Web Semantics: Science, Services and Agents on the World Wide Web 76 (2023) 100761

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods

Jixiong Liu [a,b], Yoan Chabot [a,*], Raphaël Troncy [b], Viet-Phi Huynh [a], Thomas Labbé [a], Pierre Monnin [a]

[a] Orange, France
[b] EURECOM, Sophia Antipolis, France

ARTICLE INFO

Article history:
Received 13 December 2021
Received in revised form 5 October 2022
Accepted 25 October 2022
Available online 19 November 2022

Keywords:
Semantic table interpretation
Table annotation
Tabular data
Knowledge graph

ABSTRACT

Tabular data often refers to data that is organized in a table with rows and columns. We observe that this data format is widely used on the Web and within enterprise data repositories. Tables potentially contain rich semantic information that still needs to be interpreted. The process of extracting meaningful information out of tabular data with respect to a semantic artefact, such as an ontology or a knowledge graph, is often referred to as Semantic Table Interpretation (STI) or Semantic Table Annotation. In this survey paper, we aim to provide a comprehensive and up-to-date state-of-the-art review of the different tasks and methods that have been proposed so far to perform STI. First, we propose a new categorization that reflects the heterogeneity of table types that one can encounter, revealing different challenges that need to be addressed. Next, 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. Finally, we detail what are the remaining scientific barriers to be able to truly automatically interpret any type of tables that can be found in the wild Web.

1. Introduction

Data formats such as CSV/TSV, PARQUET, XML and JSON are commonly used to train machine learning algorithms. We focus on CSV, TSV and spreadsheet files and we argue that while this tabular data format is, compact, readable and simple to process, it does not self-explain the meaning of the information even if headers are present. 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 [1–3]. 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 knowledge graphs (KG) which can be of general purposes such as DBpedia [4] and Wikidata [5], or enterprise specific. This problem is known as Semantic Table Interpretation (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. It opens up opportunities for new semantic-based services: leverage semantic annotation to better index datasets in search engines [6,7], improve question/answering systems [8–10], enrich knowledge bases [11–13] or enhance dataset recommendation [14]. The emergence of specialized search engines for datasets such as Google Dataset Search [15,16] is another prominent example.

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 [17]. For example, in the table depicted in Fig. 7(b) (page 7), 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

* Corresponding author.
E-mail address: yoan.chabot@orange.com (Y. Chabot).

https://doi.org/10.1016/j.websem.2022.100761
1570-8268/© 2022 Elsevier B.V. All rights reserved.

particular, the fact that the "Manufacturer" column only contains companies.

This work aims to comprehensively define the various sub-tasks that belong to STI and to review the many methods that have been proposed so far, along with their limitations and performances on well-established evaluation datasets of the STI community. To the best of our knowledge, such a survey is still missing in the community. Related surveys have recently been published, on Web tables retrieval and enrichment [18], on tables extraction, transformation and understanding [19], on deep learning with tabular data [20,21] and on information extraction on the Web [22]. However, those surveys do not focus on the STI process. We, therefore, aim to complement these surveys by providing a new categorization reflecting the heterogeneity of tabular data that one can encounter and that yields new challenges.

The remainder of this paper is structured as follows. In Section 2, we describe the research methodology that has been used to make this survey. Next, we provide some preliminaries that are essential for understanding the context of STI and we propose a new fine-grained taxonomy of table types (Section 3). In Section 4, we define five sub-tasks that are relevant to STI, namely: cell-entity annotation, column-type annotation, columns-property annotation [2,3], topic annotation, and row-to-instance annotation [11]. We also propose a macro-common pipeline that fulfils the tasks of STI from pre-processing of the input table to the final annotation results. In Section 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. In Section 6, we make an inventory of the gold-standard datasets commonly used to evaluate STI approaches and we analyse their strengths and weaknesses. We describe the current performances of STI systems on these datasets in Section 7. We elicit the open scientific challenges for the community in Section 8 before concluding the paper in Section 9.

2. Research methodology

In this section, we describe our research methodology for collecting and to analysing scholar articles relevant to STI tasks, approaches and results.

We observe that most of the work was published between 2000 and 2021. Consequently, our work covers articles published within these two decades. To collect the relevant papers, we primarily make use of Google Scholar. We first use a combination of keywords from two lists: a list of terms reflecting the tasks we are interested in ("semantic annotation", "annotation", "semantic interpretation", "interpretation", "knowledge graph matching", "semantic matching", "semantic labelling", "type prediction", "entity linking", "entity typing", "knowledge graph mapping", "semantic mapping", "relation extraction") and another list of terms scoping the domain ("web table", "table", "tabular data", "structured data"). This list of terms was manually created when reading relevant papers such as the ones from the SemTab challenge.[1] For example, a valid combination is "web table semantic annotation". For each query, we retrieve the 10 most cited papers among the relevant results. We read the abstract of each retrieved paper and judge whether the paper is relevant for STI (e.g., presentation of a method for matching a table element with a given KG, introduction of a new dataset) or not. Then, we extend this initial corpus by retrieving, for each paper previously selected, their 10 most cited papers. The newly selected papers are then filtered according to the methodology used in the first phase. Besides Google Scholar, we use the same keyword-based methodology and co-citation network for finding relevant papers that have been made available on arXiv. This enables us to account for the very latest research work in this fast-moving field even if these papers have not yet been peer-reviewed or cited. Finally, we have also collected the papers from the SemTab challenge which is the most relevant competition for this domain.

Using this method, we generated 48 combinations of keywords to search on Google Scholar and we selected 38 distinct references in the first phase. The co-citation network has brought 58 additional papers in the second phase. We added 7 arXiv papers and 22 SemTab papers to this selection. In summary, while more than seven hundred papers have been collected and read, 114 papers have been assessed to be relevant in this two-phase selection process. Fig. 1 depicts the time distribution of the publication of these articles. It shows the growing importance of STI in the literature over the last decade. We manually tagged each article into six categories (Fig. 2). Note that some articles may address more than one research focus. For example, [23] provides both a dataset and a method for performing STI. The core of this survey is focused on STI systems but it also covers tangentially related topics such as the need for knowledge graphs and schemas to anchor the interpretation of tabular data or the importance of pre-processing tables, using NLP techniques for example.

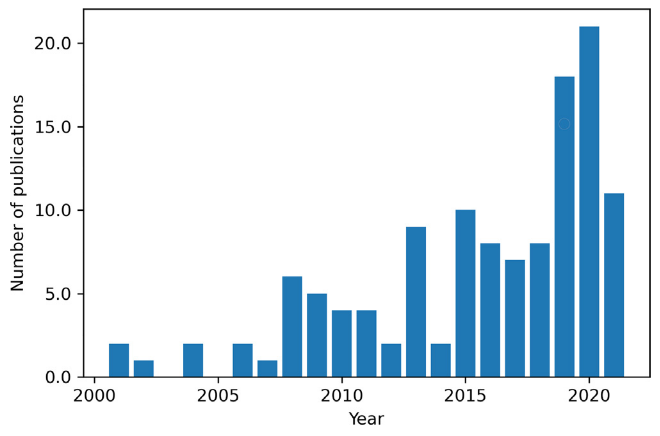


Fig. 1. Distribution of the publication years of the papers selected for this review.

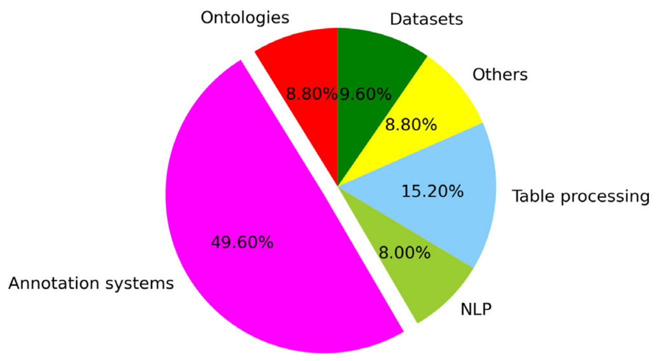


Fig. 2. Distribution of the topics of the papers selected for this survey.

3. Preliminaries

The first and main input of an STI system is the table itself. As there is an important heterogeneity of tables considering their layout, provenance, and usage, we propose a new fine-grained

[1] https://www.cs.ox.ac.uk/isg/challenges/sem-tab/.

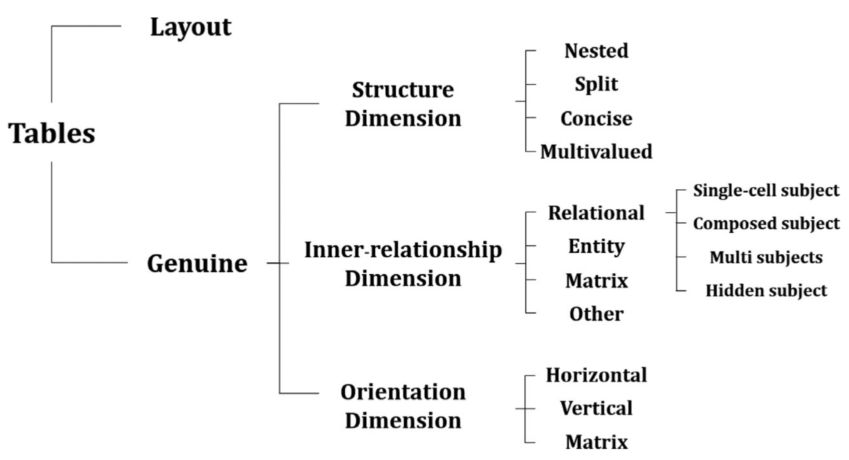


Fig. 3. Classification of table types with a finer-grained analysis of genuine tables along three dimensions: structure, inner-relationship and orientation.

classification based on existing classifications with a deeper analysis of relational tables (Section 3.1). 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.

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. Section 3.2 provides a list of elements attached to the table carrying semantic information useful for interpretation.

Finally, STI uses KGs as sources of information and as references for producing annotations. Section 3.3 highlights the most commonly used KGs for table interpretation and discusses their specificities and what they imply for the STI tasks.

3.1. 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 $\mathcal{T}_{ij}$ $(0 \le i \le n-1, 0 \le j \le m-1)$ 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, before interpreting a table, it is important to identify its type so that potential specificities can be taken into account in the STI process.

