Web Table Extraction, Retrieval, and Augmentation: A Survey

SHUO ZHANG and KRISZTIAN BALOG, University of Stavanger

Tables are powerful and popular tools for organizing and manipulating data. A vast number of tables can be found on the Web, which represent a valuable knowledge resource. The objective of this survey is to synthesize and present two decades of research on web tables. In particular, we organize existing literature into six main categories of information access tasks: table extraction, table interpretation, table search, question answering, knowledge base augmentation, and table augmentation. For each of these tasks, we identify and describe seminal approaches, present relevant resources, and point out interdependencies among the different tasks.

CCS Concepts: • **Information systems** → **Information integration**; **Data extraction and integration**; **Retrieval models and ranking**; *Data management systems*; *Information storage systems*;

Additional Key Words and Phrases: Table extraction, table search, table retrieval, table mining, table augmentation, table interpretation

ACM Reference format:
Shuo Zhang and Krisztian Balog. 2020. Web Table Extraction, Retrieval, and Augmentation: A Survey. *ACM Trans. Intell. Syst. Technol.* 11, 2, Article 13 (January 2020), 35 pages.
https://doi.org/10.1145/3372117

1 INTRODUCTION

Tables are practical and useful tools in many application scenarios. Tables can be effectively utilized for collecting and organizing information from multiple sources. With the help of additional operations, such as sorting, filtering, and joins, this information can be turned into knowledge and, ultimately, can be used to support decision-making. Thanks to their convenience and utility, a large number of tables are being produced and are made available on the Web. These tables represent a valuable resource and have been a focus of research for over two decades now. In this survey article, we provide a systematic overview of this body of research.

Tables on the Web, referred to as *web tables* further on in this article, differ from traditional tables (that is, tables in relational databases and tables created in spreadsheet programs) in a number of ways. First, web tables are embedded in webpages. There is a lot of contextual information, such as the embedding page's title and link structure, the surrounding text, and so on, that can be utilized. Second, web tables are rather heterogeneous regarding their quality, organization, and content. For example, tables on the Web are often used for layout and navigation purposes. Among the different table types, *relational tables* (also referred to as *genuine tables*) are of special interest. These describe

Authors' addresses: S. Zhang and K. Balog, Dept. of Electrical Engineering and Computer Science, University of Stavanger, NO-4036 Stavanger, Norway; emails: {shuo.zhang, krisztian.balog}@uis.no.

2157-6904/2020/01-ART13 $15.00
https://doi.org/10.1145/3372117

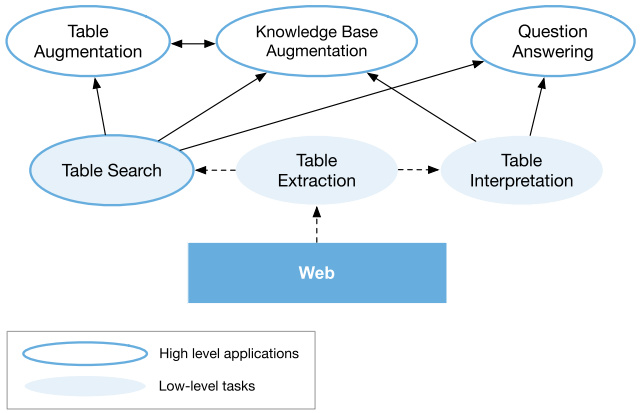

Fig. 1. Table-related information access tasks and their relationships.

a set of entities (such as people, organizations, locations, etc.) along with their attributes [15, 19, 23, 73, 79]. Relational tables are considered to be of high quality, because of the relational knowledge contained in them. However, unlike from tables in relational databases, these relationships are not made explicit in web tables; uncovering them is one of the main research challenges.

We organize relevant literature based on the task that is being addressed into six main categories. These are: table extraction (Section 3), table interpretation (Section 4), table search (Section 5), question answering on tables (Section 6), knowledge base augmentation (Section 7), and table augmentation (Section 8). The relationship between the different tasks is shown in Figure 1.

- *Table extraction* refers to the process of detecting tables in webpages, extracting them, and storing them in a consistent format, resulting in a table corpus.
- *Table interpretation* aims to uncover the semantics of the data contained in a table, with the aim of making tabular data intelligently processable by machines. This entails classifying tables according to some taxonomy, identifying what table columns are about, recognizing and disambiguating entity mentions in table cells, and uncovering the relationships between table columns.
- *Table search* (or *table retrieval*) is the task of answering a search query with a ranked list of tables. The search query may be a sequence of keywords or it may be a table itself.
- *Question Answering* utilizes structured data in tables for answering natural language questions.
- *Knowledge base augmentation* leverages tabular data for exploring, constructing, and augmenting knowledge bases.
- *Table augmentation* is directed at expanding an existing table with additional data. Specific subtasks include populating a table with new rows or columns, or finding missing cell values.

For each of these tasks, summarized in Table 1, we identify seminal work, describe the key ideas behind the proposed approaches, discuss relevant resources, and point out interdependencies among the different tasks.

The remainder of this article is organized as follows. In Section 2, we introduce the different table types and table corpora. Sections 3–8 are dedicated to the six main table tasks we have identified above. Finally, we conclude with a discussion of past progress and future research directions in Section 9.

Table 1. Overview of Table-related Information Access Tasks

Task	Input	Output	Key references
Table extraction	Webpages	Tabular data	Balakrishnan et al. [4], Bhagavatula et al. [8], Cafarella et al. [13, 15], Chen and Cafarella [16], Eberius et al. [23], and Lehmberg et al. [40]
Table interpretation	Table(s)	Structured data	Bhagavatula et al. [8], Cafarella et al. [15], Chen and Cafarella [16], Crestan and Pantel [19], Efthymiou et al. [24], Fan et al. [26], Hassanzadeh et al. [32], Ibrahim et al. [36], Lautert et al. [37], Lehmberg et al. [40], Limaye et al. [44], Muñoz et al. [48], Mulwad et al. [49, 50], Nishida et al. [54], Ritze and Bizer [59], Ritze et al. [61], Sekhavat et al. [63], Venetis et al. [68], Wang and Hu [73, 74], Wu et al. [75], and Zhang et al. [86]
Table search	Query	Ranked list of tables	Ahmadov et al. [2], Bhagavatula et al. [7], Cafarella et al. [13, 14], Das Sarma et al. [20], Lehmberg et al. [41], Limaye et al. [44], Nargesian et al. [51], Nguyen et al. [53], Pimplikar and Sarawagi [56], Yakout et al. [76], and Zhang and Balog [82, 85]
Question Answering	Natural language query	Structured data	Banerjee et al. [5], Berant et al. [6], Fader et al. [25], Neelakantan et al. [52], Pasupat and Liang [55], Sarawagi and Chakrabarti [62], and Sun et al. [65]
Knowledge base augmentation	Table(s)	Structured data	Bhagavatula et al. [8], Dong et al. [22], Fan et al. [26], Ibrahim et al. [36], Lehmberg et al. [40], Ritze and Bizer [59], Ritze et al. [60, 61], Sekhavat et al. [63], and Zhang et al. [86]
Table augmentation	Table	Table	Ahmadov et al. [2], Bhagavatula et al. [7, 8], Cafarella et al. [13, 14], Das Sarma et al. [20], Lehmberg et al. [41], Venetis et al. [68], Wang et al. [69], Yakout et al. [76], Zhang and Chakrabarti [78], and Zhang and Balog [81, 84]

2 TABLE TYPES AND CORPORA

In this section, we formally introduce tables (Section 2.1), present various types of tables (Section 2.2), and provide an overview of publicly available datasets (Section 2.3).

