the input table. The system can understand the context of the target table with these annotations and enrich the table according to the background knowledge from the KG. DAGOBAH UI differs from the existing works by: 1) instead of automatically providing values to an empty cell, we adopted a semi-automatic approach enabling the user to validate suggestions according to his own needs; 2) we leverage KG as our background knowledge source.

6.2 Data Imputation

In this section, we introduce data imputation module of DAGOBAH UI. Where the system is described in Section 6.2.1, then we report and analyse the result in Section 6.2.2.

6.2.1 System

In DAGOBAH UI, we apply the data imputation task following the annotation steps of DAGOBAH SL. When a cell is empty, the user could choose whether or not to fill this empty cell. For data imputation, we observe that the majority of approaches propose a unique value for these

empty cells. However, in some cases, one might have multiple correct candidates at the same time. For example, in the table of Figure 6.1, supposing the missing value is the cell "The French Dispatch" in the first column, the suitable movies for this cell could be many. For example, the movie "Happening" ("Q107675028") directed by Audrey Diwan is also filmed in Angouleme in 2021 and is suitable for this cell. Listing all suitable candidates is sometimes not that adequate with the table format since all other cells only depict one movie. Hence, we do not aim to provide a unique result for the imputation but generate a list for the user to choose from.

DAGOBAH UI uses SPARQL queries to generate the candidate list as illustrated in Listing 6.1, where "annotated-type" is the annotated type for the target column, "annotated-relation" is the generated relation annotation for the target empty cell, and "annotated-neighbor" is the neighboring annotated entity associated with the relation.

```
SELECT ?candidate ?candidateLabel
WHERE
{
  ?candidate wdt:P31 annotated-type.
  ?candidate annotated-relation1 annotated-neighbor1.
  ........
  ?candidate annotated-relation2 annotated-neighbor2.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE]". }
}
```

Code Listing 6.1: Data imputation query

Figure 6.2 illustrates a data imputation example with DAGOBAH UI. The original table describes the companies and their industry domains, where we do not know the industry domain of the company "Adobe Systems". DAGOBAH UI provides 3 candidates (all of them are correct) for this cell and the user could select one of them or keep all of these candidates.

6.2.2   Result and Analysis

We have manually extracted 100 tables from the T2D corpus as the dataset for evaluation. For each table, we mask one cell as the target empty cell for data imputation, where half of the empty cells are located in subject columns (introduced in Section 2.1.2) and the rest are located in attribute columns that describe the attributes of subjects. We let DAGOBAH UI generate a candidate entity list for each empty cell and manually evaluate the generated list according to the following metrics.

Let K equals the number of empty cells, we introduce the precision indicator and the quality indicator for the evaluation with the following functions:

Figure 6.2: Example of data imputation with DAGOBAH UI with table 52572391_0_8684896999787304275.csv from T2D dataset.

$$Precision = \frac{\# \, List \, contains \, correct \, entities}{K} \tag{6.1}$$

$$Quality = \frac{1}{K} \sum_{j < K} \left(\frac{\# \, correct \, entities \, in \, list_j}{length(list_j)} \right) \tag{6.2}$$

Equation 6.1 reveals the percentage of the generated lists having the correct entities. Equation 6.1 indicates the quality of the generated lists.

We report the result based on the precision indicator and the quality indicator in Figure 6.3, where "A_columns" depicts the attribute columns and "S_columns" depicts the subject columns. We also show the distribution of the generated list length in Figure 6.4.

For the overall result, we have successfully generated a candidate list for 64% of the empty cells, where 22 subject columns and 14 attribute columns failed due to the misleading pre-processing negative results and annotation errors. We notice that in our sample set, once the system generates a candidate list for attribute columns, it always contains at least one correct candidate. However, we did not find the correct entities in the candidate list of four subject columns. That gives 60% of the precision in all tables at the end, also 72% precision in
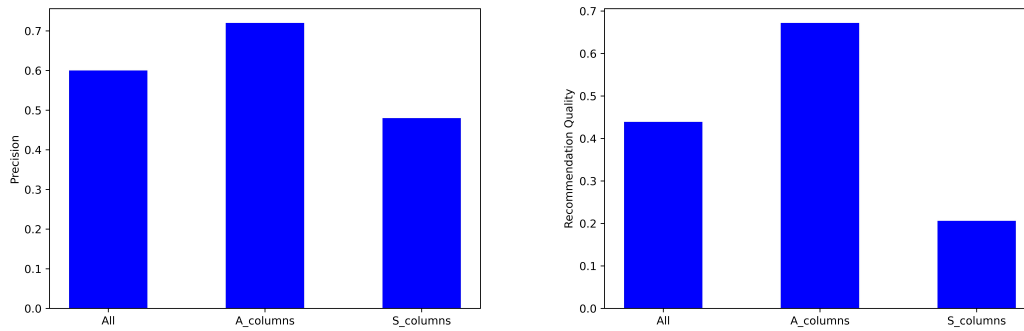
Figure 6.3: The evaluation of data imputation in 100 selected T2D tables in all columns, attribute columns, and subject columns: a) precision indicator; b) quality indicator
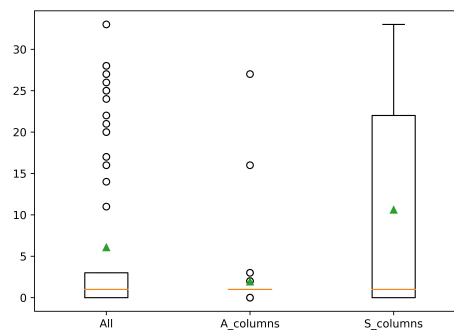
Figure 6.4: The list length distribution of data imputation in 100 selected T2D tables in all columns, attribute columns, and subject columns

attribute columns and 48% precision in subject columns as shown in Figure 6.3.a.

This overall result shows that our system performs better in attribute columns rather than in subject columns. Similar cases could be found in other evaluations such as the quality of the list for attribute columns is much higher than subject columns (Figure 6.3.b), and the candidate list's length for attribute columns is much shorter than subject columns (Figure 6.4). We argue that it is because we currently do not use the literal cells for the disambiguation and the filtering of the candidate list. Literal values in a table are often used for describing the subject. Lacking this information weakens the performance of the system on subject columns, however, it does not affect a lot on attribute columns since it only needs the subject and the relation to filter candidates.

6.3 Schema Augmentation

In this section, we introduce the schema augmentation module of DAGOBAH UI. The system is described in Section 6.3.1, then we report and analyse the results in Section 6.3.2.

6.3.1 System

In DAGOBAH UI, we add additional columns based on a fixed target entity column. The user could select this target column from the interface. After the interpretation of the target table, every cell from this column is associated with an entity from the target KG. Each annotated cell entity has multiple relations in this background KG. For each cell entity relation, we count the number (or occurrence) of this relation in the target column and rank these relations according to their frequency in the column to generate a list of candidate relationships. We show this list to the user in the interface and the user can select one of the relations that he/she wants to add to the table. Once selected, the system extends the table. For each row, we know the subject (which is the cell annotation from subject column) and the predicate (which is the relation selected by the user), hence, the task is to find the missing object of the triple (which is composed by $< subject, predicate, object >$).

One example is shown in Figure 6.5. We suppose that a user wants to add the platforms for each game presented in the annotated table from Figure 6.5.a. The user will have a list of all the potential elements as shown in Figure 6.5.b to add according to the first column. Once he chooses the term "platform" from the list to achieve his goal, DAGOGAH UI automatically adds the column as the second column from as illustrated in Figure 6.5.c.

6.3.2 Result and Analysis

Since DAGOBAH UI is a semi-automatic system, its performance depends on the quality of table annotations. We argue that the schema augmentation performance of DAGOBAH UI relies on the quality of the target KG. Hence one could switch their target KG to improve the coverage of the relations and cell values. To the best of our knowledge, we are the first system that allows human intervention during the selection of added columns, yet we still lack suitable criteria for evaluating the performance of the system.