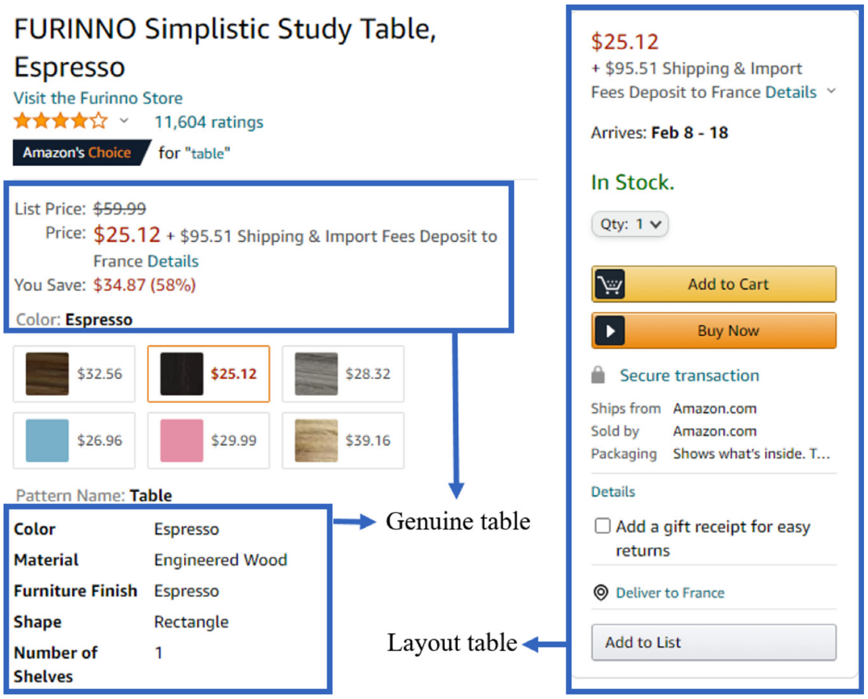
We introduce a multi-level classification of tables based on several aspects. The first classification effort splits tables into two high-level categories: genuine and non-genuine [24,25]. Genuine tables are firstly defined as two-dimensional structures with simple cells (i.e. short and without any complex structures) and a high level of coherence (syntactically and semantically) within rows and columns in [24]. Non-genuine tables are structures used to group contents for easy viewing [25]. 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 Fig. 6(b). According to the definition provided in [24] and used in [26], this table will not be considered a genuine table while, arguably, this table carries semantic information worth to be processed.

In more recent works, [27,28] 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 Fig. 6(d) where there is no information about the common relationship between each cell.

In order to cope with these shortcomings, we propose a new classification of table types, shown in Fig. 3, that rely on the existing work presented in [18,24,26,28–31] and consider overlapping dimensions. This classification also contributes to a 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 in order to maximize user comfort and site usability. A layout table used on Amazon to provide the order interface is given in Fig. 4. **Genuine tables** represent in rows or columns human-understandable knowledge. In the literature, genuine tables are said to be short and without complex structure [24]. We relax this concept by considering that tables with a high level of coherence (syntactically and semantically) within rows and columns are genuine tables, without considering the table complexity. For example, in Fig. 4, the semantic of the two genuine tables is the description attributes (e.g., price, colour) 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 [18,26,28,29]. 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 Fig. 5(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.


FURINNO Simplistic Study Table, Espresso
Visit the Furinno Store
11,604 ratings
Amazon's Choice for "table"

List Price: ~~$59.99~~
Price: $25.12 + $95.51 Shipping & Import Fees Deposit to France Details
You Save: $34.87 (58%)
Color: Espresso

$32.56 $25.12 $28.32
$26.96 $29.99 $39.16

Pattern Name: Table

Color	Espresso
Material	Engineered Wood
Furniture Finish	Espresso
Shape	Rectangle
Number of Shelves	1

$25.12
+ $95.51 Shipping & Import Fees Deposit to France Details
Arrives: Feb 8 - 18
In Stock.
Qty: 1
Add to Cart
Buy Now
Secure transaction

Ships from	Amazon.com
Sold by	Amazon.com
Packaging	Shows what's inside. T...

Details
Add a gift receipt for easy returns
Deliver to France
Add to List

Fig. 4. Illustration of genuine tables and layout tables on the Amazon website. [2] (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.1.1. Structure dimension

[28] 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 Fig. 5. **Nested tables** contain one or more tables in one or more of their cells. Fig. 5(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. [28] 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 Fig. 5(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 in order to avoid repetitions of cells referring to the same content in rows and/or columns. In the table presented in Fig. 5(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 Fig. 5(d), the cells of the column "lines used" contain a list of route lines.

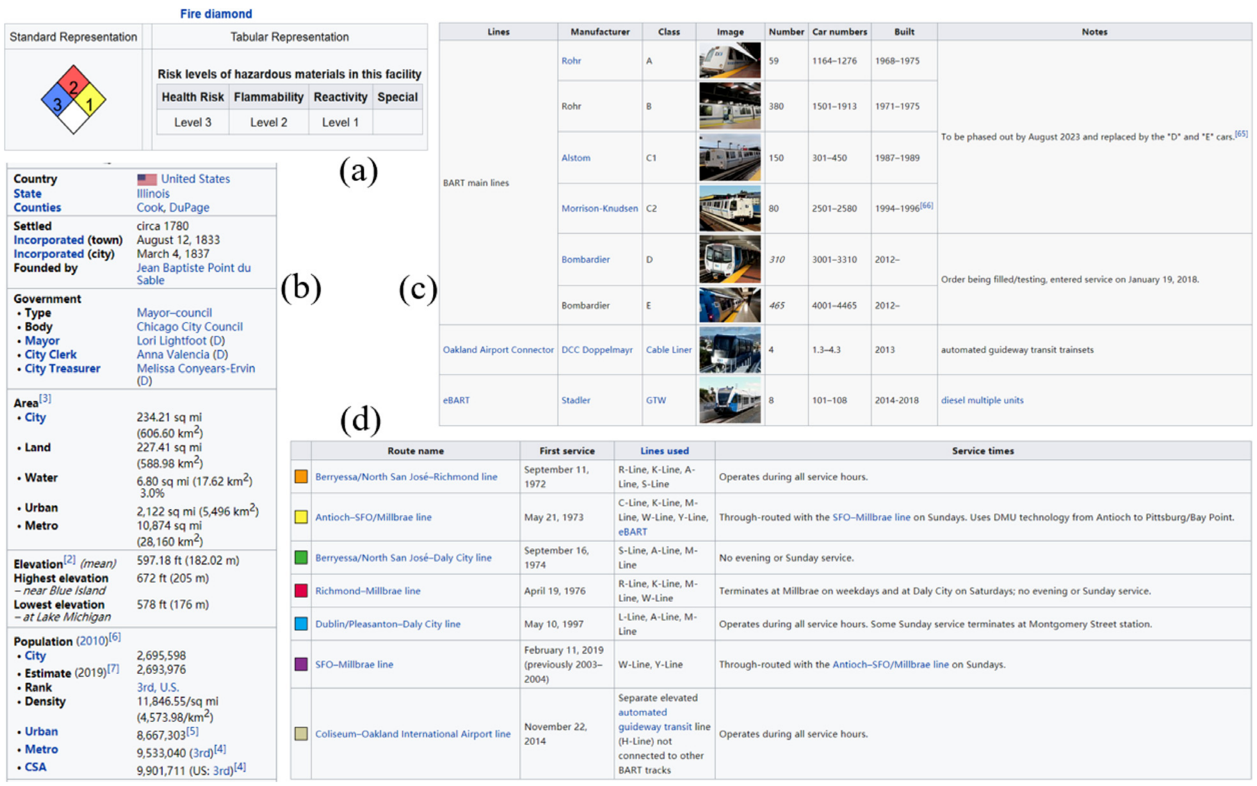
3.1.2. Inner-relationship dimension

The inner-relationship dimension considers the topology of the semantic connection between the cells. [27] first gave a detailed classification of relational knowledge tables into listings, attribute or value tables, matrices, enumerations, and forms. [29] has extracted Web tables that are classified into listings, matrices and other tables to capture enumerations, calendars, etc. [26,30] 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 header, usually in the first row or the first few rows for horizontal tables. As an example, Fig. 6(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 Fig. 6(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. Fig. 6(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 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, Fig. 6(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 [26,30,31]. However, relational tables exhibit an important diversity, especially in the representations of entities. We propose to further classify them depending

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



(a)

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	

(b)

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]

(c)

Lines	Manufacturer	Class	Image	Number	Car numbers	Built	Notes
BART main lines	Rohr	A		59	1164–1276	1968–1975	To be phased out by August 2023 and replaced by the "D" and "E" cars.[65]
	Rohr	B		380	1501–1913	1971–1975	
	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.

Fig. 5. Examples of different structures of tables: (a) a nested table,[3] (b) a split table,[4] (c) a concise table (column "lines"),[5] (d) a multivalued table (column "lines used").[6]

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 Fig. 6(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 (Fig. 7). **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 Fig. 7(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 Fig. 7(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 Fig. 7(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 Fig. 7(d), each row describes the result of a football match, but the mention of the match itself is not made explicit in the table.

3 https://en.wikipedia.org/wiki/Table_(information).

4 https://en.wikipedia.org/wiki/Chicago.

5 https://en.wikipedia.org/wiki/Bay_Area_Rapid_Transit#Rollingstock.

6 https://en.wikipedia.org/wiki/Bay_Area_Rapid_Transit#Infrastructure.