2.1 The Anatomy of a Table

A table T is a grid of cells arranged in rows and columns. Tables are used as visual communication patterns and as data arrangement and organization tools. In this article, our primary focus is on *web tables*, that is, tables embedded in webpages. Below, we define elements of a web table. We refer to Figure 2 for an illustration.

Table page title. The table page title T_p is the title of the webpage that embeds the table T.

Table caption. The caption of a table, T_c, is a short textual label summarizing what the table is about.

Fig. 2. Illustration of table elements in a web table: table page title (T_p), table caption (T_c), table headings (T_H), table cell ($T_{[i,j]}$), table row ($T_{[i,:]}$), table column ($T_{[:,j]}$), and table entities (T_E).

Table headings. Table headings, T_H, is a list of labels defining what each table row/column is about. Headings are typically in the first row/column in a table. In case of relational tables (see below, in Section 2.2), table headings are also referred to as *table schema* or *attribute names*.

Table cell. A table cell $T_{[i,j]}$ is specified with the row index i and column index j. Table cells hold (possibly empty) values and are considered as atomic units in a table.

Table row. A table row $T_{[i,:]}$ is a list of table cells lying horizontally in line i of a table.

Table column. A table row $T_{[:,j]}$ is a list of table cells lying vertically in column j of a table.

Table entities. Tables often mention specific entities, such as persons, organizations, locations. Table entities T_E is a set consisting of all the entities that are mentioned in the table.

2.2 Types of Tables

A number of table classification schemes have been proposed in the literature. We start by reviewing those, then propose a normalized categorization based on the main aspects these share.

In early work, Wang and Hu [73] make a distinction between genuine and non-genuine tables:

- *Genuine tables* are leaf tables, i.e., do not contain other tables, lists, forms, images or other non-text formatting tags in a cell. Furthermore, they contain multiple rows and columns.
- *Non-genuines tables* refer to all those that are not leaf tables.

Cafarella et al. [15] classify web tables into five main categories:

- *Extremely small tables* are those having fewer than two rows or columns.
- *HTML forms* are used for aligning form fields for user input.
- *Calendars* are a specific table type, for rendering calendars.
- *Non-relational tables* are characterized by low quality data, e.g., used only for layout purposes (many blank cells, simple lists, etc.).
- *Relational tables* contain high-quality relational data.

Crestan and Pantel [19] develop a fine-grained classification taxonomy, organized into a multi-layer hierarchy.

- *Relational knowledge tables* contain relational data.
  - —*Listings* refer to tables consisting a series of entities with a single attribute. In terms of layout direction, these are further classified as *vertical listings* or *horizontal listings*.
  - —*Attribute/value tables* describe a certain entity along with its attributes.
  - —*Matrix tables* have the same value type for each cell at the junction of a row and a column. *Calendars*, for example, can be regarded as matrix tables.
  - —*Enumeration tables* list a series of objects that have the same ontological relation (e.g., hyponomys or siblings).
  - —*Form tables* are composed of input fields for the user to input or select values.
- *Layout tables* do not contain any knowledge and are used merely for layout purposes.
  - —*Navigational tables* are meant for navigating within or outside a website.
  - —*Formatting tables* are used for visually organizing content.

Lautert et al. [37] refine the classification scheme of Crestan and Pantel [19].

- *Relational knowledge tables*
  - —*Horizontal tables* place attribute names on top (column header). Each column corresponds to an attribute.
  - —*Vertical tables* place attribute names on the left (row header). Each row represents an attribute.
  - —*Matrix tables* are three-dimensional data sets, where headers are both on the top and on the left.
- *Layout tables*, as before, are subdivided into *navigational tables* and *formatting tables*.

Relational knowledge tables are further classified according to a secondary type taxonomy.

- *Concise tables* contain merged cells (i.e., cells with the same value conflated together) to avoid value repetition.
- *Nested tables* contain a table in a cell.
- *Multivalued tables* refer to tables containing multiple values in a single cell. If all values in one cell come from one domain, then they are named as *simple multivalued web tables*; if not, then they are called *composed multivalued value tables*.
- *Splitted tables* present sequentially ordered repetitions in row/column headers (i.e., each label is repeated in every x cell).

With a particular focus on web spreadsheets, Chen and Cafarella [16] define the following type taxonomy:

- *Data frame spreadsheets* contain data frames, each consisting of two regions: data (numeric values) and headings (attribute names). These are further classified based on how they are arranged:
  - —*Hierarchical left spreadsheets* place attributes on the left of the data region.
  - —*Hierarchical top spreadsheets* put attributes on top of the data region.
- *Non-data frame (flat) spreadsheets* do not contain a data frame.
  - —*Relation spreadsheets* can be converted into the relational model [18].
  - —*Form spreadsheets* are designed for human-computer interaction.
  - —*Diagram spreadsheets* are for visualization purposes.
  - —*List spreadsheets* consist of non-numeric tuples.
  - —*Other spreadsheets* include schedules, syllabi, scorecards, and other files without a clear purpose.

Table 2. Classification of Table Types in This Article

Dimension	Type	Description
Content	Relational	Describes a set of entities with their attributes
	Entity	Describes a specific entity
	Matrix	A three-dimensional data set, with row and column headers
	Other	Special-purpose tables, including lists, calendars, forms, etc.
Layout	Navigational	Tables for navigational purposes
	Formatting	Tables for visual organization of elements

Our primary focus is on relational tables.

Table 3. Overview of Table Corpora

Table corpora	Type	#tables	Source
WDC 2012 Web Table Corpus	Web tables	147M	Web crawl (Common Crawl)
WDC 2015 Web Table Corpus	Web tables	233M	Web crawl (Common Crawl)
Dresden Web Tables Corpus	Web tables	174M	Web crawl (Common Crawl)
WebTables	Web tables	154M	Web crawl (proprietary)
WikiTables	Wikipedia tables	1.6M	Wikipedia
TableArXiv	Scientific tables	0.34M	arxiv.org

Eberius et al. [23] distinguish tables along two dimensions: content and layout. In terms of content, they adopt the classification scheme by Wang and Hu [73]. Considering layout purposes, they sort tables according to their logical structure into the following categories:

- *Horizontal listings* align cells horizontally.
- *Vertical listings* align cells vertically.
- *Matrix tables* refer to numerical tables.

Lehmberg et al. [40] distinguish between three main types of tables:

- *Relational tables* contain a set of entities, which could exist in rows (*horizontal*) or columns (*vertical*); the remainder of the cells contain their descriptive attributes.
- *Entity tables* describe a certain entity.
- *Matrix tables* refer to tables with numerical values only.

The above categorization systems are quite diverse, which is not surprising considering that each was designed with a different use-case in mind. Nevertheless, we can observe two main dimensions along which tables are distinguished: content and layout. We propose a normalized classification scheme, which is presented to Table 2. In the remainder of this article, we shall follow this classification when referring to a certain type of table. Among all table types, relational tables have received the bulk of attention in the literature. Accordingly, we focus primarily on relational tables and the tasks based on them in this survey.

2.3 Table Corpora

A number of table corpora have been developed in prior work, which are summarized in Table 3.

2.3.1 WDC Web Table Corpus. There are two versions of WDC Web Table Corpus,[1] which were released in 2012 and 2015, respectively. The 2012 version contains 147 million web tables, which were extracted from the 2012 Common Crawl corpus (consisting of 3.5 billion HTML pages). Tables

[1] http://webdatacommons.org/framework/.

in this corpus are roughly classified as relational or non-relational in terms of layout. Statistically, 3.3 billion HTML pages were parsed and 11.2 billion tables were identified; tables that are not innermost (that is, contain other tables in their cells) were discarded. 1.3% of the remaining tables (originating from 101 million different webpages) were labeled as relational tables. Tables in this corpus are not classified further and neither is table context data provided.