To give a minimal indication of the system's capabilities, we manually count the possible relation number with the 100 tables used in the evaluation of data imputation module. Where we only focus on the subject columns since the additional columns in a table are always used for describing the subjects. The annotation is based on DAGOBAH SL with the Wikidata dumps extracted in 2022.

Figure 6.5: Schema augmentation of a table of games with DAGOBAH UI: a) the initial table with the annotation generated with DAGOBAH SL; b) the menu for users to choose one candidate relation based the first column; c) the extended table with a new column about "platform".

We report the number of the relations in the candidate set that we provide to the user in Figure 6.6 with a box figure. The relation in our example is in an interval between 8 and 108 with an average of 37.8. Most of the relation numbers for subject columns are between 20 and 40. Evidently, it is not easy for a user to choose a relation within a large list like this. Hence, in future work, we aim to study how to rank the candidate relations to optimise the user experience.

Using the example of Figure 6.5, the following aspect could be used for ranking the candidate columns.

- **Row coverage**: The entity coverage could be an important indicator for ranking the columns. Choosing "platform" (P400) as the added column covers about 94% of the
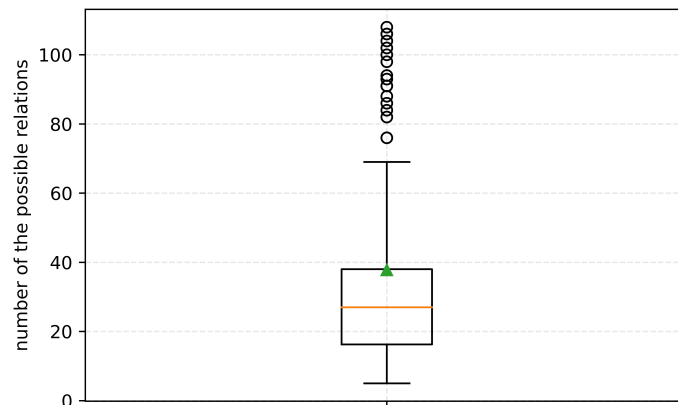
Figure 6.6: The list length distribution of data imputation in 100 selected T2D tables in all columns, attribute columns, and subject columns

rows from the original table, while choosing "CERO rating"(P853) only covers 8% of the rows. In such cases, the relation that covers more rows should be prioritised.

- **Relation Popularity**: A relation with many associated entities in a KG is more likely to be used in a given case. For example, in Wikidata, we have more triples about the country of origin (P495) than CERO rating (P853). The user is therefore more likely to choose the country of origin.

- **Relatedness of the context**: In some cases, we can find additional information from the metadata, such as the surrounding text or page title. If the table is located in a paragraph that discusses the evolution of video games in recent decades, it might be helpful to prioritise "publication date" (P577) as the recommendation of the additional column since it is associated with the topic of the paragraph.

- **Co-occurrence with other columns**: If some columns appear in one different table, then it is more likely to have a similar column combination for the target table. The column wised co-occurrence has been used for STI [129] and schema augmentation [38]. While TURL [38] trains the model based on the existing table dataset and adds the column according to this dataset. We aim to provide a list for the user to choose from, hence, we did not adopt this at first. However, the columns wised co-occurrence could be an aspect to rank the relations that we aim to optimise in the future.

6.4 Discussions

In this chapter, we introduce the functionalities of table augmentation in DAGOBAH UI. DAGOBAH UI fills missing cell values and extends table columns relying on STI annotations generated by DAGOBAH SL. Unlike other table augmentation tools that directly output the result from the back-end without human intervention, DAGOBAH UI provides a very user-friendly interface to manipulate the table, and the users can select their own proper way to customise their own table.

Our future work includes the development of new features around KG enrichment from tables. As discussed in this chapter, tables can benefit from KGs through data imputation and schema augmentation with new columns. Conversely, tables are also a great source of dormant knowledge that can be leveraged to enrich open or enterprise KGs. To this aim, DAGOBAH UI will enable to export the annotations as Resource Description Framework (RDF) triples to enrich KGs. At the same time, we aim to optimise the system by using the literal values from the table. Currently, we only make use of the entity columns from the annotation results, however, the numbers, units, and long sentences from the table also help to precise the enrichment. Furthermore, our future work also includes the evaluation of DAGOBAH UI usability with real users. To this aim, the availability of DAGOBAH UI within the company will allow us to collect interesting feedbacks to progress on the adoption of STI tools.

Chapter 7

Conclusion and Future Work

Tabular data is a valuable data source that is available on the Web and in companies. Understanding and making use of the information from tabular data is an open-research question that has been increasingly attracting attention from the scientific community over the last decades. Such process includes the operations like collection, extraction, interpretation, and augmentation of tables. However, tabular data is challenging to process by machines because of the limited context available to resolve semantic ambiguities and the layout of tables that can be difficult to handle.

In this thesis, we provide contributions from collecting tables in the wild to downstream tasks. This dissertation covers various topics, with a special focus on the interpretation of tables with the help of the intersection between tables and KGs. We interpret and enrich tables using the overlapped information between these two data sources.

In the following, we summarise the main contributions of this thesis on extracting, interpreting, and augmenting tables in Section 7.1. We also highlight the limitations of this work in Section 7.2 and suggest some perspective about future works on this research topic.

7.1 Contributions

This thesis focuses on making use of tabular data with the help of the interpretation with a given KG. To reach this target, we set a list of research questions in Section 1.2 for this thesis. These research questions are recalled briefly below:

1. What are the different table types and features that one can encounter on the Web and how can we extract them?

2. What possible correspondences can we establish between the elements from tables and knowledge graphs?

(a) Can the overlapped information from these two data sources support the matching elements between the two sides?

(b) Can these latent relationships between the table elements also reveal the common topics from these tables to improve the disambiguation with the help graph embeddings?

3. Finally, what downstream tasks could be realized following a semantic table interpretation process?

To answer these research questions, a set of concepts and tools are proposed. In the following, we recall the contributions of the thesis from three aspects: the extraction of tables (Section 7.1.1), the interpretation of tables (Section 7.1.2), and the augmentation of tables (Section 7.1.3).

7.1.1 Table Extraction from the Wild

Tabular data has different layout structures and different purposes of creation. Facing this heterogeneity, a classification of tables is necessary. Based on existing categorisation, we further refresh the classification of tables from the perspective of table interpretation in which we propose to classify a given table using three dimensions, including structure, interrelationship and orientation dimension. This work has been published in a survey paper [74].

We also contribute with two systems (**HTW** and **CorpusWalker**) for collecting tables from the Web and local documents. These systems provide features for ingesting, managing, storing and visualising the collected tables and provide the data sources for table interpretation.

Table extraction is concerned with the problem of identifying and classifying tables, which encompasses a range of more specific tasks, such as relational table classification, header detection, and orientation classification [149]. Focusing on relational tables, we propose a table pre-processing tool named **DAGOBAH-Pre-processing** [24]. DAGOBAH pre-processing generates metadata about a table via four main tasks: orientation detection, header detection, key column detection and column primitive typing.

7.1.2 Semantic Table Interpretation

The main idea for making tabular data intelligently processable by machines is to find correspondences between the elements composing the table and entities, concepts, or relations described in KG, known as STI. One of our contributions is to provide a complete, comprehensive and up-to-date state-of-the-art of the different tasks and methods that have been proposed so far to perform STI [74]. We define five major sub-tasks that STI deals with even

if the literature has mostly focused on three sub-tasks so far. We review and group the many approaches that have been proposed into three macro families and we discuss their performance and limitations with respect to the various datasets and benchmarks proposed by the community. To the best of our knowledge, such a survey is still missing in the community and we aim to highlight the potential problems that one can encounter and that yield new challenges.