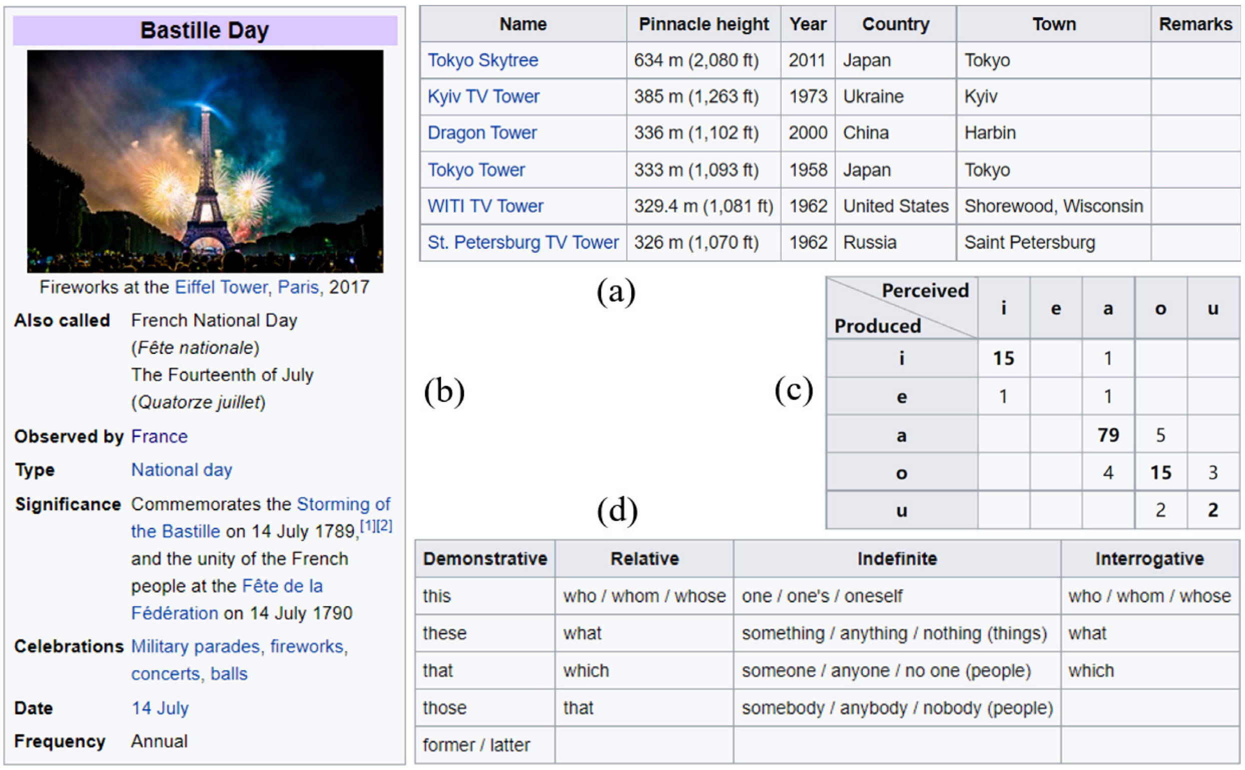
3.1.3. *Orientation dimension*

The orientation dimension considers the direction of the relationships inside a table [28,29]. 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 Fig. 6(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 Fig. 6(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 Fig. 6(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.

3.1.4. *Table types statistics*

We introduced in Fig. 3 a multi-dimensional and fine-grained classification of table types. In this section, we aim to survey how frequent these types of tables are used in the wild. DWTC [29] has randomly selected 26,645 tables from the WDC Web Table Corpus [30] and has concluded that the resulting corpus was made of 96% layout tables and 4% genuine tables. Regarding the structure dimension, [28] 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. [28] has also provided the distribution of nested tables, split tables, concise tables, and multivalued tables among the



(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	

(b)

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

(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			

Fig. 6. Examples of different inner-relationships of tables: (a) a horizontal relational table,[7] (b) an entity table,[8] (c) a matrix table,[9] (d) an enumeration table[10] which belongs to "other genuine tables".

relational knowledge tables, which are respectively 3.7%, 2.6%, 12.9%, and 74.9%. Regarding the inner-relationship dimension, [30] 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 [28], 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.

3.2. Metadata

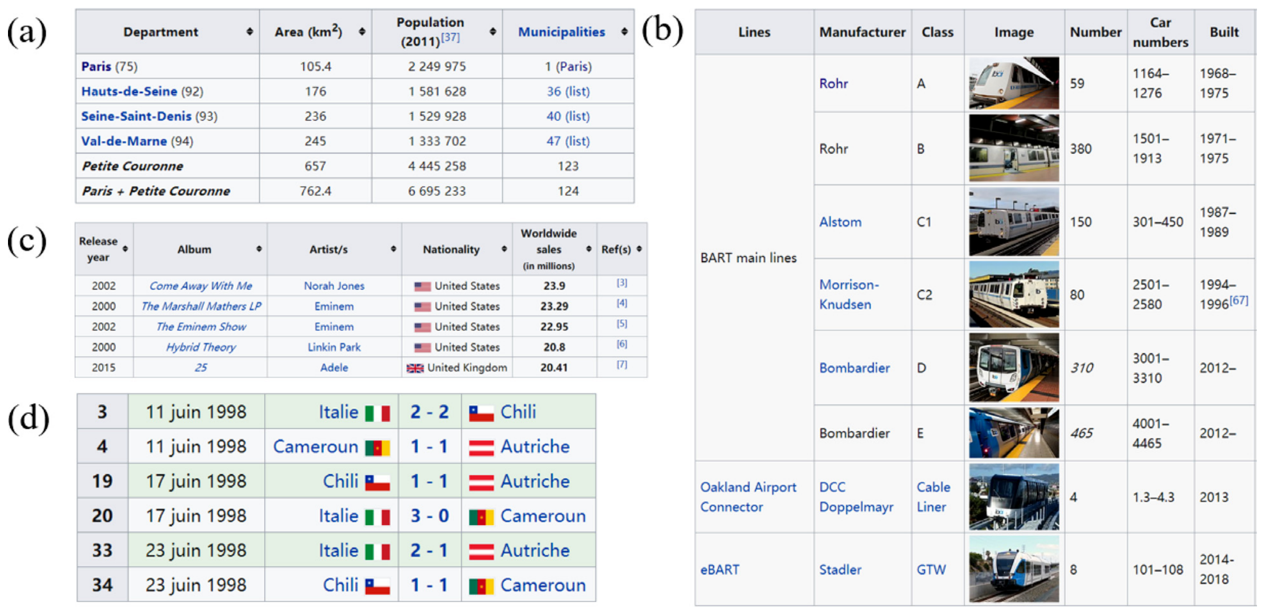
Several STI works stress that one of the challenges to be addressed is the loss of context when annotating a table [18]. Indeed, tables do not constitute the unique source of information that can be used by STI processes since the context in which they appear may provide complementary or novel information. 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, preserve, and manage extracted knowledge units. Depending on its structure, purpose, and provenance, metadata is split into descriptive, structural, and administrative metadata [32]. Such a definition was originally used in digital collection [33] 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 Fig. 8(a) can include its provenance (i.e. the 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 [34,35].

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 Fig. 8(b) allow to detect table cells, row ordering and the presence of headers. Such structural patterns could benefit STI approaches such as [36,37].

Administrative metadata often captures information such as the process of creation or data acquisition for a table. For example, in Fig. 8(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

7 https://en.wikipedia.org/wiki/Eiffel_Tower.
8 https://en.wikipedia.org/wiki/Bastille_Day.
9 https://en.wikipedia.org/wiki/Whistled_language#Lack_of_comprehension.
10 https://en.wikipedia.org/wiki/Pronoun#English_pronouns.



(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

Fig. 7. Examples of different relational tables: (a) a single-cell-subject relational table,[11] (b) a composed-subject relational table,[12] (c) a multi-subject relational table,[13] (d) a hidden-subject relational table.[14]

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 [38]. Furthermore, if a table element consists of a hyperlink (e.g., hyperlinks in infoboxes of Wikipedia), this mapping relationship also constitutes metadata. STI systems that leverage metadata as an input source are identified in Table 1.

3.3. Knowledge graphs

Knowledge graphs are often associated with linked data technologies and projects since they focus on interrelations between concepts and entities [39]. In essence, KGs are semantic networks that formally describe things or entities of the real world and their relationships [40]. Each entity is identified by a globally unique URI [41]. Atomic elements of KGs are triples ⟨subject, predicate, object⟩. For example, the population of Paris can be represented by the triple ⟨Paris, hasPopulation, 2m⟩ where the predicate hasPopulation qualifies the relationship holding between the entity Paris and the value 2m.

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 particular 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 [42]. To illustrate, in the "Crop, Pest, and Diseases" field, domain-specific KGs play a substantial role in agriculture [43], greenhouse environment [44], and economic benefit analysis [45]. The Bio2RDF project which focuses on Life Sciences [46] is an example of a domain-specific KG that is largely used.

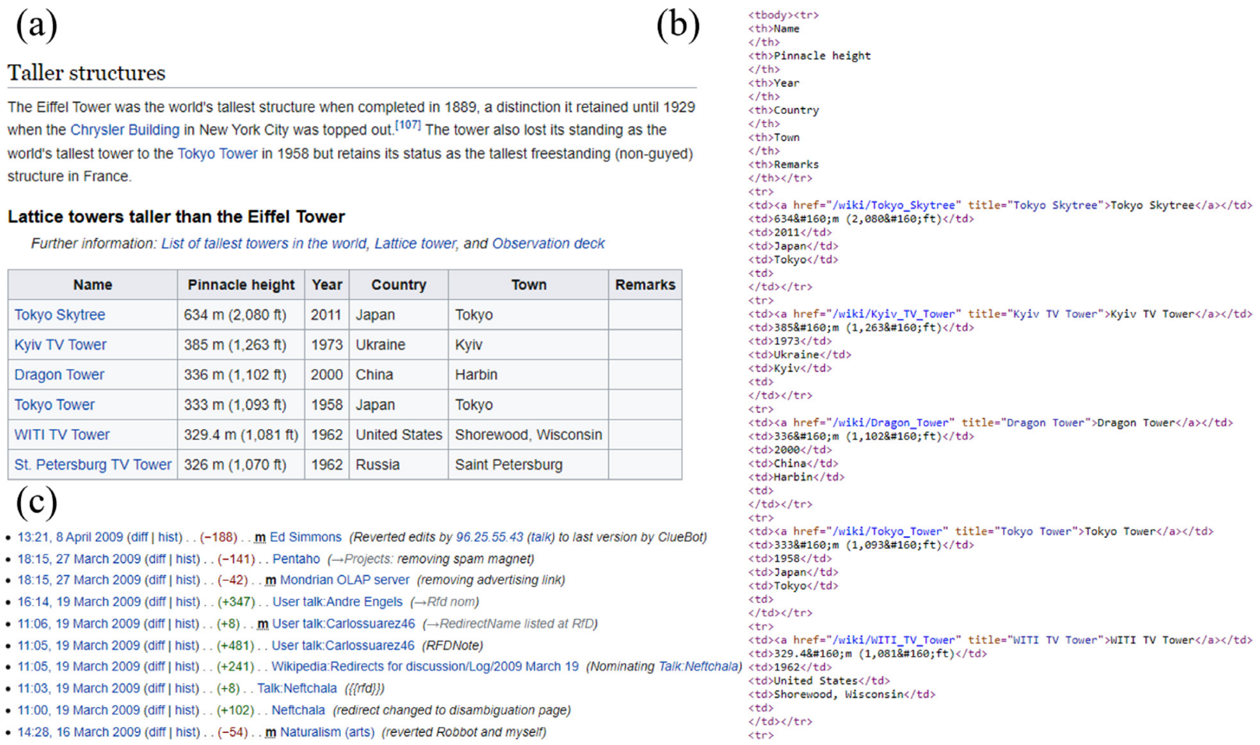
An encyclopedic KG is generally large, spanning multiple domains, and is often openly and collaboratively edited. This openness is reflected by the Linked Open Data cloud [47] which includes the following largest encyclopedic KGs. **DBpedia** [4] is one of the main hubs of the LOD cloud because of its 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 [48] that are convenient for mapping mentions contained in unstructured data with KG entities. **Wikidata** [5] 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** [49] 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) [50] 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. The column KG of Table 1 provides the specific KGs which have been used in each STI system reviewed in this paper. Indeed, understanding the content of a table comes down to identifying the entities mentioned in the table cells and the relationships

[11] https://en.wikipedia.org/wiki/France#Major%20cities.
[12] https://en.wikipedia.org/wiki/Bay_Area_Rapid_Transit.
[13] https://en.wikipedia.org/wiki/List_of_best-selling_albums_of_the_21st_century.
[14] https://fr.wikipedia.org/wiki/Coupe_du_monde_de_football_1998.