The WDC 2015 Web Table Corpus, constructed by Lehmberg et al. [40], contains 10.24 billion *genuine* tables. The extraction process consists of two steps: table detection and table classification. The percentages of *relational*, *entity*, and *matrix* tables are 0.9%, 1.4%, and 0.03%, respectively. The remaining 97.75% accounts for *layout* tables. When storing a table, its orientation is also detected, indicating how the attributes are placed. In horizontal tables, the attributes are placed in columns, while in vertical tables they represent rows. There are 90.26 million relational tables in total. Among those, 84.78 million are horizontal and 5.48 million are vertical. The average number of columns and rows in horizontal tables are 5.2 and 14.45. In vertical tables, these numbers are 8.44 and 3.66, respectively. Lehmberg et al. [40] also extract the column headers and classify each table column as being numeric, string, data, link, boolean, or list. The percentages of the numeric and string columns are 51.4% and 47.3%, respectively. Besides, the text surrounding the table (before and after) is also provided. Furthermore, Lehmberg et al. [40] provide the English-language Relational Subset, comprising of relational tables that are classified as being in English, using a naive Bayesian language detector. The language filter considers a table's page title, table header, as well as the text surrounding the table to classify it as English or non-English. The average number of columns and rows in this subset are 5.22 and 16.06 for horizontal tables, and 8.47 and 4.47 for vertical tables. The percentages of numeric and string columns are 51.8% and 46.9%.

A total of 139 million tables in the WDC 2015 Web Table Corpus are classified as entity tables. Out of these, 76.70 million are horizontal and 62.99 million are vertical tables. The average number of columns and rows are 2.40 and 9.08 for horizontal tables, and 7.53 and 2.06 for vertical tables. The column data types are quite different from that of relational tables. String columns are the most popular, amounting to 86.7% of all columns, while numeric columns account for only 9.7%.

The complete corpus as well as the different subcorpora are made publicly available.[2]

2.3.2 Dresden Web Table Corpus. Eberius et al. [23] also extracted tables from the Common Crawl web corpus. The total number of tables is 174 million, which is reduced to 125 million after filtering with regards to content-based duplication. The Dresden Web Table Corpus contains only the core table data, and not the entire HTML page. Even though the corpus is not available for download directly, the table extraction framework (extractor code and companion library for working with the data set) is made publicly available.[3]

2.3.3 WebTables. Cafarella et al. [15] extracted 154 million high-quality relational web tables from a (proprietary) general-purpose web crawl. Unfortunately, this corpus is not made public. However, frequency statistics of attributes, known as the ACSDb dataset (cf. Section 8.2), is available for download.[4]

2.3.4 Wikipedia Tables. Bhagavatula et al. [8] focused on Wikipedia and extracted 1.6 million high-quality relational tables. Each table is stored as a JSON file, including table body, table caption, page title, column headers, and the number of row and columns. The existing links in the tables are also extracted and stored in a separate file. The corpus is available for download.[5]

[2] http://webdatacommons.org/webtables/#results-2015.
[3] https://wwwdb.inf.tu-dresden.de/misc/dwtc/.
[4] https://web.eecs.umich.edu/~michjc/data/acsdb.html.
[5] http://websail-fe.cs.northwestern.edu/TabEL/.

Table 4. Selected Features for Relational Table Classification (RTC), Header Detection (HD), and Table-type Classification (TTC) (Part 1/2)

Features	Explanation	Task	Source
Global layout features			
Max rows	Maximal number of cells per row	RTC, TTC	[19, 23]
Max cols	Maximal number of cells per column	RTC, TTC	[19, 23]
Max cell length	Maximal number of characters per cell	RTC, TTC	[19, 23]
#rows	Number of rows in the table	RTC, HD	[15]
#cols	Number of columns in the table	RTC, HD	[15]
%rows	Percentage of rows that are mostly NULL	RTC	[15]
#cols non-string	Number of columns with non-string data	RTC	[15]
μ	Average length of cell strings	RTC	[15]
δ	Standard deviation of cell string length	RTC	[15]
$\frac{\mu}{\delta}$	Cell string length	RTC	[15]
%length one	Percentage of columns with $\lvert len(row_1) - \mu \rvert > 2\delta$	HD	[15]
%length two	Percentage of columns with $\delta \leq \lvert len(row_1) - \mu \rvert \leq 2\delta$	HD	[15]
%length three	Percentage of columns with $\lvert len(row_1) - \mu \rvert < \delta$	HD	[15]
Avg rows	Average number of cells across rows	RTC, TTC	[23, 74]
Avg cols	Average number of cells across columns	RTC, TTC	[23, 74]
Avg cell length	Average number of characters per cell	RTC, TTC	[19, 23, 74]

2.3.5 Scientific Tables. Scientific tables are a particular type of table, which contain valuable knowledge and are available in large quantities. The TableArXiv corpus[6] consists of 341,573 tables, extracted from physics e-prints on arxiv.org. Along with the corpus, 105 information needs and corresponding relevance judgements are also provided for the task of scientific table search.

3 TABLE EXTRACTION

A vast number of tables can be found on the Web, produced for various purposes and storing an abundance of information. These tables are available in heterogenous format, from HTML tables embedded in webpages to files created by spreadsheet programs (e.g., Microsoft Excel). To conveniently utilize these resources, tabular data should be extracted, classified, and stored in a consistent format, resulting ultimately in a table corpus. This process is referred to as *table extraction*. In this section, we present approaches for the table extraction task, organized around three main types of tables: web tables, Wikipedia tables, and spreadsheets.

3.1 Web Table Extraction

Table extraction is concerned with the problem of identifying and classifying tables in webpages, which encompasses a range of more specific tasks, such as relational table classification, header detection, and table-type classification. These three tasks (relational table classification, header detection, and table-type classification) are commonly approached as a supervised learning problem and employ similar features; these features are summarized in Tables 4 and 5. In Sections 3.1.1–3.1.3, we organize the literature according to the three tasks.

3.1.1 Relational Table Classification. The identification of tables on the Web is usually straightforward based on HTML markup. Tables, however, are also used extensively for formatting and layout purposes. Therefore, web table extraction involves a data cleaning subtask, i.e., identifying

[6] http://boston.lti.cs.cmu.edu/eager/table-arxiv/.

Table 5. Selected Features for Relational Table Classification (RTC), Header Detection (HD), and Table-type Classification (TTC) (Part 2/2)

Features	Explanation	Task	Source
Layout features			
Std dev rows	Standard dev. of the number of cells per row	RTC	[23, 74]
Std dev cols	Standard dev. of the number of cells per column	RTC	[23, 74]
Std dev cell length	Standard dev. of the number of characters per cell	RTC	[19, 23, 74]
Local length avg	Average size of cells in segment	RTC	[19, 23]
Local length variance	Variance of size of cells in segment	RTC	[19, 23]
Content features			
%body non-string	Percentage of non-string data in table body	HD	[15]
%header non-string	Percentage of non-string data in the first row	HD	[15]
%header punctuation	Percentage of columns with punctuation in the first row	HD	[15]
Local span ratio	Ratio of cells with a ⟨span⟩ tag	RTC, TTC	[19, 23]
Local ratio header	Cells containing a ⟨th⟩ tag	RTC, TTC	[19, 23]
Local ratio anchor	Cells containing an ⟨a⟩ tag	RTC, TTC	[19, 23]
Local ratio input	Cells containing an ⟨input⟩ tag	RTC, TTC	[19, 23]
Ratio img	Ratio of cells containing images	RTC, TTC	[19, 23, 74]
Ratio form	Ratio of cells containing forms	RTC, TTC	[23, 74]
Ratio hyperlink	Ratio of cells containing hyperlinks	RTC, TTC	[23, 74]
Ratio alphabetic	Ratio of cells containing alphabetic characters	RTC, TTC	[23, 74]
Ratio digit	Ratio of cells containing numeric characters	RTC, TTC	[23, 74]
Ratio empty	Ratio of empty cells	RTC, TTC	[23, 74]
Ratio other	Ratio of other cells	RTC, TTC	[23, 74]

and filtering out "bad" tables (where "bad" usually denotes non-relational tables). *Relational table classification* (also known as identifying high-quality or genuine tables) refers to the task of predicting whether a web table contains relational data.