We propose a continually optimised, heuristic-based, STI annotation system, named **DAGOBAH SL** [58, 59]. DAGOBAH SL automatically interprets relational tables with the support of Wikidata and DBpedia by associating each cell with an entity (CEA), each column with a class (CTA), and each pair of columns with a property (CPA). DAGOBAH SL is one of the state-of-the-art systems and it won the first prize on the accuracy track in both SemTab 2021 and SemTab 2022 challenges. The system is available via an API for developers as well as via a user-friendly web interface [108].

However, heuristic-based approaches like DAGOBAH SL suffer from some shortcomings. One of them is that heuristic-based systems often lack the knowledge of the common theme for a given table. To handle this issue, we propose two graph embeddings based methods, named **DAGOBAH Embeddings** [24] and **Radar Station** [75]. We rely on the intuition that the entities in the same column should exhibit semantic similarities and we aim to use graph embeddings to reveal these similarities. DAGOBAH Embeddings uses a clustering algorithm to the candidate entities for filtering candidates by choosing the right clusters. Experimental results prove that this approach has successfully improved the accuracy of the CTA task. However, the system is also misled by incorrect candidates in the selected clusters during the CEA task. Radar Station is a hybrid plugin system that aims to add a semantic disambiguation step after a previously identified CEA. RadarStation has been evaluated on top of different heuristics-based systems (DAGOBAH SL, BBW, MTab) and have consistently demonstrated an accuracy improvement of around 3%. Furthermore, the system shows empirical evidences that among the various graph embeddings families, the ones relying on fine-tuned translation distance have superior performance compared to other models.

7.1.3 Table Augmentation

Table augmentation refers to the task of extending an existing table with additional data. In DAGOBAH UI [108], we provide semi-automatic table augmentation features including data imputation and schema augmentation, where we aim to find missing cell values and populate a table with new columns. For each task, DAGOBAH UI provides a list of candidate values (resp. types) for each missing-value cell (resp. potential columns) and lets the user choose their appropriate data. The system is driven by the annotation generated by DAGOBAH SL and enriches the table based on the information of the mapped KG.

7.2 Future Work

While our efforts have made significant progress in the field of STI, existing approaches have several limitations: (i) they mainly focus on single-cell subjects within relational or entity tables, and make strong assumptions about the coherence and the simplicity of their layout; (ii) they are highly confident in both the completeness and the correctness of the target KG; (iii) they only partially leverage the information of the table, in both substance and form. iv) they discuss rarely the use case of the annotation generated by STI systems. Based on these observations, we formulate some possible guidelines to sketch the directions for our future research.

7.2.1 Beyond Simple Table Type

From the literature, we observe that most of the works focus on single-cell subject tables, the simplest type of relational tables. A few approaches focus on entity tables such as the infoboxes from Wikipedia pages [17, 136] but the other types of tables defined in the classification we proposed in Section 2.1 are still hardly handled. Moreover, current approaches do not dig deeper into relational table complexities such as hidden subjects or composed subjects, to name a few.

As a result, existing systems are far from being generalisable to any table type. To fill the gap and stimulate the search for new solutions, we believe it is important to broaden the spectrum of corpus complexities. To that end, we aim to create new datasets and more robust annotation systems with multiple table structures and complex contents to tackle the whole diversity of real-world data. Regarding the future datasets, we argue that content-level complexity should not be restricted to noise added to mentions, whether synthetically or manually, as these artefacts happened to be not so difficult to handle in the light of the SemTab experience, as well as not so close to real data tables. Introducing numerical mentions with heterogeneous units or lists within cells (multivalued tables), for instance, could be more challenging and therefore beneficial for the community. Last but not least, a ground truth shall be associated with these datasets to allow a fair comparison between the future approaches. This latter requirement suggests prioritising quality over quantity for evaluation datasets to bootstrap new challenges quickly.

7.2.2 KG Incompleteness and Incorrectness

Our existing approaches assume that the target KG is complete and error-free. As a consequence, an annotation can always be generated even if the correct result is not in the KG, whether it concerns an instance, a type or a relation. This situation can be harmful, especially as it may spread the error from one annotation to the whole column or even the whole table.

Suppose for instance a table where a column contains the last names of writers (which can be very common), and another column related to books' titles (for the sake of the example, we assume the majority of these books have been adapted for the cinema). If the target KG covers extensively movies but only a few literature works (or is less accurate on books than movies), the annotation process might lead to type the second column as "film", which could lead to wrongly disambiguate the mentions in the first column (if some related actors have similar last names for example). As a result, this table will be interpreted as an actors-movies item instead of the correct writers-books target.

Some existing mechanisms, such as giving a confidence score for each candidate, can help filtering the incorrect annotations further. The T2Dv2 benchmark, for instance, added negative examples that can be leveraged to that end. However, rare studies focus on this challenge which is far from being trivial as it implies to have the capability to identify the KG coverage w.r.t. the tables to be processed, as well as to detect the possible errors. To improve both the completeness and the correctness, we believe that leveraging multiple KGs is the first step to make. Indeed, it could enhance the coverage and provide a basis for confidence scoring through popularity computation. However, evaluation procedures should be discussed and updated as judging from different sources might be challenging. Finally, we highlight that our future approaches will also consider to tackle domains where only nascent KGs exist, with the objective of using STI to augment these KGs in a virtuous iterative loop.

In Table 3.3, many annotation systems (e.g., MTab,DAGOBAH SL) achieve very high performances on some synthetic datasets (e.g., SemTab 2020, SemTab 2021 R2-Hard). This can be explained since synthetic tables are automatically and accurately generated using a reference KG (see Section 3.4). Therefore, their content is almost fully represented in the KG and provides rich discriminative information for the disambiguation. In contrast, manually curated tables (R3-BiodivTable), complex tables (ToughTable), or tables coming from diverse and specific domains (R3-GitTables) are still particularly challenging for annotation systems. In real situations, KGs are often incomplete. As a consequence, an existing entity may not be fully described in a KG (e.g., lack of literal attributes for a given entity), or an unpopular/heterogeneous domain may provide little information (e.g., R3-GitTables is made of tables from GitHub). Hence, the graph context provided by the KG can sometimes be insufficient for disambiguating tables in R3-BiodivTable, ToughTable, and R3-GitTables, which are much more ambiguous than synthetic tables. We aim to optimise the current systems that could enrich and better handle both explicit and implicit contextual information by exploiting knowledge graph reasoning or table representation learning (e.g., Transformer) to improve the performance on these kinds of table datasets.

7.2.3 Table Context

We observe that many approaches leverage only partially the elements of the table (see Table 3.2), even if more recent ones tend to extend their view. As we discussed in Section 3.6, we believe that leveraging as many elements as possible should increase the accuracy by adding more contextual information. In that sense, language models generated from transformers could be better used. Indeed, one could consider a table as a way of structuring the language: in the simplest case, one table row can be seen as a sentence describing a subject with some attributes. The same applies to the corresponding sub-graph in the target KG. Thus, sentence representation could be used to compute similarities. Nonetheless, the specificity of tabular data as well as KGs should be taken into account, which implies adapting attention mechanisms to this very structure. The visibility matrix used in [38] is an attempt to do so in relational tables, but it should be extended to other types of tabular data.

We also notice that most approaches treat tables independently. However, some tables are related to each other since they can be generated with the same template, be part of a coherent corpus of tables or be related to keys such as SQL database tables. Inter-table relations as studied by [129] can constitute an interesting complementary approach with appropriate target tables. We believe that STI systems could significantly take advantage of combining table elements with inter-table connections (which can be considered as another context layer added to capture richer prior information about the data to be processed) in our future work.