Table 1
STI systems are classified into three families. We only consider the annotation tasks declared by the authors and when they have related evaluations. "R2I" indicates the task "Row-to-instance"; "TA" indicates the task "Topic annotation"; "$\mathcal{T}_{i*}$" indicates that, when labelling 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.

Approaches			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. [51]		✓	✓				✓				Custom	Custom Web Tables	2011
		Wang et al. [37]		✓				✓	✓	✓			Probase	Custom Wikipedia Tables	2012
		Deng et al. [52]		✓					✓				FreeBase, YAGO	Custom Wikipedia Tables	2013
		Sekhavat et al. [53]			✓				✓				DBpedia	Custom Web Tables	2014
		TabEL [54]	✓					✓	✓				YAGO	Limaye	2015
		ADOG [55]	✓	✓	✓			✓	✓				DBpedia	SemTab 2019	2019
		Tabularisi [56]	✓	✓	✓				✓				DBpedia	T2D, VizNet	2019
		C^2 [57]		✓					✓	✓	✓		DBpedia, Wikidata	Limaye, ISWC2017, SemTab 2019, T2D, Semantification, Custom Data	2020
		Magic [58]	✓	✓	✓				✓	✓			DBpedia, Wikidata	SemTab 2021	2021
		Alobaid et al. [59]		✓						✓			DBpedia	SemTab 2021, T2D	2022
	Iterative	Zwicklbauer et al. [60]		✓					✓				DBpedia	Custom Wikipedia Tables	2013
		T2K [11]			✓	✓	✓	✓	✓				DBpedia	T2D	2015
		TableMiner+ [17]			✓	✓	✓	✓	✓	✓		✓	Freebase	Limaye, IMDB, MusicBrainz	2017
		LOD4ALL [61]	✓	✓	✓			✓	✓				DBpedia	SemTab 2019	2019
		CSV2KG [62]	✓	✓	✓			✓	✓	✓			DBpedia	SemTab 2019	2019
		MTab [63–65]	✓	✓	✓			✓	✓	✓			DBpedia, Wikidata	SemTab 2019–2021	2019
		LinkingPark [66]	✓	✓	✓			✓	✓				Wikidata	SemTab 2020	2019
		DAGOBAH SL [67–69]	✓	✓	✓			✓	✓				DBpedia, Wikidata	SemTab 2019–2022	2019
		MantisTable [70,71]	✓	✓	✓	✓		✓	✓				DBpedia, Wikidata	SemTab 2019–2021	2019
		JenTab [72–74]	✓	✓	✓			✓	✓				DBpedia, Wikidata	SemTab 2020–2021	2020
Feature engineering based		Limaye et al. [23]	✓	✓	✓			✓	✓	✓			YAGO	Limaye	2010
		Mulwad et al. [75,76]	✓	✓	✓			✓	✓	✓			Wikitology	Limaye	2010
		SemanticTyper [77]		✓					✓				DBpedia	Museum	2015
		DSL [78]		✓					✓				DBpedia	City, Museum, Weather, Custom Soccer	2016
		Neumaier et al. [79]		✓					✓				DBpedia	Government Data Portal	2016
		NUMER [80]		✓					✓		✓		DBpedia	NumDB	2018
Deep learning based	KG modelling	Vasilis et al. [81]	✓					✓	✓				Wikidata	Limaye, T2D, Wikipedia	2017
		Biswas et al. [34]					✓					✓	DBpedia	Custom Wikipedia inforbox	2018
		DAGOBAH Embeddings [82]	✓	✓				✓	✓	✓			DBpedia, Wikidata	SemTab 2019	2019
		Radar Station [83]	✓						✓				Wikidata	Limaye, T2Dv2, SemTab 2020	2022
	Table modelling	Sherlock [84]		✓					✓				DBpedia	T2D, VizNet	2019
		Sato [85]		✓					✓		✓		DBpedia	VizNet	2019
		ColNet [38]		✓				✓	✓				DBpedia	Limaye, T2Dv2	2019
		Guo et al. [86]		✓					✓			✓	DBpedia	T2Dv2	2020
		Zhang et al. [13]	✓		✓			✓	✓				DBpedia	T2Dv2	2020
		TURL [35]	✓	✓	✓			✓	✓	✓	✓	✓	DBpedia	WikiGS, WikiTable, T2D	2020
		TCN [87]		✓	✓			✓	✓		✓	✓	–	Custom Web Tables, WikiTable [35]	2021
		DUDUO [88]		✓	✓			✓	✓		✓		–	WikiTable, VizNet	2021
		Singh et al. [89]			✓				✓	✓		✓	DBpedia	T2Dv2	2021
		Zhou et al [90]		✓					✓		✓		DBpedia	Custom Wikipedia Tables	2021


(a)

Taller structures

The Eiffel Tower was the world's tallest structure when completed in 1889, a distinction it retained until 1929 when the Chrysler Building in New York City was topped out.[107] The tower also lost its standing as the world's tallest tower to the Tokyo Tower in 1958 but retains its status as the tallest freestanding (non-guyed) structure in France.

Lattice towers taller than the Eiffel Tower

Further information: List of tallest towers in the world, Lattice tower, and Observation deck

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	

(b)

```
<tbody><tr>
<th>Name
</th>
<th>Pinnacle height
</th>
<th>Year
</th>
<th>Country
</th>
<th>Town
</th>
<th>Remarks
</th></tr>
<tr>
<td><a href="/wiki/Tokyo_Skytree" title="Tokyo Skytree">Tokyo Skytree</a></td>
<td>634&#160;m (2,080&#160;ft)</td>
<td>2011</td>
<td>Japan</td>
<td>Tokyo</td>
<td>
</td></tr>
<tr>
<td><a href="/wiki/Kyiv_TV_Tower" title="Kyiv TV Tower">Kyiv TV Tower</a></td>
<td>385&#160;m (1,263&#160;ft)</td>
<td>1973</td>
<td>Ukraine</td>
<td>Kyiv</td>
<td>
</td></tr>
<tr>
<td><a href="/wiki/Dragon_Tower" title="Dragon Tower">Dragon Tower</a></td>
<td>336&#160;m (1,102&#160;ft)</td>
<td>2000</td>
<td>China</td>
<td>Harbin</td>
<td>
</td></tr>
<tr>
<td><a href="/wiki/Tokyo_Tower" title="Tokyo Tower">Tokyo Tower</a></td>
<td>333&#160;m (1,093&#160;ft)</td>
<td>1958</td>
<td>Japan</td>
<td>Tokyo</td>
<td>
</td></tr>
<tr>
<td><a href="/wiki/WITI_TV_Tower" title="WITI TV Tower">WITI TV Tower</a></td>
<td>329.4&#160;m (1,081&#160;ft)</td>
<td>1962</td>
<td>United States</td>
<td>Shorewood, Wisconsin</td>
<td>
</td></tr>
<tr>
```

(c)

- 13:21, 8 April 2009 (diff | hist) . . (−188) . . m Ed Simmons *(Reverted edits by 96.25.55.43 (talk) to last version by ClueBot)*
- 18:15, 27 March 2009 (diff | hist) . . (−141) . . Pentaho *(→Projects: removing spam magnet)*
- 18:15, 27 March 2009 (diff | hist) . . (−42) . . m Mondrian OLAP server *(removing advertising link)*
- 16:14, 19 March 2009 (diff | hist) . . (+347) . . User talk:Andre Engels *(→Rfd nom)*
- 11:06, 19 March 2009 (diff | hist) . . (+8) . . m User talk:Carlossuarez46 *(→RedirectName listed at RfD)*
- 11:05, 19 March 2009 (diff | hist) . . (+481) . . User talk:Carlossuarez46 *(RFDNote)*
- 11:05, 19 March 2009 (diff | hist) . . (+241) . . Wikipedia:Redirects for discussion/Log/2009 March 19 *(Nominating Talk:Neftchala)*
- 11:03, 19 March 2009 (diff | hist) . . (+8) . . Talk:Neftchala *({{rfd}})*
- 11:00, 19 March 2009 (diff | hist) . . (+102) . . Neftchala *(redirect changed to disambiguation page)*
- 14:28, 16 March 2009 (diff | hist) . . (−54) . . m Naturalism (arts) *(reverted Robbot and myself)*

Fig. 8. Metadata of the Web table "Lattice towers taller than the Eiffel Tower"[15]: (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.[16]

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 as well the specificity of the annotations. Regarding vertical KGs, the lack of knowledge from other domains can lead to a reduced system generalization. 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.

4. Annotation tasks and pipeline

The previous section introduced the type of tables to annotate and the KGs generally used for the annotation. In this section, we focus on the STI process. We first present five STI tasks in Section 4.1. Next, we describe a common pipeline to perform STI tasks in Section 4.2.