One of the pioneering works utilizing tables on the Web is the WebTables project [14, 15]. Cafarella et al. [15] regard relational tables as high-quality tables, and filter those by training a rule-based classifier. The classifier uses table characteristics, like table size and table tags, as features. The model is trained on a set of manually annotated tables (as being relational or non-relational) by two human judges. As a result, they construct a high-quality table corpus, consisting of 154 million tables, filtered from 14.1 billion HTML tables (cf. Section 2.3.3). Balakrishnan et al. [4] follow a similar approach for relational table classification, but use a richer set of features, which include both syntactic and semantic information. Syntactic features are related to the structure of the table, as in Reference [15] (e.g., number of rows and columns). Semantic features are obtained by (1) determining whether the table falls into a boilerplate section of the containing page, (2) detecting subject columns (using a binary SVM classifier trained based on one thousand manually labeled tables), (3) identifying column types (which will be detailed later, in Section 4.1), (4) and detecting binary relationships between columns (by analyzing how these relationships are expressed in the text surrounding the table). Wang and Hu [74] define a table as *genuine*, if it is a leaf table where no subtable exists in any of the cells. They employ machine learned classifiers (decision trees and support vector machines) to classify relational tables, using three main groups of features: layout features, content-type features, and word group features. The layout features and most of the content features are listed in Tables 4 and 5. As for word group features, Wang and Hu [74] treat each table as a document and compute word frequency statistics. In follow-up work, the authors

also experiment with other machine learning methods (Naive Bayes and weighted kNN), using the same set of features [73]. Building on Reference [74], Eberius et al. [23] carry out relational table classification as well as classification according to layout type (vertical listings, horizontal listing, and matrix tables). Their first method performs classification along both dimensions simultaneously, using a single layer. Their second approach separates the two tasks into two layers, where the first layer executes table detection and, subsequently, the second layer determines the layout type. Various machine learning methods are employed, including decision trees, Random Forests, and SVMs, using a combination of global and local features; a selection of features are listed in Table 5. As a result, Eberius et al. [23] classify millions of tables and generate the Dresden Web Table Corpus (DWTC, cf. Section 2.3.2).

To obtain metadata for relational tables, Eberius et al. [23] consider whether tables have a header row or not. They find that 71% of the tables in the corpus have a relational header. For the remaining 29%, they attempt to generate synthetic labels by comparing the column content to similar columns that have proper labels. Cafarella et al. [13] design a system called OCTOPUS, which combines search, extraction, data cleaning, and integration. Further challenges related applying WebTables in practice, including table identification and table semantics recovery, are detailed in Reference [4]. The resulting system, Google Fusion Tables, is made publicly available.[7]

3.1.2 Header Detection. To extract data in a structured format, the semantics of tables need to be uncovered to some extent. One question of particular importance is whether the table contains a header row or column. This is known as the task of *header detection*. Headers may be seen as a particular kind of table metadata. Header detection is commonly addressed along with the other two tasks and uses similar features (cf. Tables 4 and 5).

3.1.3 Table-type Classification. Another type of metadata that can help to uncover table semantics is table type. *Table-type classification* is the task of classifying tables according to a pre-defined type taxonomy (cf. Section 2.2 for the discussion of various classification schemes). Additional metadata extracted for tables includes the embedding page's title, the table's caption, and the text surrounding the table.

The same features that are intended for relational table classification and header detection can also be used for table-type classification [15, 16, 40, 73, 74]. For example, the features listed in Tables 4 and 5 are used in Reference [23] for both relational table classification and table-type classification. Instead of directly classifying tables as relational or not, this can also be done indirectly by saying that a table is relational if relational information can successfully be extracted from it [16]. Table extraction is also involved in a number of other studies, but these datasets are not publicly available. For example, with the purpose of data integration, Wang et al. [71] use a rule-based filtering method to construct a corpus of 1.95 billion tables. For a type-classification study, Crestan and Pantel [19] extract a corpus of 8.2 billion tables. Using a more fine-grained type taxonomy (see Section 2.2), table-type classification is approached as a multi-class classification problem. Crestan and Pantel [19] propose a rich set of features, including global layout features, layout features, and content features. Global layout features include the maximum number of rows, cols, and maximum cell length. Layout features include average length of cells, length variance, and the ratio of row/column span. Content features include HTML features (based on HTML tags) and lexical features (based on cell content). As a follow-up work, Lautert et al. [37] additionally consider the category obtained in Reference [19] as one features to further classify tables into a multi-layer taxonomy. The first layer of classification is similar to the one in Reference [19]. A second layer of classification focuses on relational knowledge, by additionally dividing relational knowledge tables

[7] https://research.google.com/tables.

into concise, nested, multivalued (simple or composed), and split tables. Lehmberg et al. [40] construct a web table corpus from Common Crawl (WDC Web Table Corpus, cf. 2.3.1). First, they filter out non-genuine tables (referred to as not innermost tables, i.e., tables that contain other tables in their cells) and tables that contain less than 2 columns or 3 rows. Then, using the table extraction framework of DWTC, the filtered tables are classified as either relational, entity matrix, or layout tables [23]. Recently, deep-learning methods have also been used for table-type classification. For example, Nishida et al. [54] regard a table as a matrix of texts, which is similar to an image. Utilizing the type taxonomy from Reference [19], they design a framework named TabNet, consisting of RNN Encoder, CNN Encoder, and Classifier. The RNN Encoder encodes the input table cells to create a 3D table volume, like image data, in the first step. The CNN encoders encode the 3D table volume to capture table semantics, which is used for table-type classification by the Classifier. Even though TabNet is designed to capture table structure, it can be applied to any matrix for type classification.

3.2 Wikipedia Table Extraction

Wikipedia tables may be regarded as a special case of web tables. They are much more homogeneous than regular web tables and are generally of high quality. Therefore, no additional data cleaning is required. Bhagavatula et al. [8] construct a Wikipedia table corpus, consisting of 1.6 million tables, with the objective of extracting machine-understandable knowledge from tables. For details, we refer to Section 2.3.4.

3.3 Spreadsheet Extraction

The Web contains a great variety and number of Microsoft Excel *spreadsheets*. Spreadsheets are often roughly relational. Chen and Cafarella [16] design an automatic system to extract relational data, to support data integration operations, such as joins. A *data frame* is defined as a block of numerical data. Chen and Cafarella [16] extract 410,554 Microsoft Excel files from the ClueWeb09 Web crawl by targeting Excel-style file endings that contain a data frame. Within a data frame, the attributes might lie on the left or top. Chen and Cafarella [16] find that 50.5% of the spreadsheets contain a data frame and 32.5% of them have hierarchical top or left attributes (the rest are called *flat* spreadsheets). Among the 49.5% non-data frame spreadsheets, 22% are relational, 10.5% are forms, 3.5% are diagrams, 3% are lists, and 10.5% are other spreadsheets. For each spreadsheet, the extraction system firstly finds the data frame, then extracts the attribute hierarchy (top or left), and finally builds relational tuples (see Section 4.3 for more details).