7.2.4 Applications of Semantic Table Interpretation

In the existing work, we leverage generated semantic annotations in data imputation and schema augmentation in the DAGOBAH UI system. CorpusWalker also provides the possibility for researching a document with a given id from the KG. However, we aim to study more possible use cases of the semantic table annotations.

We will also focus on the correlation between or inside tables/KGs. We will explore whether we can use this relatedness to support a downstream task such as indexing and recommendation through our existing work. For example, a recommendation process or the generation of summaries. In the context of data exchange, a recommendation engine provides advice to users on which datasets are likely to be of interest to them and to serve their use cases. As a first approach, content-based recommendations (distance measure between a vector representing the user and vectors representing the datasets) and exploiting an enriched description of the labelled datasets can be considered. Vector representation of items taking into account heterogeneous information coming from a graph (textual descriptions, structural knowledge, etc.) requires the use of appropriate techniques. The so-called "graph folding" techniques seem particularly suitable to achieve this objective of constructing latent vectors from triples.

Publications List

The research carried out during this reporting period has lead to the publication of the following scientific papers:

Journal

- Jixiong Liu, Yoan Chabot, Rapaël Troncy, Viet-Phi Huynh, Thomas Labbé, Pierre Monnin. **From Tabular Data to Knowledge Graphs: A Survey of Semantic Table Interpretation Tasks and Methods**. In Journal of Web Semantics, 2023.

International Conference and Workshop

- Jixiong Liu, Viet-Phi Huynh, Yoan Chabot, Raphaël Troncy. **Radar Station: Using KG Embeddings for Semantic Table Interpretation and Entity Disambiguation**. In 21st International Semantic Web Conference (ISWC), Research Track, 2022.

- Viet-Phi Huynh, Yoan Chabot, Thomas Labbé, Jixiong Liu, and Raphaël Troncy. **From Heuristics to Language Models: A Journey Through the Universe of Semantic Table Interpretation with DAGOBAH**. In Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2022. **First prize on accuracy track**.

- Viet-Phi Huynh*, Jixiong Liu*, Yoan Chabot, Frédéric Deuzé, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. **DAGOBAH: Table and Graph Contexts for Efficient Semantic Annotation of Tabular Data**. In Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2021. **First prize on accuracy track**.

- Yoan Chabot, Pierre Monnin, Frédéric Deuzé, Viet-Phi Huynh, Thomas Labbé, Jixiong Liu, and Raphaël Troncy. **A Framework for Automatically Interpreting Tabular Data at Orange**. In 20th International Semantic Web Conference (ISWC), Industry Track, 2021.

- Viet-Phi Huynh*, Jixiong Liu*, Yoan Chabot, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. **DAGOBAH: Enhanced Scoring Algorithms for Scalable Annotations of Tabular**

Data. In Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2020. Third prize.

- Yoan Chabot, Thomas Labbé, Jixiong Liu, Raphaël Troncy. **DAGOBAH: an end-to-end context-free tabular data semantic annotation system** In Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2019.

Poster and Demo

- Christophe Sarthou-Camy, Guillaume Jourdain, Yoan Chabot, Pierre Monni, Fréedéric Deuzé, Viet-Phi Huynh, Jixiong Liu, Thomas Labbé, Raphaël Troncy. **DAGOBAH UI: A New Hope For Semantic Table Interpretation**. In European Semantic Web Conference (ESWC), Demo Track, 2022.

National Conference and Workshop

- Viet-Phi Huynh, Jixiong Liu, Yoan Chabot, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. **DAGOBAH: Annotation sémantique de données tabulaires par comparaison du contexte des tables et d'un graphe de connaissances**. In French Conference on Knowledge Engineering (IC), 2022. **Best Paper Award**.

- Yoan Chabot, Thomas Labbé, Jixiong Liu, Raphaël Troncy. **DAGOBAH : Un système d'annotation sémantique de données tabulaires indépendant du contexte**. In French Conference on Knowledge Engineering (IC), 2020.
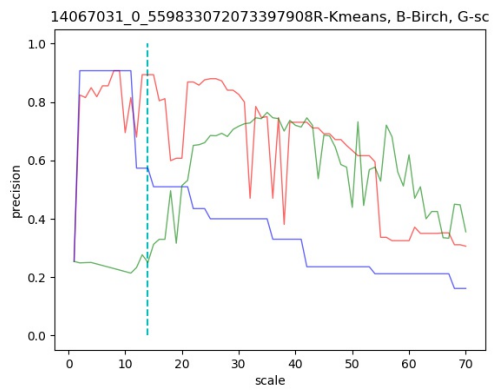
Chapter A

Clustering Sample Evaluation on 15 Selected T2D Tables

DAGOBAH Embeddings (Section 5.2) hypothesizes that all entities in the same column of the table should be close to each other in the embedding vector space. Consequently, the correct candidates are assumed to belong to a few clusters. During the development of DAGOBAH Embeddings, the choose of a suitable clustering algorithm and a suitable cluster number is always a crucial task. In this appendix, we introduce the empirical sample test results with different clustering algorithms and different cluster numbers. To this end, we launched a grid-search procedure on 15 sampled tables listed in this appendix. We have tested three clustering algorithms that can configure the number of clusters from the hyper-parameters, including K-means, BIRCH, and Spectral Clustering. We also introduce a new indicator p where p is equal to the number of look-ups divided by twice the number of rows of the table to be annotated. We had a grid-search on all integers between 2 and $5 * p$ as tested K values, evaluating with the precision of the CEA annotation result generated with DAGOBAH-Embeddings system. In Figure A.1, A.2, A.3, and A.4, we use the red color for the result of KMeans clustering, blue color for the BIRCH, and green for Spectral Clustering. We also illustrate the p value with a dotted line in each table.
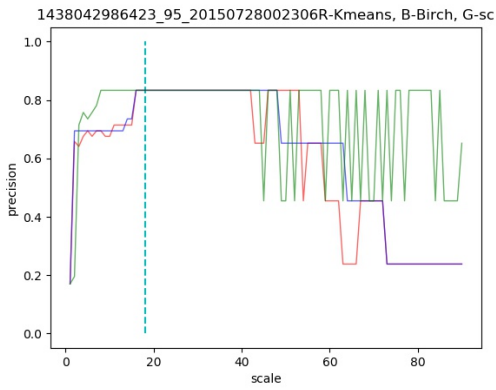
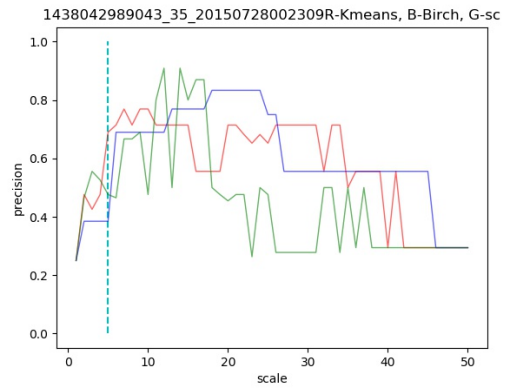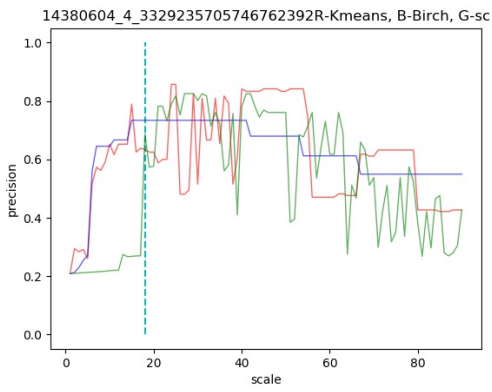(a) Table 11833461_1_3811022039809817402   (b) Table 14067031_0_559833072073397908