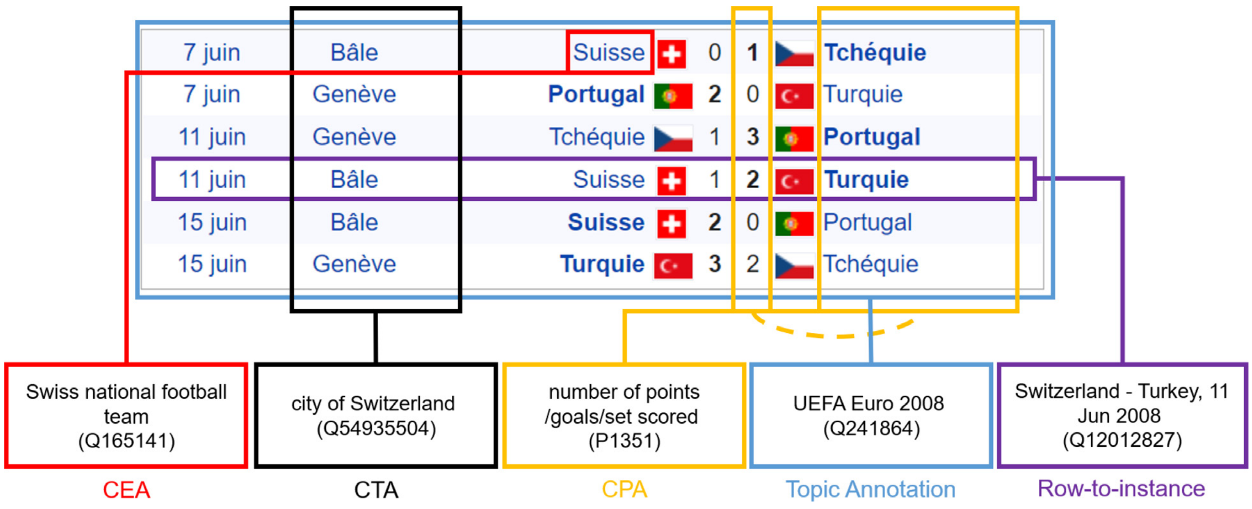
4.1. Annotation tasks

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). We propose to decompose STI into five main tasks: cell-entity annotation, column-type annotation, columns-property annotation [2], topic annotation, and row-to-instance [91] (illustrated in Fig. 9).

Cell-Entity Annotation (CEA) is also known as Entity Linking. It aims to annotate a cell with a KG entity. For example, in Fig. 9, 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 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 Fig. 9 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 Fig. 9, 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

15 https://en.wikipedia.org/wiki/Eiffel_Tower.

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



7 juin	Bâle	Suisse	0	**1**	**Tchéquie**
7 juin	Genève	**Portugal**	**2**	0	Turquie
11 juin	Genève	Tchéquie	1	**3**	**Portugal**
11 juin	Bâle	Suisse	1	**2**	**Turquie**
15 juin	Bâle	**Suisse**	**2**	0	Portugal
15 juin	Genève	**Turquie**	**3**	2	Tchéquie

Swiss national football team (Q165141)
city of Switzerland (Q54935504)
number of points /goals/set scored (P1351)
UEFA Euro 2008 (Q241864)
Switzerland - Turkey, 11 Jun 2008 (Q12012827)
CEA
CTA
CPA
Topic Annotation
Row-to-instance

Fig. 9. Illustration of five STI tasks for a table describing the UEFA Euro 2008 group A results.[17] (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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 Fig. 9 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. Fig. 9 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., on hidden-subject tables). For example, in Fig. 9, the fourth row is represented by ("Switzerland - Turkey, 11 Jun 2008" (Q12012827)) which cannot be extracted by the CEA task.

4.2. Annotation 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 (Section 4.2.1), candidate generation (Section 4.2.2), table elements processing (Section 4.2.3), and iterative disambiguation (Section 4.2.4). 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 [11,17].

4.2.1. 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., 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 [64,66]. 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) [18,30], its orientation [82,92], primitive types for its columns (resp. rows), its header positions, if any, and its key column position which carries the row's subject [82]. 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).

4.2.2. 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, [69] uses Levenshtein-based distances whereas some public knowledge bases expose a public index allowing users to extract and generate candidates, e.g., DBpedia spotlight [48].

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

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

the CEA candidates' types [61,69]. However, in some learning-based methods [77,85], 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 [56,63]. 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, [61,66] leverage the BM25 [93] weights from an Elasticsearch index, that can help to select the candidates with the highest usage rate.

4.2.3. 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 components of candidate confidence [61,69]. For example, in Fig. 9, 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 Fig. 9, 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 Fig. 9, 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 Fig. 7(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 Fig. 8(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 [35]. Another way to enrich the information used for the disambiguation of the target table relies on inter-table relationships [87].

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 [35,88].

4.2.4. Iterative disambiguation

When an STI system jointly undertakes multiple tasks among the five tasks defined in Section 4.1, 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 [66]. We call this process iterative disambiguation. This iterative technique is frequently used in heuristic-based approaches (Section 5.1.2) 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., [66] 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 [11,17].

5. Semantic table interpretation approaches

In this section, 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 5.1) which mostly rely on heuristic techniques such as entity matching, TF–IDF, majority voting, or simple probabilistic frameworks to predict a target; (2) feature-engineering based methods (Section 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 5.3) that leverage a large number of tables and neural networks to learn deep and contextualized representations of elements of the table, requiring little feature engineering. More details on this classification are shown in Table 1 including representative algorithms, target tasks, table elements used, reference KG, and year of publication. Finally, 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 5.4).

5.1. Heuristic 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 [37,55,61], majority voting [60], TF–IDF [55,56] or probabilistic frameworks [63]. The context of the table, including the header, the title, and the neighbouring cells [37,69] is also taken into account but not thoroughly. We further identify two subclasses of heuristic approaches: lookup based approaches (Section 5.1.1) and iterative approaches (Section 5.1.2).

5.1.1. 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. [51] 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 [94]. 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. [37] 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 [95] 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. [52] 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. [53] leverage NELL [96], a Web text corpus, and natural language patterns (PATTY [97]) 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 $p_1, \ldots, p_k$ 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 $p_1, \ldots, p_k$ in PATTY. The problem then comes down to computing the posterior probability of r given evidences $p_1, \ldots, p_k$: $Pr(r|p_1, \ldots, p_k)$. A Bayesian framework is used to compute this posterior.

TabEL [54] 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 [55] 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 [18] to load DBpedia and index its components. The frequency of classes and properties is then used to obtain the CTA and CPA results.

Tabularisi [56] 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 [57] 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 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 [58] 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. [59] 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.

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

5.1.2. 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 4.2.4, play a significant role in the improvement of the model performance.

Zwicklbauer et al. [60] 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 [98]. 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 [11] 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 [99] and the following work **TableMiner+** [17], 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 Web pages. 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 [61] 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 [93]. The system uses a similar approach as [60] 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** [62]. 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 [63] employs four different lookup services. This approach analyses signals from server's lookup ranking, header context analyses, SpaCy type prediction,[19] Duckling type prediction.[20] 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 [100] 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 [64,65] 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 2020 challenges.

LinkingPark [66] 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 [67–69] 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 [68] enhances the system with CTA disambiguation. 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 challenge. DAGOBAH SL [67] also uses language models to extract more meaning from the table headers and largely expands its aliases table using external sources for improving the lookup coverage. The system won again the first prize in accuracy in SemTab 2022 challenge.

MantisTable [70,101] 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** [71] optimizes the system by updating the scoring function, accessing the LamAPI[21] 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.

JenTab [72–74] 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

19 https://spacy.io/.

20 https://github.com/facebook/duckling.

21 https://bitbucket.org/disco-unimib/lamapi.

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.

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 [75], Random Forest [78] and K-Nearest Neighbour [79] 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. [23] 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. [75] leverage majority voting on the type of cells candidates for the CTA annotation. 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 [77] 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. [79] 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 min. 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** [80] proposes to link subject cells with KG entities first, and then only extract the linked properties for enabling the column-wise number distribution study.

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 5.3.1) and table modelling (Section 5.3.2).

5.3.1. KG modelling

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 [102], TransH [103]) 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 [81] or Euclidean distance [82].

Vasilis et al. [81] 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. [34] 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 [104] 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 Embeddings [82] 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.

Radar Station [83] went a step further in proposing a hybrid system that aims to add a semantic disambiguation step after a previously identified CEA. Radar Station takes into account the entire column as context and uses graph embeddings to capture latent relationships between entities to improve their disambiguation. 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.

5.3.2. Table modelling

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 [38,84] or language models like BERT [35,88,90].

Sherlock [84] learns to perform the CTA task using 1588 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** [85] 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 [38] 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.** [86] 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. [13] 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 [105]) 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 the 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 [35] 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 [106] 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. [89] 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 [87] 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 [88], 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 utilization of inter-table context remains challenging since it requires prior knowledge about tables' schemas which are generally diverse.

DODUO [88] 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. [90] focus on the column type detection (CTA) task. This work leverages the Star-Transformer model [107] 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 [85] 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.

5.4. 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).

5.4.1. 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 Embeddings [82] and ColNet [38] 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. [35,77,78,84,87,88] predict the CTA within a predefined set of around one hundred types. ColNet [38] 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, [77,78,84] 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 [77,78,88].

5.4.2. 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 [84] is based on 1588 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. [81]) and table modelling with BERT-like models (e.g., TCN [87]). 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 [87,88,90] try to simplify the table representation to a collection of unordered lists for columns or rows, ignoring their index and other structural information. TURL [35] 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.

5.4.3. Coverage of table elements

Annotation models use different table elements that are analysed in depth in Section 4.2.3. We observe that more and more table elements are considered in recent approaches. This phenomenon is characteristic of learning-based models. [77–79,84] primarily concentrate on extracting features from a single column. With the rise of the deep learning and attention mechanism, [35,85,87,88] start to pay more attention to other complex table elements. A typical example is TURL [35], in which the authors consider all of the six table elements discussed in Section 4.2.3. However, we cannot 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.

5.4.4. *Effectiveness vs efficiency*

Annotation systems usually deal with a trade-off between effectiveness and efficiency. TableMiner+ [17] 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., [63]) 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.

5.4.5. *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 [35,77,78,84,87,88] 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 [88] 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, [51] built an isA database using Web documents which contains three times more types than Freebase.

6. Datasets and benchmarks

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 2) that are used by STI approaches.

Limaye [23] 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 [76] which were corrected by [54]'s work in 2015. Later on, [81] updated the disambiguation links to the DBpedia KG.

T2D [108] is taken from the Web Data Commons project.[22] 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 [109]. T2D and its supplementary version T2Dv2 [109] have been largely used to evaluate approaches in [17,38,84], and together with Limaye, they became the de facto gold standard datasets to use for evaluating STI approaches. However, [110] 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`. [110] has proposed a revised version of the dataset named T2D*.

WDC [30] leverages the DWTC framework [111] 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 3.1. 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 [54] (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** [112] further enhanced the dataset by merging TabEL with 2.6 million tables from the TableL [113] dataset that were extracted from 1.5M Common Crawl Web pages using the DWTC framework [111]. The TableL dataset covers mostly five major topics: finance, environment, health, politics, and sports. These datasets have generally been used to train machine learning based STI approaches [54].

Zhang et al. [17] proposed datasets[23] 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[24]) colocated with the International Semantic Web Conference in 2019, 2020 and 2021 provides the biggest datasets. This competition has attracted nearly 50 participant teams over the three years. The SemTab 2019 datasets [2] use DBpedia as the target KG, while SemTab 2020 [3] uses Wikidata and SemTab 2021 [1] 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. 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 [114] and BiodivTab [115], 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

[22] http://www.webdatacommons.org/webtables/.