4 TABLE INTERPRETATION

Table interpretation encompasses methods that aim to make tabular data processable by machines. Specifically, it focuses on interpreting tables with the help of existing knowledge bases. Bhagavatula et al. [8] identify three main tasks aimed at uncovering table semantics: (1) *column-type identification*, that is, associating a table column with the type of entities or relations it contains, (2) *entity linking*, which is the task of identifying mentions of entities in cells and linking them to entries in a reference knowledge base, and (3) *relation extraction*, which is about associating a pair of columns in a table with the relation that holds between their contents. See Figure 3 as the task illustration. Table 6 provides an overview of studies addressing either or all of these tasks.

4.1 Column-type Identification

In relational tables, the *core column* (also referred to as *subject column*, *name column*, or *entity column* [38]) is a special column that contains entities. Commonly, this is the leftmost column in a table (and other table columns correspond to attributes or relationships of these entities). The

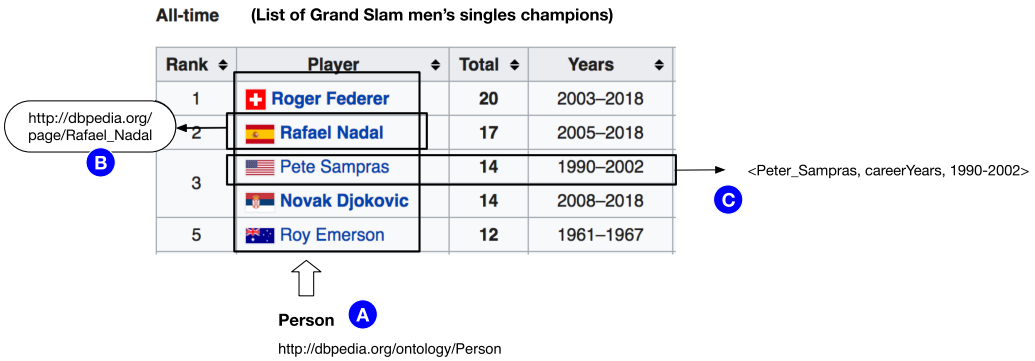

Fig. 3. Illustration of table interpretation: (A) Column-type Identification. (B) Entity Linking. (C) Relation extraction.

Table 6. Overview of Table Interpretation Tasks Addressed in Various Studies

Reference	Column-type Identification	Entity Linking	Relation Extraction
Bhagavatula et al. [8]		✓	
Chen and Cafarella [16]			✓
Efthymiou et al. [24]		✓	
Fan et al. [26]	✓		
Hassanzadeh et al. [32]		✓	
Ibrahim et al. [36]		✓	
Lehmberg and Bizer [38]	✓		
Limaye et al. [44]	✓	✓	
Muñoz et al. [48]			✓
Mulwad et al. [49]			✓
Mulwad et al. [50]	✓	✓	✓
Ritze and Bizer [59]		✓	
Ritze et al. [61]		✓	
Sekhavat et al. [63]			✓
Venetis et al. [68]	✓		✓
Wang et al. [71]	✓		
Wu et al. [75]		✓	
Zhang and Chakrabarti [78]	✓	✓	
Zhang [87]	✓	✓	✓

identification of the core column is a central pre-processing step for entity linking, table augmentation, and relation extraction. Most of the existing work assumes the presence of a single core column. Such tables are also known as single-concept relational tables. However, in some cases, a relational table might have multiple core columns that may be located at any position in the table [11], called a multi-concept relational table. Braunschweig et al. [11] extend a single-concept method, which utilizes table headings as well as intrinsic data correlations, with more features, like the correlation with the left neighbor, to determine all the core columns. We focus on single-concept relational tables in the remainder of this section.

Generally, *column-type identification* is concerned with determining the types of columns, including locating the core column. This knowledge can then be used to help interpret a table. Table 7

Table 7. Comparison of Column-type Identification Tasks

Reference	Knowledge base	Method
Fan et al. [26]	Freebase	Concept-based method + crowdsourcing
Lehmberg and Bizer [38]	DBpedia	Feature-based classification
Mulwad et al. [50]	Wikitology	Entity search
Wang et al. [71]	Probase	Heading-based search
Venetis et al. [68]	Automatically built IS-A database	Majority vote
Zhang and Chakrabarti [78]	-	Semantic graph method
Zhang [87]	Freebase	Unsupervised featured-based method

displays a summary of the methods, which we shall discuss below. Venetis et al. [68] argue that the meaning of web tables is "only described in the text surrounding them. Header rows exist in few cases, and even when they do, the attribute names are typically useless" [68]. Therefore, they add annotations to tables to describe the sets of entities in the table (i.e., column-type identification). This is accomplished by leveraging an IS-A database of entity-class pairs. This IS-A database is created by aggregating all the entity-class $\langle e, C \rangle$ pairs that are mined from the Web (100 million English documents using 50 million anonymized queries) using the pattern "C [such as|including] e [and|,|.]." A class label is assigned to a column if a certain fraction of entities in that column is identified with that label in the IS-A database. Venetis et al. [68] conclude that using a knowledge base (YAGO) results in higher precision, while annotating against the IS-A database has better coverage, i.e., higher recall. Mulwad et al. [50] map each cell's value in a column to a ranked list of classes, and then selects a single class that best describes the whole column. To get the ranked list of classes, a complex query, based on cell values, is submitted to the Wikitology knowledge base [66]. Possible class labels are obtained by utilizing the relevant entities in the knowledge base. Then, a PageRank-based method is used to compute a score for the entities' classes, from which the one with the highest score is regarded as the class label. Mapping each column to one of the four types ("Person," "Place," "Organization," and "Other"), Mulwad et al. [50] achieve great success on "Person" and "Places," and moderate success on "Organization" and "Other" types, due to their sparseness in the reference knowledge base.

Because of the inherent semantic heterogeneity in web tables, not all tables can be matched to a knowledge base using pure machine learning methods. Fan et al. [26] propose a "two-pronged" approach for matching web tables' columns to a knowledge base. First, a concept-based method is used to map each column to the best knowledge base concept. Specifically, they employ Freebase as the concept catalog. Second, a hybrid human-machine framework discerns the concepts for some exceptional columns manually. The matches between table columns and their candidate concepts are represented as a bipartite graph, where relationships correspond to edges. Fan et al. [26] employ crowdsourcing for this task, and find that a higher payment leads to better accuracy.

A table corpus is constructed in Reference [71] and it is classified according to a probabilistic taxonomy called Probase, which is able to understand entities, attributes, and cells in tables. To get the table semantics, a top-k candidates concepts are returned based on the table headings, which is similar to the idea in Reference [44] (cf. Section 4.2). The candidate concepts assist to detect entities in a given column by computing the maximum number of common concepts. In turn, the entity column type is obtained based on the confidence of the concepts. Wang et al. [71] demonstrate that table headers can help to understand the columns as well as to identify the core column.

Lehmberg and Bizer [38] propose a categorization scheme for web table columns that distinguishes the different types of relations that appear in tables on the Web. First, a binary relation is a relation that holds between the core column and the values in another column, e.g., populations

of cities. Second, an N-ary relation is a relation that holds between the core column and additional entities and values in other columns. Third, an independent column is one that has no direct relation with the core column. Lehmberg and Bizer [38] propose a feature-based classifier that distinguishes between these three types of relations for better table interpretation.