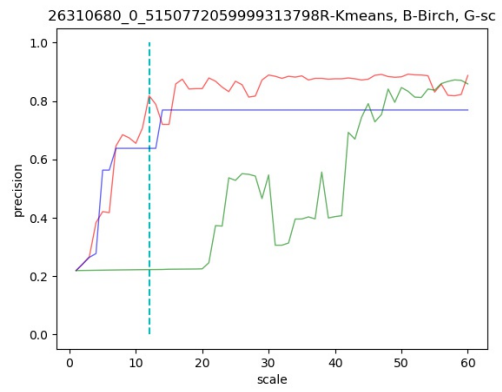Figure A.1: Results of the clustering evaluation for table 1-2
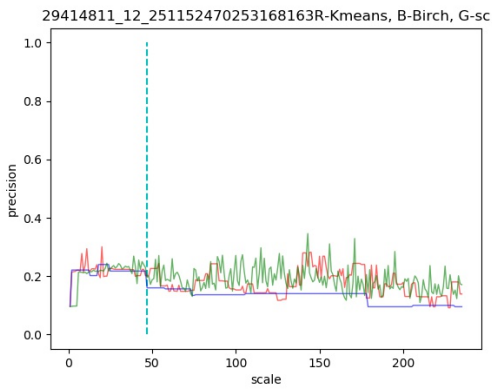
(a) Table 1438042986423_95_20150728002306

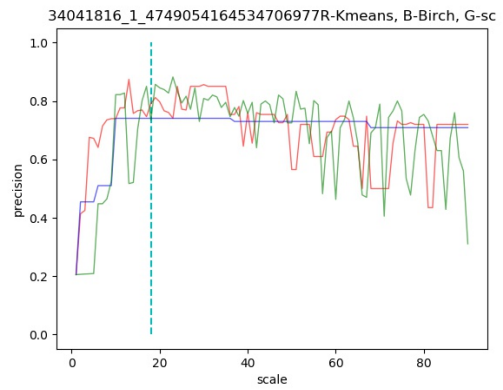(b) Table 1438042989043_35_20150728002309

(c) Table 14380604_4_3329235705746762392
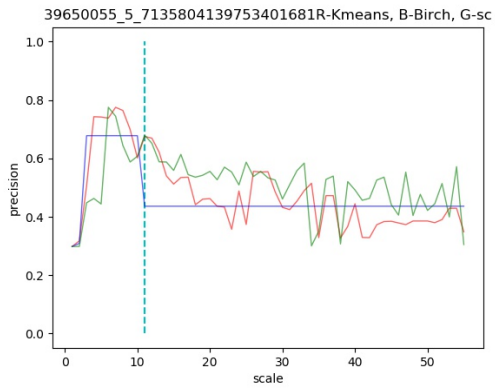
(d) Table 26310680_0_5150772059999313798
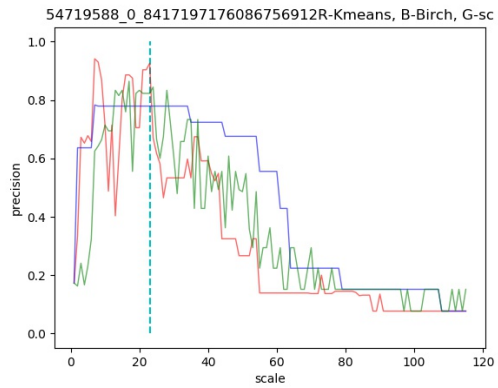
(e) Table 29414811_12_251152470253168163

(f) Table 34041816_1_4749054164534706977

Figure A.2: Results of the clustering evaluation for table 3-8
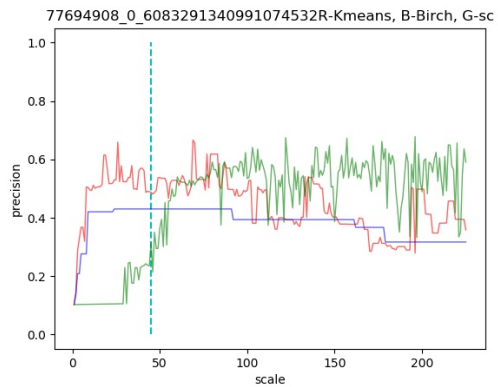
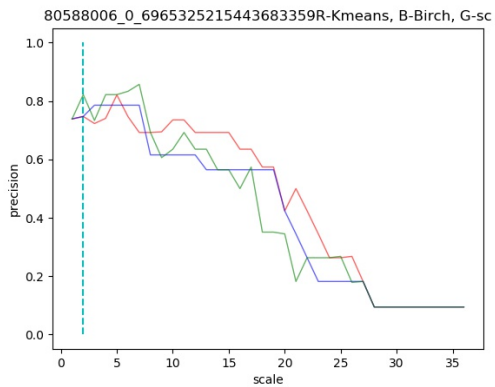(a) Table 39650055_5_7135804139753401681
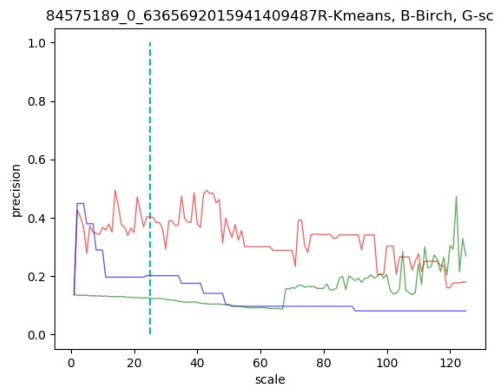
(b) Table 54719588_0_8417197176086756912
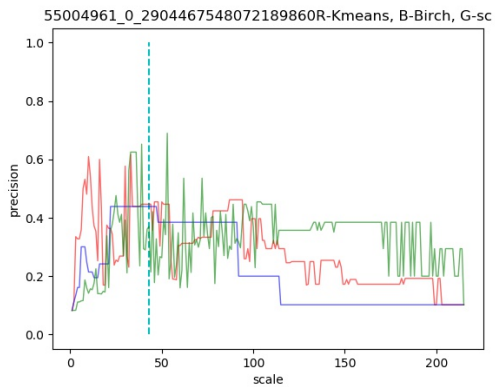
(c) Table 55004961_0_2904467548072189860

(d) Table 77694908_0_6083291340991074532

(e) Table 80588006_0_6965325215443683359

(f) Table 84575189_0_6365692015941409487

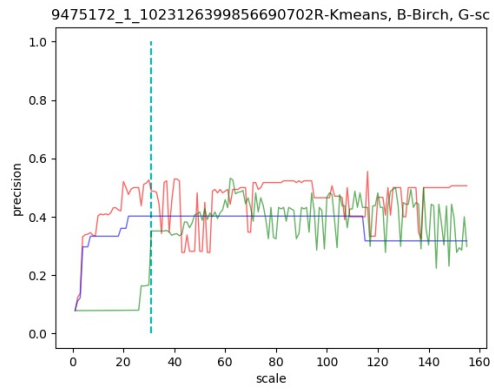Figure A.3: Results of the clustering evaluation for table 9-16

Figure A.4: Results of the clustering evaluation for table 17

Bibliography

[1] Nora Abdelmageed and Sirko Schindler. JenTab: Matching Tabular Data to Knowledge Graphs. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, pages 40–49, 2020.

[2] Nora Abdelmageed and Sirko Schindler. JenTab: A Toolkit for Semantic Table Annotations. In *2nd International Workshop on Knowledge Graph Construction*, 2021.

[3] Nora Abdelmageed and Sirko Schindler. JenTab Meets SemTab 2021's New Challenges. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEURWS. org*, 2021.

[4] Nora Abdelmageed, Sirko Schindler, and Birgitta König-Ries. BiodivTab: A Tabular Benchmark based on Biodiversity Research Data. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2021.

[5] Ahmad Ahmadov, Maik Thiele, Julian Eberius, Wolfgang Lehner, and Robert Wrembel. Towards a hybrid imputation approach using web tables. In *International Symposium on Big Data Computing*, pages 21–30. IEEE, 2015.