[23] https://github.com/ziqizhang/data/tree/master/webtable%20entity%20linking.

[24] http://www.cs.ox.ac.uk/isg/challenges/sem-tab/.

Table 2
Gold standard datasets for evaluating STI approaches. Table type refers to the classification introduced in Section 3.1 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	6085	20	2	131,807	–	–	Web	Wikipedia	2010
T2D[b]	T2D[c]	SR-V	762	157	5	25,119	7983	–	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	8418	120	116	Web	DBpedia	2019
	R2	SR-H	11,924	25	5	463,796	14,780	6762	Web	DBpedia	2019
	R3	SR-H	2161	71	5	406,827	14,780	7575	Synthetic	DBpedia	2019
	R4	SR-H	713	63	4	107,352	1732	2747	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	1080	5	663,830	539	–	Synthetic	Wikidata	2020
SemTab 2021	R1-DBP	SR-H	180	1081	4	663,656	540	–	Synthetic	DBpedia	2021
	R1-WD	SR-H	180	1081	4	667,244	540	–	Synthetic	Wikidata	2021
	R2-Bio	SR-H	110	2449	6	1,381,325	657	547	Synthetic	Wikidata	2021
	R2-Hard	SR-H	1750	17	3	47,440	2191	3836	Synthetic	Wikidata	2021
	R3-Hard	SR-H	7207	9	2	58,949	7207	10,695	Synthetic	Wikidata	2021
	R3-Git	SR-H	1101	59	17	–	123/60	–	Github	DBpedia, Schema.org	2021
	R3-BiodivTab	SR-H	50	260	24	31,468	614	–	Open data	Wikidata	2021

[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/.

explains the very high accuracy of the top-performing approaches (F1 score up to 0,99 for some tasks [3]). 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 [116]. Tough Tables is composed of specially designed 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, the organizers bring new challenges with, in particular, the BiodivTab [115] 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 [114] 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.

7. Evaluation

Traditionally, STI systems are evaluated using the information retrieval metrics: accuracy, recall, and F1 scores [2,17,35,78,82]. 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 (Eq. (1)) and AP (Eq. (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 [38]. The SemTab Challenge evaluates systems using the AIcrowd evaluator[25] and STILTool [117].

$$AH = \frac{\#\text{PerfectA} + 0.5 \times \#\text{OkayA} - \#\text{WrongA}}{\#\text{TargetColumns}} \tag{1}$$

$$AP = \frac{\#\text{PerfectA}}{\#\text{AnnotatedColumns}} \tag{2}$$

SemTab 2020 and 2021 further enhance the CTA evaluation metrics by introducing the correctness score *cscore* (Eq. (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.

$$cscore(\alpha) = \begin{cases} 0.8^{d(\alpha)}, & \text{if } \alpha \text{ is an ancestor of the GT,} \\ 0.7^{d(\alpha)}, & \text{if } \alpha \text{ is a descendant of the GT,} \\ 0, & \text{otherwise;} \end{cases} \tag{3}$$

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

$$AP = \frac{\sum cscore(\alpha)}{\#\text{Annotations}}, \quad AR = \frac{\sum cscore(\alpha)}{\#\text{Targets}} \tag{4}$$

$$AF1 = \frac{2 \times AP \times AR}{AP + AR} \tag{5}$$

Table 3 gives the performance of the top three systems with the highest F1, AP, or AF1 scores for the CEA, CTA and CPA tasks

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

Table 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] 0.894	TabEAno[d] [118] 0.88	T2K ++ 0.87	T2K ++ 0.88	Guo et al 0.852	MantisTable 0.84	Mulwad et al. 0.89	T2K ++ 0.80	TableMiner+ 0.76
T2D		TabEAno 0.91	Zhang et al. 0.90	Kruit et al. 0.89	ColNet[e] 0.976	Alobaid et al. [59] 0.96	MantisTable 0.95	T2K ++ 0.91	Singh et al. 0.71	MantisTable 0.51
SemTab 2019	R2	MTab 0.911	CSV2KG 0.883	Tabularisi 0.826	MTab 1.414	CSV2KG 1.376	Tabularisi 1.099	CSV2KG 0.881	IDLab 0.877	Tabularisi 0.790
	R3	MTab 0.970	CSV2KG 0.962	ADOG 0.912	MTab 1.956	CSV2KG 1.864	Tabularisi 1.702	MTab 0.844	CSV2KG 0.841	Tabularisi 0.827
	R4	MTab 0.983	MantisTable 0.973	CSV2KG 0.907	MTab 2.012	CSV2KG 1.846	Tabularisi 1.716	MTab 0.832	CSV2KG 0.830	Tabularisi 0.823
SemTab 2020	R1	MTab 0.987	LinkingPark 0.987	MantisTable 0.982	JenTab 0.962	LinkingPark 0.926	MTab 0.885	MTab 0.971	LinkingPark 0.967	JenTab 0.963
	R2	MTab 0.995	DAGOBAH[f] 0.993	LinkingPark 0.993	LinkingPark 0.984	MTab 0.984	DAGOBAH 0.983	MTab 0.997	LinkingPark 0.993	DAGOBAH 0.992
	R3	MTab 0.991	LinkingPark 0.986	DAGOBAH 0.985	LinkingPark 0.978	MTab 0.976	DAGOBAH 0.974	MTab 0.995	DAGOBAH 0.993	bbw [119] 0.989
	R4	MTab 0.993	LinkingPark 0.985	DAGOBAH 0.984	MTab 0.981	bbw 0.98	DAGOBAH 0.972	MTab 0.997	bbw 0.995	DAGOBAH 0.995
	2T	MTab 0.907	bbw 0.863	DAGOBAH 0.830	DAGOBAH 0.743	MTab 0.728	LinkingPark 0.686	– –	– –	– –
SemTab 2021	R1 (DBpedia)	DAGOBAH 0.945	GBMTab 0.692	JenTab 0.607	JenTab 0.46	DAGOBAH 0.422	Magic 0.159	– –	– –	– –
	R1 (WikiData)	DAGOBAH 0.923	MTab 0.907	AMALGAM [120] 0.658	DAGOBAH 0.832	MTab 0.728	JenTab 0.697	– –	– –	– –
	R2-Hard	MTab 0.985	DAGOBAH 0.975	MantisTable 0.968	MTab 0.977	DAGOBAH 0.976	MantisTable 0.955	MTab 0.997	JenTab 0.996	DAGOBAH 0.996
	R2-Bio	DAGOBAH 0.970	MTab 0.964	MantisTable 0.93	MTab 0.956	Magic 0.916	DAGOBAH 0.916	MTab 0.947	DAGOBAH 0.899	JenTab 0.899
	R3-Biodiv	JenTab 0.602	MTab 0.522	DAGOBAH 0.496	KEPLER-aSI [121] 0.593	DAGOBAH 0.391	MTab 0.123	– –	– –	– –
	R3-Hard	DAGOBAH 0.974	MTab 0.968	MantisTable 0.959	DAGOBAH 0.99	MTab 0.984	MantisTable 0.965	MTab 0.993	JenTab 0.992	DAGOBAH 0.991
	R3-Git (DBp)	– –	– –	– –	DAGOBAH 0.07	KEPLER-aSI 0.041	MantisTable 0.037	– –	– –	– –
	R3-Git (Sch)	– –	– –	– –	MantisTable 0.205	DAGOBAH 0.183	– –	– –	– –	– –

[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]TabEAno is a sub system of MTab.
[e]ColNet was evaluated on T2Dv2. Thus, we consider only the result from 237 PK columns, which is almost the 233 tables from T2D.
[f]This method is named DAGOBAH SL in [67–69].

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) [57] 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 [38,57] is about 60%, probably because [57] 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: (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 [54] and Viznet [122]) 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. [86], 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. [86] have been able to become one of the top three systems for real-world datasets like Limaye and T2D.

8. Challenges and future directions

While recent works have made significant progress in the field of STI, existing approaches have several limitations: (i) they mainly focus on single-cell subject 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. Based on these observations, we formulate some possible guidelines to sketch the future directions for the STI field.

8.1. 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 [34,123] but the other types of tables defined in the classification we proposed in Section 3.1 are still hardly handled. Moreover, current approaches do not dig deeper into relational tables complexities such as hidden subjects or composed subjects, to name a few.

As a result, existing systems are far from being generalizable 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 recommend creating new datasets with multiple table structures and complex contents to tackle the whole diversity of real-world data. We advise 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 prioritizing quality over quantity for evaluation datasets to bootstrap new challenges quickly.

8.2. KG incompleteness and incorrectness

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 emphasize that future approaches should 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, 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 6). 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 argue that annotation systems should 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.

8.3. Table context

We observe that many approaches leverage only partially the elements of the table (see Table 1), even if more recent ones tend to extend their view. As we discussed in Section 5.4.3, 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 [35] 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 related to keys such as SQL database tables. Inter-table relations as studied by [87] 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.

9. Conclusion

The last few years have seen significant growth in the field of STI, with the development of many new approaches and the proposal of ever more complete datasets, notably under the impulse of initiatives such as the SemTab challenge. In this survey, we have provided a comprehensive and up-to-date overview of the STI field. Our work also gathers and proposes a set of definitions to structure and unify the field. It first defines the inputs (the different types of tables, their context and their metadata) of the STI process as well as the KGs used both as a support and as a repository to be enriched via the STI process. Then, we propose a generic pipeline for STI and a description of the different tasks performed by existing approaches. These are classified into three families, heuristics, feature engineering and deep learning, with an emphasis on the strengths and weaknesses of each.

We have also summarized what are the performances of the various approaches on the datasets that have been proposed in the community. We have highlighted the best performing systems for each dataset. Finally, we have listed several challenges to address for improving STI systems. We observe that recent works have tried to develop modern web-based user interfaces on top of STI systems such as [124] which will undoubtedly empower end-users when adopting these systems.