Zhang [87] presents TableMiner+ for semantic table interpretation, where core column detection and type identification linking are executed at the same stage. Zhang [87] first simply uses regular expressions and classifies cells as "empty," "entities," "numbers," "data," "text," or "other." Then, evidence is gathered from the Web for each column to predict the likelihood of it being the subject (core) column. Specifically, a keyword query is composed from all text content in each row, and the subject entity in this row is detected by recognizing the top-ranked page. Finally, an unsupervised feature-based method is employed to find the core column and type by aggregating evidence across all rows. Features include the fraction of empty cells, the fraction of cells with unique content, context match score (heading frequency within surrounding text), and web search score. The main differences between TableMiner+ and other methods are twofold: (1) TableMiner+ uses context outside the tables while others not, and (2) it adopts an iterative process to optimize and enforce the interdependence between different annotation tasks (entity linking and relation extraction).

The above methods work well for string values and static attributes but perform poorly for numeric and time-varying attributes. Zhang and Chakrabarti [78] build a semantic graph over web tables suited for numeric and time-varying attributes by annotating columns with semantic labels, like timestamp, and converting columns by comparing with columns from other tables. While this method is designed for entity augmentation, it can also be utilized for column-type identification.

4.2 Entity Linking

Recognizing and disambiguating specific entities (such as persons, organizations, locations, etc.), a task commonly referred to as *entity linking*, is a key step to uncovering semantics [8]. Since many web tables are relational, describing entities, entity linking is a key step to understanding what the table is about. A number of table-related tasks, such as row population [69, 81], column population [81], and table search [82], rely on entity linking in tables. Table 8 compares the tasks we will discuss below.

Limaye et al. [44] pioneered research on table entity linking. They introduce and combine five features, namely, the TF-IDF scores between cell text and entity label, the TF-IDF scores between the column header and the type label, the compatibility between column type and cell entity, compatibility between relation and pair of column types, and the compatibility between relation and entity pairs. Their idea of a factor graph-based entity linking approach influenced later research. For example, Bhagavatula et al. [8] design a system called TabEL for table entity linking. TabEL employs a graphical model that "assigns higher likelihood to sets of entities that tend to co-occur in Wikipedia documents and tables" [8]. Specifically, it uses a supervised learning approach and annotated mentions in tables for training. TabEL focuses on Wikipedia table and executes mention identification for each table cell, then obtains a set of candidate entities for disambiguation. The disambiguation technique is based on the assumption that entities in a given row and column tend to be related. They use a collective classification technique to optimize a global coherence score for a set of entities in a given table. By comparing against traditional entity linking methods for unstructured text, Bhagavatula et al. [8] demonstrate the need for entity linking methods designed specifically for tables.

Unlike most methods, which consider a single knowledge base, Wu et al. [75] propose an entity linking method for web tables that considers multiple knowledge bases to ensure good coverage. From each knowledge base, entities whose names share at least one word with the content of a given table cell are taken as candidates. Then, an entity disambiguation graph is constructed,

Table 8. Comparison of Entity Linking Tasks

Reference	Knowledge base	Method
Bhagavatula et al. [8]	YAGO	Graphical model
Efthymiou et al. [24]	DBpedia	Vectorial representation and ontology matching
Hassanzadeh et al. [32]	DBpedia, Schema.org, YAGO, Wikidata, and Freebase	Ontology overlap[a]
Ibrahim et al. [36]	YAGO	Probabilistic graphical model
Lehmberg and Bizer [39]	DBpedia	Feature-based method
Lehmberg et al. [40]	Google Knowledge Graph	—
Limaye et al. [44]	YAGO catalog, DBpedia, and Wikipedia tables	Inference of five types of features[b]
Mulwad et al. [50]	Wikitology	SVM classifier
Ritze and Bizer [59]	DBpedia	Feature-based method
Ritze et al. [60, 61]	DBpedia	Feature-based method
Wu et al. [75]	Chinese Wikipedia, Baidu Baike, and Hudong Baike	Probabilistic method[c]
Zhang et al. [86]	DBpedia	Instance-based schema mapping
Zhang [87]	Freebase	Optimization

[a] KB comparison.
[b] Designed for table search.
[c] Multiple KBs.

consisting of mention nodes, entity nodes, mention-entity edges, and entity-entity edges. The method utilizes entity linking "impact factors," which are two probabilities, for ranking candidates and for disambiguating entities, based on mention nodes and edges. To incorporate multiple knowledge bases, "same-As" relations between entities from different knowledge bases are leveraged to reduce errors and to improve coverage. This system shares many similarities with TabEL. TabEL, however, does not consider synonyms and deals with a single KB. Efthymiou et al. [24] propose three unsupervised annotation methods for matching web tables with entities. The first is a lookup-based method, which relies on the minimal entity context from the tables to discover correspondences to the knowledge base. A second method exploits a vectorial representation of the rich entity context in a knowledge base to identity the most relevant subset of entities in web tables. The third method is based on ontology matching, and exploits schematic and instance information of entities available both in a knowledge base and in a web table. Efthymiou et al. [24] find that hybrid methods that combine the second and third methods (in any order) tend to perform best. The column-type identification component of TableMiner+ [87] has already been discussed earlier, in Section 4.1. Building on this, TableMiner+ uses the partial annotations from column-type identification for all columns to guide entity linking in the rest of the table. It re-ranks table rows under the assumption that some cells are easy to disambiguate, i.e., they have more candidates or the text is less ambiguous (candidate sampling). In each iteration of this so-called *learning* phase, it searches new candidates and compares the feature representation of each candidate entity (web search results) against all the feature representations of that cell (using the same features as for column-type identification). The associated concepts with the highest scoring entity are gathered as candidate concepts for the column. These are further compared against those from the previous iteration in the *learning* phase (optimization). The process is repeated until convergence is reached.

Mulwad et al. [50] exploit the predicted class labels for columns (see Section 4.1) as additional evidence, to link entities in table cells. A knowledge base is queried to construct a feature vector,

which comprises the entity's retrieval score, Wikipedia page length, PageRank, and so on, which are used for computing the similarity score against the table cell's value. The feature vectors are input to an SVMRank classifier, which outputs a ranked list of entities. The top-ranked entity is selected and is used to introduce two more features for a final classification (the SVM rank score for the top-ranked entity and the score difference between the top two entities). The final classification yields a binary outcome whether the entity should be linked or not. Similar to the column-type identification task, this method performs very well on the "Person" and "Place" entity types, achieves moderate accuracy on "Organization," and low accuracy on "Other" (for the same reason of sparseness, as before). A similar approach is taken by Lehmberg et al. [40], but they perform entity linking in table cells first, using the Google Knowledge Graph, and then use this information for getting class labels for columns.

Another study on knowledge base matching in Reference [36] aims to overcome the problem of table matching and aggregation by making sense of entities and quantities in web tables. Ibrahim et al. [36] map the table elements of table headers, entity tables cells, and numeric table cells to different knowledge bases. Specifically, (1) tables headers denote classes or concepts and are linked to a taxonomic catalog or to Wikipedia pages, (2) named entities are mapped to a knowledge base (YAGO), and (3) numeric cells, which denote quantities, are mapped to normalized representations. An interesting observation made about quantity linking is that many of the linking errors are (1) due to the absence of specific measures or units and (2) because of ambiguous headings, like "Nat."

As mentioned in Section 3.1, a *relational table* refers to an entity-attribute table, where a set of entities and their attributes are listed. Zhang et al. [86] propose an instance-based schema mapping method to map entity-attribute tables to a knowledge base. In Reference [86], an entity-attribute table is supposed to have a key column, which contains a set of entities. Each tuple is an entity with its attributes. Then, memory-based indexes are used to judge whether a tuple contains candidate entities, resulting in an evidence mapping vector. This vector is then used for finding a table-to-KB schema mapping, which essentially serves as a bridge between web tables and knowledge bases.