[6] Ahmad Alobaid and Oscar Corcho. Balancing coverage and specificity for semantic labelling of subject columns. *Knowledge-Based Systems*, page 108092, 2022.

[7] Rabia Azzi and Gayo Diallo. AMALGAM: making tabular dataset explicit with knowledge graph. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, pages 9–16, 2020.

[8] Wiem Baazouzi, Marouen Kachroudi, and Sami Faiz. KEPLER-asi at SemTab 2021. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2021.

[9] Wiem Baazouzi, Marouen Kachroudi, and Sami Faiz. Yet Another Milestone for Kepler-aSI at SemTab 2022. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2022.

[10] Sreeram Balakrishnan, Alon Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. Applying webtables in practice. In *In 7th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2015.

[11] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *ACL-08: HLT*, pages 28–36, 2008.

[12] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, 41(5):706–716, 2008.

[13] Omar Benjelloun, Shiyu Chen, and Natasha Noy. Google Dataset Search by the Numbers. In *International Semantic Web Conference (ISWC), In-Use Track*, 2020.

[14] Tim Berners-Lee. Linked Data - Design Issues. http://www.w3.org/DesignIssues/LinkedData.html, 2006.

[15] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on wikipedia. In *ACM SIGKDD workshop on interactive data exploration and analytics*, pages 18–26, 2013.

[16] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Tabel: Entity linking in web tables. In *14th International Semantic Web Conference*, pages 425–441. Springer, 2015.

[17] Russa Biswas, Rima Türker, Farshad Bakhshandegan Moghaddam, Maria Koutraki, and Harald Sack. Wikipedia Infobox Type Prediction Using Embeddings. In *DL4KGS@ ESWC*, pages 46–55, 2018.

[18] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *International Journal on Semantic Web and Information Systems*, pages 205–227. IGI global, 2009.

[19] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia-A crystallization point for the Web of Data. *Journal of web semantics*, 7(3):154–165, 2009.

[20] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*, 2008.

[21] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[22] Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. Ten years of webtables. *Proceedings of the VLDB Endowment*, 11(12):2140–2149, 2018.

[23] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. In *VLDB Endowment*, pages 538–549. VLDB Endowment, 2008.

[24] Yoan Chabot, Thomas Labbe, Jixiong Liu, and Raphaël Troncy. DAGOBAH: an end-to-end context-free tabular data semantic annotation system. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching*, pages 41–48, 2019.

[25] Jiaoyan Chen, Ernesto Jimenez-Ruiz, Ian Horrocks, and Charles Sutton. ColNet: Embedding the Semantics of Web Tables for Column Type Prediction. In 33rd *AAAI International Conference on Artificial Intelligence*, 2018.

[26] Shuang Chen, Alperen Karaoglu, Carina Negreanu, Tingting Ma, Jin-Ge Yao, Jack Williams, Andy Gordon, and Chin-Yew Lin. Linkingpark: An integrated approach for semantic table interpretation. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2020.

[27] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6), 2015.

[28] Jacob Cohen. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

[29] Marco Cremaschi, Roberto Avogadro, Andrea Barazzetti, and David Chieregato. MantisTable SE: an Efficient Approach for the Semantic Table Interpretation. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2020.

[30] Marco Cremaschi, Roberto Avogadro, and David Chieregato. MantisTable: an Automatic Approach for the Semantic Table Interpretation. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, pages 15–24, 2019.

[31] Marco Cremaschi, Roberto Avogadro, and David Chieregato. s-elBat: a Semantic Interpretation Approach for Messy taBle-s. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2022.

[32] Marco Cremaschi, Flavio De Paoli, Anisa Rula, and Blerina Spahiu. A fully automated approach to a complete semantic table interpretation. *Future Generation Computer Systems*, 112:478–500, 2020.

[33] Marco Cremaschi, Alessandra Siano, Roberto Avogadro, Ernesto Jimenez-Ruiz, and Andrea Maurino. STILTool: a semantic table interpretation evaluation tool. In *European Semantic Web Conference*, pages 61–66. Springer, 2020.

[34] Eric Crestan and Patrick Pantel. Web-scale table census and classification. In *4th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 545–554, 2011.

[35] Vincenzo Cutrona, Federico Bianchi, Ernesto Jiménez-Ruiz, and Matteo Palmonari. Tough tables: Carefully evaluating entity linking for tabular data. In *19th International Semantic Web Conference (ISWC)*, pages 328–343. Springer, 2020.

[36] Vincenzo Cutrona, Jiaoyan Chen, Vasilis Efthymiou, Oktie Hassanzadeh, Ernesto Jiménez-Ruiz, Juan Sequeda, Kavitha Srinivas, Nora Abdelmageed, Madelon Hulsebos, D Oliveira, et al. Results of SemTab 2021. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, pages 1–12. CEUR Workshop Proceedings, 2022.

[37] Dong Deng, Yu Jiang, Guoliang Li, Jian Li, and Cong Yu. Scalable column concept determination for web tables using large knowledge bases. In *PVLDB*, pages 1606–1617. VLDB Endowment, 2013.

[38] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. TURL: Table Understanding through Representation Learning. arXiv:2006.14806, 2020.

[39] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. Building the dresden web table corpus: A classification approach. In *IEEE 2nd International Symposium on Big Data Computing (BDC)*, pages 41–50. IEEE, 2015.

[40] Julian Eberius, Maik Thiele, Katrin Braunschweig, and Wolfgang Lehner. Top-k Entity Augmentation Using Consistent Set Covering. In *SSDBM*, 2015.

[41] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *16th International Semantic Web Conference (ISWC)*, pages 260–277. Springer, 2017.

[42] Lisa Ehrlinger and Wolfram Wöß. Towards a Definition of Knowledge Graphs. *SEMANTiCS*, 48(1-4):2, 2016.

[43] Ivan Ermilov and Axel-Cyrille Ngonga Ngomo. TAIPAN: automatic property mapping for tabular data. In *European Knowledge Acquisition Workshop*, pages 163–179. Springer, 2016.

[44] Yasamin Eslahi, Akansha Bhardwaj, Paolo Rosso, Kurt Stockinger, and Philippe Cudré-Mauroux. Annotating web tables through knowledge bases: A context-based approach. In 7th *Swiss Conference on Data Science (SDS)*, pages 29–34. IEEE, 2020.

[45] Jing Fang, Prasenjit Mitra, Zhi Tang, and C Lee Giles. Table header detection and classification. In 26th *Conference on Artificial Intelligence*, 2012.

[46] Besnik Fetahu, Avishek Anand, and Maria Koutraki. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference*, pages 2736–2742, 2019.

[47] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *the 25th ACM international on conference on information and knowledge management*, pages 55–64, 2016.

[48] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. arXiv:1902.09113, 2019.

[49] Tong Guo, Derong Shen, Tiezheng Nie, and Yue Kou. Web table column type detection using deep learning and probability graph model. In *International Conference on Web Information Systems and Applications*, pages 401–414. Springer, 2020.

[50] Maryam Habibi, Johannes Starlinger, and Ulf Leser. DeepTable: a permutation invariant neural network for table orientation classification. *Data Mining and Knowledge Discovery*, pages 1–21, 2020.

[51] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 139–144, 2018.

[52] Vinh Thinh Ho, Koninika Pal, and Gerhard Weikum. QuTE: Answering Quantity Queries from Web Tables. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2740–2744, 2021.

[53] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. arXiv:2003.02320, 2020.

[54] Kevin Hu, Snehalkumar'Neil'S Gaikwad, Madelon Hulsebos, Michiel A Bakker, Emanuel Zgraggen, César Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çağatay Demiralp. Viznet: Towards a large-scale visualization learning and benchmarking repository. In *CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[55] Madelon Hulsebos, Çağatay Demiralp, and Paul Groth. GitTables: A Large-Scale Corpus of Relational Tables. arXiv:2106.07258, 2021.