CRediT authorship contribution statement

Jixiong Liu: Methodology, Investigation, Writing – review & editing. **Yoan Chabot:** Methodology, Supervision, Writing – review & editing. **Raphaël Troncy:** Methodology, Supervision, Writing – review & editing. **Viet-Phi Huynh:** Investigation, Resources, Writing – review & editing. **Thomas Labbé:** Writing – review & editing. **Pierre Monnin:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] V. Cutrona, J. Chen, V. Efthymiou, O. Hassanzadeh, E. Jiménez-Ruiz, J. Sequeda, K. Srinivas, N. Abdelmageed, M. Hulsebos, D. Oliveira10, et al., Results of SemTab 2021, in: CEUR Workshop Proceedings, 2021.
[2] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, SemTab 2019: Resources to benchmark tabular data to knowledge graph matching systems, in: European Semantic Web Conference, ESWC, Springer, 2020, pp. 514–530.
[3] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, V. Cutrona, Results of SemTab 2020, in: CEUR Workshop Proceedings, Vol. 2775, 2020, pp. 1–8.
[4] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, DBpedia-A crystallization point for the Web of Data, J. Web Semant. 7 (3) (2009) 154–165.
[5] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledge base, Commun. ACM 57 (10) (2014) 78–85.
[6] C.S. Bhagavatula, T. Noraset, D. Downey, Methods for exploring and mining tables on wikipedia, in: ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, 2013, pp. 18–26.
[7] M.J. Cafarella, A. Halevy, D.Z. Wang, E. Wu, Y. Zhang, Webtables: exploring the power of tables on the web, in: VLDB Endowment, VLDB Endowment, 2008, pp. 538–549.
[8] P. Pasupat, P. Liang, Compositional semantic parsing on semi-structured tables, 2015, arXiv:1508.00305.
[9] H. Sun, H. Ma, X. He, W.-t. Yih, Y. Su, X. Yan, Table cell search for question answering, in: 25th International Conference on World Wide Web, 2016, pp. 771–782.
[10] P. Yin, G. Neubig, W.-t. Yih, S. Riedel, TaBERT: Pretraining for joint understanding of textual and tabular data, 2020.
[11] D. Ritze, O. Lehmberg, C. Bizer, Matching html tables to dbpedia, in: 5th International Conference on Web Intelligence, Mining and Semantics, 2015, pp. 1–6.
[12] M. Yakout, K. Ganjam, K. Chakrabarti, S. Chaudhuri, Infogather: entity augmentation and attribute discovery by holistic matching with web tables, in: ACM SIGMOD International Conference on Management of Data, 2012, pp. 97–108.
[13] S. Zhang, E. Meij, K. Balog, R. Reinanda, Novel entity discovery from web tables, in: The Web Conference, 2020, pp. 1298–1308.
[14] S. Zhang, K. Balog, Recommending related tables, 2019, arXiv:1907.03595.
[15] O. Benjelloun, S. Chen, N. Noy, Google dataset search by the numbers, in: International Semantic Web Conference (ISWC), in-Use Track, 2020.
[16] N. Noy, M. Burgess, D. Brickley, Google dataset search: Building a search engine for datasets in an open web ecosystem, in: 28th the Web Conference, WWW, 2019.
[17] Z. Zhang, Effective and efficient semantic table interpretation using tableminer+, Semant. Web 8 (6) (2017) 921–957.
[18] S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation: A survey, ACM Trans. Intell. Syst. Technol. 11 (2) (2020) 1–35.
[19] S. Bonfitto, E. Casiraghi, M. Mesiti, Table understanding approaches for extracting knowledge from heterogeneous tables, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. (2021) e1407.
[20] G. Badaro, M. Saeed, P. Papotti, Transformers for Tabular Data Representation: A Survey of Models and Applications, Tech. rep., EURECOM, 2021, https://www.eurecom.fr/publication/6721.
[21] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, G. Kasneci, Deep neural networks and tabular data: A survey, 2021, arXiv:2110.01889.
[22] J.L. Martinez-Rodriguez, A. Hogan, I. Lopez-Arevalo, Information extraction meets the semantic web: a survey, Semant. Web J. 11 (2) (2020) 255–235.
[23] G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, Proc. VLDB Endow. 3 (1–2) (2010) 1338–1347.
[24] G. Penn, J. Hu, H. Luo, R. McDonald, Flexible web document analysis for delivery to narrow-bandwidth devices, in: 6th International Conference on Document Analysis and Recognition, IEEE, 2001, pp. 1074–1078.
[25] Y. Wang, J. Hu, Detecting tables in HTML documents, in: 5th IAPR International Workshop on Document Analysis Systems, Springer, 2002, pp. 249–260.
[26] D. Ritze, Web-Scale Web Table to Knowledge Base Matching (Ph.D. thesis), University of Mannheim, 2017.
[27] E. Crestan, P. Pantel, Web-scale table census and classification, in: 4th ACM International Conference on Web Search and Data Mining, WSDM, 2011, pp. 545–554.
[28] L.R. Lautert, M.M. Scheidt, C.F. Dorneles, Web table taxonomy and formalization, ACM SIGMOD Rec. 42 (3) (2013) 28–33.
[29] J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, W. Lehner, Building the dresden web table corpus: A classification approach, in: IEEE 2nd International Symposium on Big Data Computing, BDC, IEEE, 2015, pp. 41–50.

[30] O. Lehmberg, D. Ritze, R. Meusel, C. Bizer, A large public corpus of web tables containing time and context metadata, in: 25th International Conference Companion on World Wide Web, 2016, pp. 75–76.
[31] Z. Wang, H. Dong, R. Jia, J. Li, Z. Fu, S. Han, D. Zhang, TUTA: Tree-based transformers for generally structured table pre-training, in: 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1780–1790.
[32] J. Riley, Understanding Metadata, National Information Standards Organization, Washington DC, United States, 2017, http://www.niso.org/publications/press/UnderstandingMetadata.pdf.
[33] A. Zhang, D. Gourley, Creating Digital Collections: A Practical Guide, Elsevier, 2008.
[34] R. Biswas, R. Türker, F.B. Moghaddam, M. Koutraki, H. Sack, Wikipedia infobox type prediction using embeddings, in: DL4KGS@ ESWC, 2018, pp. 46–55.
[35] X. Deng, H. Sun, A. Lees, Y. Wu, C. Yu, TURL: Table understanding through representation learning, 2020, arXiv:2006.14806.
[36] C. Tao, D.W. Embley, Automatic hidden-web table interpretation, conceptualization, and semantic annotation, Data Knowl. Eng. 68 (7) (2009) 683–703.
[37] J. Wang, H. Wang, Z. Wang, K.Q. Zhu, Understanding tables on the web, in: International Conference on Conceptual Modeling, Springer, 2012, pp. 141–155.
[38] J. Chen, E. Jimenez-Ruiz, I. Horrocks, C. Sutton, ColNet: Embedding the semantics of web tables for column type prediction, in: 33rd AAAI International Conference on Artificial Intelligence, 2018.
[39] L. Ehrlinger, W. Wöß, Towards a definition of knowledge graphs, SEMANTiCS 48 (1–4) (2016) 2.
[40] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J.E.L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, et al., Knowledge graphs, 2020, arXiv:2003.02320.
[41] T. Berners-Lee, Linked data - design issues, 2006, http://www.w3.org/DesignIssues/LinkedData.html.
[42] L. Xiaoxue, B. Xuesong, W. Longhe, R. Bingyuan, L. Shuhan, L. Lin, Review and trend analysis of knowledge graphs for crop pest and diseases, IEEE Access 7 (2019) 62251–62264.
[43] P.K. Thornton, A. Stroud, N. Hatibu, C. Legg, S. Ly, S. Twomlow, K. Molapong, A. Notenbaert, R. Kruska, R. von Kaufmann, Site selection to test an integrated approach to agricultural research for development: combining expert knowledge and participatory Geographic Information System methods, Int. J. Agric. Sustain. 4 (1) (2006) 39–60.
[44] M.M. Yusof, N.F. Rosli, M. Othman, R. Mohamed, M.H.A. Abdullah, M-DCocoa: M-agriculture expert system for diagnosing cocoa plant diseases, in: International Conference on Soft Computing and Data Mining, Springer, 2018, pp. 363–371.
[45] W. Van Eeden, J.P. De Villiers, R. Berndt, W.A. Nel, E. Blasch, Micro-Doppler radar classification of humans and animals in an operational environment, Expert Syst. Appl. 102 (2018) 1–11.
[46] F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault, J. Morissette, Bio2RDF: towards a mashup to build bioinformatics knowledge systems, J. Biomed. Inform. 41 (5) (2008) 706–716.
[47] C. Bizer, T. Heath, T. Berners-Lee, Linked data: The story so far, in: International Journal on Semantic Web and Information Systems, IGI global, 2009, pp. 205–227.
[48] P.N. Mendes, M. Jakob, A. García-Silva, C. Bizer, DBpedia spotlight: shedding light on the web of documents, in: 7th International Conference on Semantic Systems, 2011, pp. 1–8.
[49] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: ACM SIGMOD International Conference on Management of Data, 2008.
[50] T.P. Tanon, G. Weikum, F. Suchanek, YAGO 4: A reason-able knowledge base, in: European Semantic Web Conference, ESWC, Springer, 2020, pp. 583–596.
[51] P. Venetis, A.Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, Recovering semantics of tables on the web, PVLDB 4 (9) (2011) 528–538.
[52] D. Deng, Y. Jiang, G. Li, J. Li, C. Yu, Scalable column concept determination for web tables using large knowledge bases, in: PVLDB, VLDB Endowment, 2013, pp. 1606–1617.
[53] Y.A. Sekhavat, F. Di Paolo, D. Barbosa, P. Merialdo, Knowledge base augmentation using tabular data, in: LDOW, 2014.
[54] C.S. Bhagavatula, T. Noraset, D. Downey, Tabel: Entity linking in web tables, in: 14th International Semantic Web Conference, Springer, 2015, pp. 425–441.
[55] D. Oliveira, M. d'Aquin, Adog-annotating data with ontologies and graphs, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2019.
[56] A. Thawani, M. Hu, E. Hu, H. Zafar, N.T. Divvala, A. Singh, E. Qasemi, P.A. Szekely, J. Pujara, Entity linking to knowledge graphs to infer column types and properties, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), Vol. 2019, 2019, pp. 25–32.
[57] U. Khurana, S. Galhotra, Semantic annotation for tabular data, 2019.
[58] B. Steenwinckel, F. De Turck, F. 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.
[59] A. Alobaid, O. Corcho, Balancing coverage and specificity for semantic labelling of subject columns, Knowl.-Based Syst. (2022) 108092.
[60] S. Zwicklbauer, C. Einsiedler, M. Granitzer, C. Seifert, Towards disambiguating web tables, in: International Semantic Web Conference (Posters & Demos), 2013, pp. 205–208.
[61] H. Morikawa, Semantic table interpretation using LOD4ALL, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2019, pp. 49–56.
[62] B. Steenwinckel, G. Vandewiele, F. De Turck, F. Ongenae, Csv2kg: Transforming tabular data into semantic knowledge, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2019.
[63] P. Nguyen, N. Kertkeidkachorn, R. Ichise, H. Takeda, MTab: Matching tabular data to knowledge graph using probability models, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2019.
[64] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, Mtab4wikidata at semtab 2020: Tabular data annotation with wikidata, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2020.
[65] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, SemTab 2021: Tabular data annotation with MTab tool, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2021.
[66] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J.-G. Yao, J. Williams, A. Gordon, C.-Y. Lin, Linkingpark: An integrated approach for semantic table interpretation, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2020.
[67] V.-P. Huynh, Y. Chabot, T. Labbé, J. Liu, R. 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.
[68] V.-P. Huynh, J. Liu, Y. Chabot, F. Deuzé, T. Labbé, P. Monnin, R. 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.
[69] V.-P. Huynh, J. Liu, Y. Chabot, T. Labbé, P. Monnin, R. Troncy, DAGOBAH: Enhanced scoring algorithms for scalable annotations of tabular data, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2020.
[70] M. Cremaschi, R. Avogadro, D. Chieregato, MantisTable: an automatic approach for the semantic table interpretation, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2019, pp. 15–24.
[71] M. Cremaschi, R. Avogadro, A. Barazzetti, D. Chieregato, MantisTable SE: an efficient approach for the semantic table interpretation, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2020.
[72] N. Abdelmageed, S. Schindler, JenTab: Matching tabular data to knowledge graphs, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2020, pp. 40–49.
[73] N. Abdelmageed, S. Schindler, JenTab: A toolkit for semantic table annotations, in: 2nd International Workshop on Knowledge Graph Construction, 2021.
[74] N. Abdelmageed, S. Schindler, JenTab meets SemTab 2021's new challenges, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2021.
[75] V. Mulwad, T. Finin, Z. Syed, A. Joshi, et al., Using linked data to interpret tables, in: 1st International Workshop on Consuming Linked Data, COLD, 2010.
[76] V. Mulwad, T. Finin, A. Joshi, Semantic message passing for generating linked data from tables, in: 12th International Semantic Web Conference, ISWC, Springer, 2013, pp. 363–378.
[77] S.K. Ramnandan, A. Mittal, C.A. Knoblock, P. Szekely, Assigning semantic labels to data sources, in: European Semantic Web Conference, ESWC, Springer, 2015, pp. 403–417.
[78] P. Minh, A. Suresh, A.K. Craig, S. Pedro, Semantic labeling: A domain-independent approach, in: 15th International Semantic Web Conference, ISWC, 2016, pp. 446—462.
[79] S. Neumaier, J. Umbrich, J.X. Parreira, A. Polleres, Multi-level semantic labelling of numerical values, in: 15th International Semantic Web Conference, ISWC, 2016, pp. 428–445.
[80] E. Kacprzak, J.M. Giménez-García, A. Piscopo, L. Koesten, L.-D. Ibáñez, J. Tennison, E. Simperl, Making sense of numerical data-semantic labelling of web tables, in: European Knowledge Acquisition Workshop, Springer, 2018, pp. 163–178.