The choice of the knowledge base for uncovering table semantics is important. Hassanzadeh et al. [32] give a detailed study on the utility of different knowledge bases, including DBpedia, Schema.org, YAGO, Wikidata, and Freebase. The method of concept linking in Reference [32] is tagging columns with entity types (classes) in the knowledge base. Specifically, they firstly get the basic statistical distribution of tables sizes and values. Then, with the help of the selected knowledge base, the distribution of overlap scores in the ontology is obtained. Finally, these scores can give an indication of how well the table's content is covered by the given knowledge base.

Ritze and Bizer [59] study the utility of different features for entity linking in tables. These features are extracted from the table itself (such as entity label, table, URL, page title, and surrounding text) or from the knowledge base (such as instance label and classes). They introduce a specific similarity linker for each feature, resulting in similarity matrices, representing feature-specific results. These matrix predictors can be used to decide which features to use for which web table. Ritze et al. [61] implement the T2K Match framework [60] to map the WDC Web corpus to DBpedia, for knowledge base extension (entity linking happens the same time with, and rely on, schema matching and table-type identification). Taking table content as evidence, the incomplete and unclear values of DBpedia can be filled and corrected. They find that "only 1.3% of all tables that were extracted from the Web crawl contained relational data. Out of these relational tables, about 3% could be linked to DBpedia" [61]. However, the above methods tend to perform better for large tables, i.e., tables with several rows. It is considered as one of the main limitations of linking tabular mentions to DBpedia. To overcome this, Lehmberg and Bizer [39] stitch tables, i.e., merge tables from the same website as a single large table, to improve entity linking performance.

Table 9. Comparison of Relation Extraction Tasks

Reference	Knowledge base	Method	Source of extraction
Chen and Cafarella [16]	—	Classification	Each value in the value region
Muñoz et al. [48]	DBpedia	Lookup-based	Any pair of entities in the same row
Mulwad et al. [49]	DBpedia	Semantic passing	Any pair of columns
Mulwad et al. [50]	DBpedia	Utilizing CTI and EL	Any pair of columns
Sekhavat et al. [63]	YAGO, PATTY	Probabilistic	Any pair of entities in the same row
Venetis et al. [68]	IS-A database	Frequency-based	Core + attribute columns
Zhang [87]	Freebase	Optimization	Any pair of columns

4.3 Relation Extraction

Relation extraction refers to the task of associating a pair of columns in a table with the relation that holds between their contents and/or extracting relationship information from tabular data and representing them in a new format (e.g., RDF). Table 9 summarizes the methods we will discuss below.

Venetis et al. [68] add annotations to tables to describe the binary relationships represented by columns. This is accomplished by leveraging a relations database of (argument1, predicate, argument2) triples. For binary relationships, the relationship between columns A and B is labeled with R if a substantial number of pairs of values from A and B occur in the relations database. Venetis et al. [68] are only able to annotate a small portion of a whole table corpus (i.e., low recall). They discover that the vast majority of these tables are either not useful for answering entity-attribute queries, or can be labeled using a handful of domain-specific methods.

Mulwad et al. [50] propose a preliminary method for relation extraction, which utilizes the results of entity linking and column-type prediction. Specifically, the method generates a set of candidate relations by querying DBpedia using SPARQL. Each pair of strings in two columns vote for the candidate relation. The normalized scores are used for ranking candidate relations and the highest one is taken as the column relation. In follow-up work, Mulwad et al. [49] implement an improved semantic message passing method to extract RDF triples from tables. The semantic message passing first pre-processes the input table, separated by table elements such as column headers, cell values, columns, and so on. Then, the processed table is passed to a *query and rank* module, which turns to knowledge bases from Linked Open Data to find candidates for each table element. Finally, a *joint inference* step uses a probabilistic graph model to rank candidate relations that were identified for the table elements. Mulwad et al. [49] point out that current methods rely on semantically poor and noisy knowledge bases and can only interpret part of a table (low recall). Moreover, systems for numeric values remain challenging, which is consistent with Reference [36].

TableMiner+ [87] interprets relations between the core column and other columns on each row independently. It computes an individual confidence score for each candidate relation from each row. The candidate set of relations for two columns is derived by collecting the winning relations on all rows. A final confidence score of a candidate relation adds up its instance and context score computed based on context overlap. It is used to find the relation with the highest confidence. A key finding in Reference [87] is that a system that is based on partial tabular data can be as good as systems that use the entire table.

Relation extraction can also be used to augment Linked Data repositories [63]. Sekhavat et al. [63] propose a probabilistic approach using under-explored tabular data. Assuming that the entities co-occurring in the same table are related, they focus on extracting relations between pairs of entities appearing in the same row of a table. Entities in table cells are mapped to a knowledge base first. Then, sentences containing both entities from the same table row are collected from a text corpus. Next, textual patterns (describing the relationship between these two entities) are

Table 5-8: Active Aviation Pilots and Flight Instructors: 2000[1]

State	Total	Students	Airplane pilots[2]			Misc.[3]	Flight instructor[4]
			Private	Commercial	Airline transport		
Alabama	7,262	1,170	3,065	1,649	1,084	294	920
Alaska	8,638	833	3,686	2,130	1,906	83	1,118
Arizona	17,429	2,329	6,508	3,345	4,654	593	2,617
Arkansas	4,988	776	2,153	1,206	788	65	634
California	71,053	10,173	31,571	13,448	12,786	3,075	8,984
Colorado	17,539	2,320	6,256	3,144	5,138	681	2,549
Connecticut	6,523	944	2,714	989	1,648	228	837

Fig. 4. Excerpt from a table containing hierarchical attributes. The example is taken from the U.S. Department of Transportation (http://www.api.faa.gov/CivilAir/index.html).

extracted. Finally, the probability of the possible relations is estimated using Bayesian inference. A new relation, which is a triple consisting of two entities and a pattern, can be added to the Linked Data repository for augmentation. Muñoz et al. [48] utilize entity annotations in Wikipedia tables. Taking existing relations between entities in DBpedia, they look these entities up in Wikipedia tables. This then indicates that the same relation stands between entities in other rows of this table.

Chen and Cafarella [16] introduce a system to automatically extract relational data from spreadsheets instead of the Web. Most of the methods on spreadsheets require users to provide sheet-specific rules [1, 35]. In contrast, Chen and Cafarella [16] realize it in an automatic manner. Generally, the system detects attributes and values, identifies the hierarchical structure of attributes, and generates relational tuples from spreadsheet data. Specifically, the so-called *frame finder* module of their system aims to identify the data frame regions within a spreadsheet. These data frames consist of attribute and value regions. First, it labels each row with one of the categories: title, header, data, or footnote. Then, the data frame regions are created, which are passed to the *hierarchy extractor* for recovering the attribute hierarchies by finding all parent-child pairs in an attribute region. See Figure 4 for an illustration, where *Airplane pilots* and *Airline transport* would be annotated as a parent-child attribute pair. Finally, a series of parent-child candidates are generated and the true parent-child pairs are identified through classification. Alternatively, a so-called enforced-tree classification is proposed, which constructs a strict hierarchical tree for attributes. In the end, relational tuples are generated from the value region, whose value is annotated with one attribute from the attribute hierarchy.

4.4 Other Tasks

Data translation is concerned with the problem of mapping raw data, collected from heterogenous sources, to a transformed version for the end user [33]. Tables encode a large number of mapping relationships as column pairs, e.g., person and birthday, which can be useful data assets for data translation. Wang and He [72] propose to automatically synthesize mapping relationships using table corpora by leveraging the compatibility of tables based on co-occurrence statistics. Braunschweig et al. [11] propose a method to normalize web tables in cases where multiple core columns and mixed concepts are detected in one table.