[56] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In *25th ACM International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pages 1500–1508, 2019.

[57] Viet-Phi Huynh, Yoan Chabot, Thomas Labbé, Jixiong Liu, and Raphaël Troncy. From Heuristics to Language Models: A Journey Through the Universe of Semantic Table Interpretation with DAGOBAH. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2022.

[58] Viet-Phi Huynh, Jixiong Liu, Yoan Chabot, Frédéric Deuzé, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. DAGOBAH: Table and Graph Contexts for Efficient Semantic Annotation of Tabular Data. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2021.

[59] Viet-Phi Huynh, Jixiong Liu, Yoan Chabot, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. DAGOBAH: Enhanced Scoring Algorithms for Scalable Annotations of Tabular Data. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2020.

[60] Yusra Ibrahim, Mirek Riedewald, Gerhard Weikum, and Demetrios Zeinalipour-Yazti. Bridging quantities in tables and text. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1010–1021. IEEE, 2019.

[61] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. A latent factor model for highly multi-relational data. In *International Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3176–3184, 2012.

[62] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.

[63] Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, and Kavitha Srinivas. SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems. In *European Semantic Web Conference (ESWC)*, pages 514–530. Springer, 2020.

[64] Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Kavitha Srinivas, and Vincenzo Cutrona. Results of SemTab 2020. In *CEUR Workshop Proceedings*, volume 2775, pages 1–8, 2020.

[65] Emilia Kacprzak, José M Giménez-García, Alessandro Piscopo, Laura Koesten, Luis-Daniel Ibáñez, Jeni Tennison, and Elena Simperl. Making sense of numerical data-semantic labelling of web tables. In *European Knowledge Acquisition Workshop*, pages 163–178. Springer, 2018.

[66] Udayan Khurana and Sainyam Galhotra. Semantic annotation for tabular data, 2019.

[67] Larissa R Lautert, Marcelo M Scheidt, and Carina F Dorneles. Web table taxonomy and formalization. *ACM SIGMOD Record*, 42(3):28–33, 2013.

[68] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A large public corpus of web tables containing time and context metadata. In 25th *International Conference Companion on World Wide Web*, pages 75–76, 2016.

[69] Oliver Lehmberg, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, and Christian Bizer. The mannheim search join engine. *Journal of Web Semantics*, 35:159–166, 2015.

[70] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. Pytorch-biggraph: A large scale graph embedding system. In *Conference on Machine Learning and Systems (MLSys)*, volume 1, pages 120–131, 2019.

[71] Xinhe Li, Shuxin Wang, Wei Zhou, Gongrui Zhang, Chenghuan Jiang, Tianyu Hong, and Peng Wang. KGCODE-Tab Results for SemTab 2022. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*. CEUR Workshop Proceedings, 2022.

[72] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3(1-2):1338–1347, 2010.

[73] Pasquale Lisena and Raphaël Troncy. Combining music specific embeddings for computing artist similarity. In 18th *International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Demo Track, Suzhou, China*, 2017.

[74] Jixiong Liu, Yoan Chabot, Raphaël Troncy, Viet-Phi Huynh, Thomas Labbé, and Pierre Monnin. From Tabular Data to Knowledge Graphs: A Survey of Semantic Table Interpretation Tasks and Methods. *Journal of Web Semantics*, 2022. Under revision.

[75] Jixiong Liu, Viet-Phi Huynh, Yoan Chabot, and Raphaël Troncy. Radar Station: Using KG Embeddings for Semantic Table Interpretation and Entity Disambiguation. In 21st *International Semantic Web Conference (ISWC)*, 2022.

[76] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[77] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia spotlight: shedding light on the web of documents. In 7th *International Conference on Semantic Systems*, pages 1–8, 2011.

[78] Pham Minh, Alse Suresh, A. Knoblock Craig, and Szekelyèle Pedro. Semantic Labeling: A Domain-Independent Approach. In 15th *International Semantic Web Conference (ISWC)*, pages 446—462. Springer, 2016.

[79] Hiroaki Morikawa. Semantic Table Interpretation using LOD4ALL. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, pages 49–56, 2019.

[80] Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic message passing for generating linked data from tables. In 12th *International Semantic Web Conference (ISWC)*, pages 363–378. Springer, 2013.

[81] Varish Mulwad, Tim Finin, Zareen Syed, Anupam Joshi, et al. Using linked data to interpret tables. In 1st *International Workshop on Consuming Linked Data (COLD)*, 2010.

[82] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Using linked data to mine RDF from wikipedia's tables. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 533–542, 2014.

[83] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145, 2012.

[84] Sebastian Neumaier, Jürgen Umbrich, Josiane Xavier Parreira, and Axel Polleres. Multi-level semantic labelling of numerical values. In 15th *International Semantic Web Conference (ISWC)*, pages 428–445. Springer, 2016.

[85] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

[86] Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. MTab: Matching Tabular Data to Knowledge Graph using Probability Models. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2019.

[87] Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. TabEAno: table to knowledge graph entity annotation. arXiv:2010.01829, 2020.

[88] Phuc Nguyen, Khai Nguyen, Ryutaro Ichise, and Hideaki Takeda. Embnum: Semantic labeling for numerical values with deep metric learning. In *Joint International Semantic Technology Conference*, pages 119–135. Springer, 2018.

[89] Phuc Nguyen, Ikuya Yamada, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. Mtab4wikidata at semtab 2020: Tabular data annotation with wikidata. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2020.

[90] Phuc Nguyen, Ikuya Yamada, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. SemTab 2021: Tabular Data Annotation with MTab Tool. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2021.

[91] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In 28th *International Conference on Machine Learning (ICML)*, 2011.

[92] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: scalable machine learning for linked data. In 21th *International Conference Companion on World Wide Web(WWW)*, pages 271–280, 2012.

[93] Natasha Noy, Matthew Burgess, and Dan Brickley. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In 28th *The Web Conference (WWW)*, 2019.

[94] Daniela Oliveira and Mathieu d'Aquin. Adog-annotating data with ontologies and graphs. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2019.

[95] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. arXiv:1508.00305, 2015.

[96] Gerald Penn, Jianying Hu, Hengbin Luo, and Ryan McDonald. Flexible web document analysis for delivery to narrow-bandwidth devices. In 6th *International Conference on Document Analysis and Recognition*, pages 1074–1078. IEEE, 2001.

[97] Aleksander Pivk, Philipp Cimiano, and York Sure. From tables to frames. In *International Semantic Web Conference (ISWC)*, pages 166–181. Springer, 2004.

[98] S Krishnamurthy Ramnandan, Amol Mittal, Craig A Knoblock, and Pedro Szekely. Assigning semantic labels to data sources. In *European Semantic Web Conference (ESWC)*, pages 403–417. Springer, 2015.

[99] Jenn Riley. Understanding metadata. *Washington DC, United States: National Information Standards Organization (http://www. niso. org/publications/press/Understanding-Metadata. pdf)*, 23, 2017.

[100] Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all? Analyzing the differences between DBpedia, YAGO, Wikidata & co. In *Joint German/Austrian*

Conference on Artificial Intelligence (Künstliche Intelligenz), pages 366–372. Springer, 2017.

[101] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer, 2016.

[102] Dominique Ritze. *Web-scale web table to knowledge base matching*. PhD thesis, University of Mannheim, 2017.

[103] Dominique Ritze and C. Bizer. Matching Web Tables To DBpedia - A Feature Utility Study. In *International Conference on Extending Database Technology (EDBT)*, pages 210—-221, 2017.

[104] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. Matching html tables to dbpedia. In 5th *International Conference on Web Intelligence, Mining and Semantics*, pages 1–6, 2015.