[81] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, V. Christophides, Matching web tables with knowledge base entities: from entity lookups to entity embeddings, in: 16th International Semantic Web Conference, ISWC, Springer, 2017, pp. 260–277.
[82] Y. Chabot, T. Labbe, J. Liu, R. Troncy, DAGOBAH: an end-to-end context-free tabular data semantic annotation system, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, 2019, pp. 41–48.
[83] J. Liu, V.-P. Huynh, Y. Chabot, R. Troncy, Radar station: Using KG embeddings for semantic table interpretation and entity disambiguation, in: 21st International Semantic Web Conference, ISWC, 2022.
[84] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, C. Hidalgo, Sherlock: A deep learning approach to semantic data type detection, in: 25th ACM International Conference on Knowledge Discovery & Data Mining, SIGKDD, 2019, pp. 1500–1508.
[85] D. Zhang, Y. Suhara, J. Li, M. Hulsebos, Ç. Demiralp, W.-C. Tan, Sato: Contextual semantic type detection in tables, 2019.
[86] T. Guo, D. Shen, T. Nie, Y. Kou, Web table column type detection using deep learning and probability graph model, in: International Conference on Web Information Systems and Applications, Springer, 2020, pp. 401–414.
[87] D. Wang, P. Shiralkar, C. Lockard, B. Huang, X.L. Dong, M. Jiang, TCN: Table convolutional network for web table interpretation, 2021, arXiv:2102.09460.
[88] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, W.-C. Tan, Annotating columns with pre-trained language models, 2021, arXiv:2104.01785.
[89] G. Singh, S. Singh, J. Wong, A. Saffari, Relation extraction from tables using artificially generated metadata, 2021, arXiv:2108.10750.
[90] Y. Zhou, S. Singh, C. Christodoulopoulos, Tabular data concept type detection using star-transformers, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 3677–3681.
[91] D. Ritze, C. Bizer, Matching web tables to DBpedia - A feature utility study, in: International Conference on Extending Database Technology, EDBT, 2017, pp. 210—221.
[92] M. Habibi, J. Starlinger, U. Leser, DeepTable: a permutation invariant neural network for table orientation classification, Data Min. Knowl. Discov. (2020) 1–21.
[93] S. Robertson, H. Zaragoza, The Probabilistic Relevance Framework: BM25 and beyond, Now Publishers Inc, 2009.
[94] M. Banko, O. Etzioni, The tradeoffs between open and traditional relation extraction, in: ACL-08: HLT, 2008, pp. 28–36.
[95] W. Wu, H. Li, H. Wang, K.Q. Zhu, Inferring a Universal Probabilistic Taxonomy from the Web, Tech. rep., Technical report, Microsoft Research, 2010.
[96] P.P. Talukdar, D. Wijaya, T. Mitchell, Acquiring temporal constraints between relations, in: The 21st ACM International Conference on Information and Knowledge Management, 2012, pp. 992–1001.
[97] N. Nakashole, G. Weikum, F. Suchanek, PATTY: A taxonomy of relational patterns with semantic types, in: Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012, pp. 1135–1145.
[98] S. Zwicklbauer, C. Seifert, M. Granitzer, Do we need entity-centric knowledge bases for entity disambiguation? in: 13th International Conference on Knowledge Management and Knowledge Technologies, 2013, pp. 1–8.
[99] Z. Zhang, Towards efficient and effective semantic table interpretation, in: 3th International Semantic Web Conference, Springer, 2014, pp. 487–502.
[100] P. Nguyen, K. Nguyen, R. Ichise, H. Takeda, Embnum: Semantic labeling for numerical values with deep metric learning, in: Joint International Semantic Technology Conference, Springer, 2018, pp. 119–135.
[101] M. Cremaschi, F. De Paoli, A. Rula, B. Spahiu, A fully automated approach to a complete semantic table interpretation, Future Gener. Comput. Syst. 112 (2020) 478–500.
[102] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Adv. Neural Inf. Process. Syst. 26 (2013).
[103] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: AAAI Conference on Artificial Intelligence, 2014.
[104] P. Ristoski, H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in: International Semantic Web Conference, Springer, 2016, pp. 498–514.
[105] J. Guo, Y. Fan, Q. Ai, W.B. Croft, A deep relevance matching model for ad-hoc retrieval, in: The 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 55–64.
[106] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017, arXiv:1706.03762.
[107] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, Z. Zhang, Star-transformer, 2019, arXiv:1902.09113.
[108] D. Ritze, O. Lehmberg, C. Bizer, Matching HTML tables to DBpedia, in: 5th International Conference on Web Intelligence, Mining and Semantics, WIMS, 2015, pp. 1–6.
[109] O. Lehmberg, D. Ritze, R. Meusel, C. Bizer, A large public corpus of web tables containing time and context metadata, in: 25th International Conference Companion on World Wide Web (WWW Companion), 2016, pp. 75–76.
[110] I. Ermilov, A.-C.N. Ngomo, TAIPAN: automatic property mapping for tabular data, in: European Knowledge Acquisition Workshop, Springer, 2016, pp. 163–179.
[111] J. Eberius, M. Thiele, K. Braunschweig, W. Lehner, Top-k entity augmentation using consistent set covering, in: SSDBM, 2015.
[112] V.T. Ho, K. Pal, G. Weikum, QuTE: Answering quantity queries from web tables, in: International Conference on Management of Data, 2021, pp. 2740–2744.
[113] Y. Ibrahim, M. Riedewald, G. Weikum, D. Zeinalipour-Yazti, Bridging quantities in tables and text, in: 2019 IEEE 35th International Conference on Data Engineering, ICDE, IEEE, 2019, pp. 1010–1021.
[114] M. Hulsebos, Ç. Demiralp, P. Groth, GitTables: A large-scale corpus of relational tables, 2021, arXiv:2106.07258.
[115] N. Abdelmageed, S. Schindler, B. König-Ries, BiodivTab: A tabular benchmark based on biodiversity research data, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2021.
[116] V. Cutrona, F. Bianchi, E. Jiménez-Ruiz, M. Palmonari, Tough tables: Carefully evaluating entity linking for tabular data, in: 19th International Semantic Web Conference, ISWC, Springer, 2020, pp. 328–343.
[117] M. Cremaschi, A. Siano, R. Avogadro, E. Jimenez-Ruiz, A. Maurino, STILTool: a semantic table interpretation evaluation tool, in: European Semantic Web Conference, Springer, 2020, pp. 61–66.
[118] P. Nguyen, N. Kertkeidkachorn, R. Ichise, H. Takeda, TabEAno: table to knowledge graph entity annotation, 2020, arXiv:2010.01829.
[119] R. Shigapov, P. Zumstein, J. Kamlah, L. Oberländer, J. Mechnich, I. Schumm, Bbw: Matching CSV to wikidata via meta-lookup, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), Vol. 2775, RWTH, 2020, pp. 17–26.
[120] R. Azzi, G. Diallo, AMALGAM: making tabular dataset explicit with knowledge graph., in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2020, pp. 9–16.
[121] W. Baazouzi, M. Kachroudi, S. Faiz, KEPLER-asi at SemTab 2021, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2021.
[122] K. Hu, S. Gaikwad, M. Hulsebos, M.A. Bakker, E. Zgraggen, C. Hidalgo, T. Kraska, G. Li, A. Satyanarayan, Ç. Demiralp, Viznet: Towards a large-scale visualization learning and benchmarking repository, in: CHI Conference on Human Factors in Computing Systems, 2019, pp. 1–12.
[123] F. Wu, D.S. Weld, Automatically refining the wikipedia infobox ontology, in: 17th International Conference on World Wide Web, 2008, pp. 635–644.
[124] C. Sarthou-Camy, G. Jourdain, Y. Chabot, P. Monnin, Deuzé, V.-P. Huynh, J. Liu, T. Labbé, R. Troncy, DAGOBAH UI: A new hope for semantic table interpretation, in: 19th European Semantic Web Conference (ESWC), Poster and Demo Track, 2022.