[105] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. Matching HTML Tables to DBpedia. In 5th *International Conference on Web Intelligence, Mining and Semantics (WIMS)*, pages 1–6, 2015.

[106] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[107] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. Finding related tables. In *ACM SIGMOD International Conference on Management of Data*, 2012.

[108] Christophe Sarthou-Camy, Guillaume Jourdain, Yoan Chabot, Pierre Monnin, Deuzé, Viet-Phi Huynh, Jixiong Liu, Thomas Labbé, and Raphaël Troncy. DAGOBAH UI: A New Hope For Semantic Table Interpretation. In 19th *European Semantic Web Conference (ESWC), Poster and Demo Track*. Springer, 2022.

[109] Yoones A Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. Knowledge base augmentation using tabular data. In *LDOW*, 2014.

[110] Renat Shigapov, Philipp Zumstein, Jan Kamlah, Lars Oberländer, Jörg Mechnich, and Irene Schumm. bbw: Matching CSV to Wikidata via Meta-lookup. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, volume 2775, pages 17–26. RWTH, 2020.

[111] Anastasia Shimorina, Johannes Heinecke, and Frédéric Herledan. Knowledge Extraction From Texts Based on Wikidata. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 297–304, 2022.

[112] Gaurav Singh, Siffi Singh, Joshua Wong, and Amir Saffari. Relation Extraction from Tables using Artificially Generated Metadata. arXiv:2108.10750, 2021.

[113] Bram Steenwinckel, Filip De Turck, and Femke Ongeane. MAGIC: Mining an Augmented Graph using INK, starting from a CSV. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2021.

[114] Bram Steenwinckel, Gilles Vandewiele, Filip De Turck, and Femke Ongenae. Csv2kg: Transforming tabular data into semantic knowledge. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2019.

[115] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. Annotating Columns with Pre-trained Language Models. arXiv:2104.01785, 2021.

[116] Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. Table cell search for question answering. In *25th International Conference on World Wide Web*, pages 771–782, 2016.

[117] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv:1902.10197, 2019.

[118] Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Acquiring temporal constraints between relations. In *the 21st ACM international conference on Information and knowledge management*, pages 992–1001, 2012.

[119] Aditya Tandon, Aiiad Albeshri, Vijey Thayananthan, Wadee Alhalabi, Filippo Radicchi, and Santo Fortunato. Community detection in networks using graph embeddings. *Physical Review E*, 103(2):022316, 2021.

[120] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. YAGO 4: A Reasonable Knowledge Base. In *European Semantic Web Conference (ESWC)*, pages 583–596. Springer, 2020.

[121] Cui Tao and David W Embley. Automatic hidden-web table interpretation, conceptualization, and semantic annotation. *Data & Knowledge Engineering*, 68(7):683–703, 2009.

[122] Avijit Thawani, Minda Hu, Erdong Hu, Husain Zafar, Naren Teja Divvala, Amandeep Singh, Ehsan Qasemi, Pedro A Szekely, and Jay Pujara. Entity Linking to Knowledge Graphs to Infer Column Types and Properties. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, volume 2019, pages 25–32, 2019.

Bibliography

[123] Philip K Thornton, A Stroud, N Hatibu, C Legg, S Ly, Stephen Twomlow, K Molapong, A Notenbaert, R Kruska, and R von Kaufmann. Site selection to test an integrated approach to agricultural research for development: combining expert knowledge and participatory Geographic Information System methods. *International Journal of Agricultural Sustainability*, 4(1):39–60, 2006.

[124] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, pages 2071–2080. PMLR, 2016.

[125] WD Van Eeden, Johan Pieter De Villiers, RJ Berndt, Willem AJ Nel, and Erik Blasch. Micro-Doppler radar classification of humans and animals in an operational environment. *Expert Systems with Applications*, 102:1–11, 2018.

[126] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv:1706.03762, 2017.

[127] Petros Venetis, Alon Y Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, and Gengxin Miao. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011.

[128] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledge base. *Communications of the ACM*, 57(10):78–85, 2014.

[129] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. Tcn: Table convolutional network for web table interpretation. arXiv:2102.09460, 2021.

[130] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q Zhu. Understanding tables on the web. In *International Conference on Conceptual Modeling*, pages 141–155. Springer, 2012.

[131] Meihong Wang, Linling Qiu, and Xiaoli Wang. A survey on knowledge graph embeddings for link prediction. *Symmetry*, 13(3):485, 2021.

[132] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[133] Yalin Wang and Jianying Hu. Detecting tables in HTML documents. In 5th *IAPR International Workshop on Document Analysis Systems*, pages 249–260. Springer, 2002.

[134] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI Conference on Artificial Intelligence*, 2014.

[135] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. In *27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790, 2021.

[136] Fei Wu and Daniel S Weld. Automatically refining the wikipedia infobox ontology. In *17th international conference on World Wide Web*, pages 635–644, 2008.

[137] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Inferring a universal probabilistic taxonomy from the web. Technical report, Technical report, Microsoft Research, 2010.

[138] Liu Xiaoxue, Bai Xuesong, Wang Longhe, Ren Bingyuan, Lu Shuhan, and Li Lin. Review and trend analysis of knowledge graphs for crop pest and diseases. *IEEE Access*, 7:62251–62264, 2019.

[139] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *ACM SIGMOD International Conference on Management of Data*, pages 97–108, 2012.

[140] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. arXiv:1412.6575, 2014.

[141] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data, 2020.

[142] Munirah Mohd Yusof, Nur Fazliyana Rosli, Muhaini Othman, Rozlini Mohamed, and Mohd Hafizul Afifi Abdullah. M-DCocoa: M-agriculture expert system for diagnosing cocoa plant diseases. In *International Conference on Soft Computing and Data Mining*, pages 363–371. Springer, 2018.

[143] Allison Zhang and Don Gourley. *Creating digital collections: a practical guide*. Elsevier, 2008.

[144] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. Sato: Contextual Semantic Type Detection in Tables, 2019.

[145] Meihui Zhang and Kaushik Chakrabarti. Infogather+ semantic matching and annotation of numeric and time-varying attributes in web tables. In *ACM SIGMOD International Conference on Management of Data*, pages 145–156, 2013.

[146] Shuo Zhang and Krisztian Balog. Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 255–264, 2017.

[147] Shuo Zhang and Krisztian Balog. Auto-completion for data cells in relational tables. In *the 28th ACM International Conference on Information and Knowledge Management*, pages 761–770, 2019.

[148] Shuo Zhang and Krisztian Balog. Recommending related tables. arXiv:1907.03595, 2019.

[149] Shuo Zhang and Krisztian Balog. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–35, 2020.

[150] Shuo Zhang, Edgar Meij, Krisztian Balog, and Ridho Reinanda. Novel entity discovery from web tables. In *The Web Conference*, pages 1298–1308, 2020.

[151] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

[152] Ziqi Zhang. Towards efficient and effective semantic table interpretation. In *3th International Semantic Web Conference*, pages 487–502. Springer, 2014.

[153] Ziqi Zhang. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, 8(6):921–957, 2017.

[154] Yiwei Zhou, Siffi Singh, and Christos Christodoulopoulos. Tabular Data Concept Type Detection Using Star-Transformers. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3677–3681, 2021.

[155] Zhaocheng Zhu, Shizhen Xu, Jian Tang, and Meng Qu. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In *The World Wide Web Conference (WWW)*, pages 2494–2504, 2019.

[156] Stefan Zwicklbauer, Christoph Einsiedler, Michael Granitzer, and Christin Seifert. Towards Disambiguating Web Tables. In *International Semantic Web Conference (Posters & Demos)*, pages 205–208, 2013.

[157] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. Do We Need Entity-Centric Knowledge Bases for Entity Disambiguation? In *13th International Conference on Knowledge Management and Knowledge Technologies*, pages 1–8, 2013.