Web tables are embedded in HTML pages, where the surrounding text can help to understand what a given table is about. However, these surrounding sentences are not equally beneficial for table understanding. Wang et al. [70] present the Table-Related Context Retrieval system (TRCR) to determine the relevance between a table and each surrounding sentence. Using TRCR, the most relevant texts are selected to uncover table semantics. Another related study is performed in Reference [28], where NLP tools, like part-of-speech tagging, dependency paths, and named-entity recognition, are explored to mine surrounding texts for understanding table semantics. Braunschweig et al. [9] propose a heuristic approach that extracts text snippets from the context of a

web table, i.e., caption, headline, surrounding text, and full text, which describe individual columns in the table and link these new labels to columns. As a follow-up, Braunschweig et al. [10] propose a contextualization method of splitting table context into paragraphs with consistent topics, providing a similarity measure that is able to match each paragraph to the table in question. Paragraphs are then ranked based on their relevance.

5 TABLE SEARCH

Table search is the task of returning a ranked list of tables in response to a query. It is an important task on its own and is regarded as a fundamental step in many other table mining and extraction tasks as well, like table integration or data completion. Table search functionality is also available in commercial products; e.g., Microsoft Power Query provides smart assistance features based on table search. Depending on the type of the query, table search may be classified as *keyword-based search* and *table-based search*.

5.1 Keyword-based Search

Given a keyword query, the process of returning a ranked list of tables is referred to as *keyword-based search* (or *keyword query search*). One of the first published methods is by Cafarella et al. [14], who implement keyword table search on top of an existing web search engine. Specifically, they extract the top-k tables from the returned webpages. In follow-up work, a similar system called OCTOPUS [13] extends the same method (referred to as SimpleRank) with a reranking mechanism (SCPRank) that considers attribute co-occurrences.

Later works search directly within a designated table corpus. Methods may be divided into *document-based* and *feature-based* approaches. According to the first group of approaches, a document-based representation is created for each table. This might contain all text included in the table or only certain elements of the table (e.g., caption or header labels). Then, these document-based representations may be ranked using traditional retrieval models, such as TF-IDF [56].

Feature-based methods employ supervised machine learning for table ranking. Features may be divided into three main categories: query features, table features and query-table features. *Query features* include query length and IDF scores of query terms. *Table features* characterize the table in terms of its dimensions (number of rows, columns) and schema coherency. With a focus on Wikipedia tables, Bhagavatula et al. [7] introduce features related to the connectivity of the Wikipedia page (pageViews, inLinks, and outLinks) and the table's importance within the page (table importance and table page fraction). Finally, *query-table features* capture the degree of matching between the user's information need and the table. Typically, these include similarity scores between the query and various table elements. Table 10 lists a selection of features for keyword table search. In terms of learning algorithm, Cafarella et al. [14] train a linear regression classifier, while Bhagavatula et al. [7] train a linear ranking model learned with Coordinate Ascent.

Instead of relying on a single keyword query as input, Pimplikar and Sarawagi [56] take q columns, each described by a set of keywords $Q_1, \ldots, Q_q$, as input (e.g., Q_1 = "chemical element," Q_2 = "atomic number," and Q_3 = "atomic weight"), and return a table with q columns as the answer. First, they rank tables using the union of words in $Q_1, \ldots, Q_q$. Then, each table column is labeled with the query column it maps to. Finally, relevant columns and rows are merged into a single table, by considering the table-level relevance scores and the column-level mapping confidence scores. To decide if two rows are duplicates of each other, they employ the method in [29]. Zhang and Balog [82] perform semantic matching between queries and tables for keyword table search. Specifically, they (1) represent queries and tables in multiple semantic spaces (both discrete sparse and continuous dense vector representations) and (2) introduce various similarity measures for matching those semantic representations. For the former, both queries and tables are represented

Table 10. A Selection of Features for Keyword-based Table Search

Query features		Source
QLEN	Number of query terms	[67]
IDF_f	Sum of query IDF scores in field f	[58]
Table features		
#rows	Number of rows in the table	[7, 14]
#cols	Number of columns in the table	[7, 14]
#of NULLs in table	Number of empty table cells	[7, 14]
PMI	ACSDb-based schema coherency score	[14]
inLinks	Number of in-links to the page embedding the table	[7]
outLinks	Number of out-links from the page embedding the table	[7]
pageViews	Number of page views	[7]
tableImportance	Inverse of number of tables on the page	[7]
tablePageFraction	Ratio of table size to page size	[7]
Query-table features		
#hitsLC	Total query term frequency in the leftmost column cells	[14]
#hitsSLC	Total query term frequency in second-to-leftmost column cells	[14]
#hitsB	Total query term frequency in the table body	[14]
qInPgTitle	Ratio of the number of query tokens found in page title to total number of tokens	[7]
qInTableTitle	Ratio of the number of query tokens found in table title to total number of tokens	[7]
yRank	Rank of the table's Wikipedia page in web search engine results for the query	[7]
MLM similarity	Language modeling score between query and multi-field document repr. of the table	[31]

as bag-of-entities, bag-of-categories, word embeddings (trained on Google news) and graph embeddings, respectively. As for the latter, matching methods, they employ the early and late fusion patterns [80]. They consider all possible combinations of semantic representations and similarity measures and use these as features in a supervised learning model. They demonstrate significant and substantial improvements over a state-of-the-art feature-based baseline. Most recently, Deng et al. [21] train word embeddings utilizing the Wikipedia table corpus and achieve comparable results.

5.2 Table-based Search

Table search is not limited to keyword queries. The input may be also be a table, in which case the task of returning related tables is referred to as *table-based search* (or *query by table*). At its core, this task boils down to computing a similarity score between the input and candidate tables, which we shall refer to as *table matching*. Search by table may be performed for different goals: (1) to be presented to the user to answer her information need [20, 44, 53] and (2) to serve as an intermediate step that feeds into other tasks, like table augmentation [2, 41, 51, 76].

One group of approaches addresses the table matching task by using certain table elements as a keyword query, and scoring tables using keyword-based methods. For example, Ahmadov et al. [2] use table entities and table headings as queries to retrieve a ranked list of tables for data completion (to be detailed in Section 8.3). The two ranked lists are then intersected afterwards to arrive at a more complete result set.

Table 11. Overview of Table Elements Used When Querying by Table for Various Table-related Applications

Reference	Application	T_E	T_H	$T_{[:,j]}$	T_p	$T_{[i,j]}$
Ahmadov et al. [2]	Data completion	✓	✓			
Das Sarma et al. [20]	Schema complement	✓	✓			
	Entity complement	✓				
Lehmberg et al. [41]	Relation join		✓			
Limaye et al. [44]	Table cell retrieval			✓		✓
Nargesian et al. [51]	Table union search	✓		✓		✓
Nguyen et al. [53]	Diverse table search		✓			✓
Yakout et al. [76]	Table augmentation		✓	✓	✓	✓
Zhang and Balog [85]	Table recommendation	✓	✓	✓	✓	✓

More commonly, table matching is tackled by dividing tables into various elements (such as table caption, table entities, column headings, cell values), then computing element-level similarities. Table 11 provides an overview of the table elements that have been utilized in past work. It is worth pointing out that in most of these cases, table search is not the ultimate goal, it is only used as a component in a larger application. The Mannheim Search Join Engine [41] seeks to extend the input table with additional attributes. It utilizes table headings by comparing the column headings between the input table and candidate tables. Specifically, they first filter tables that share at least one column heading with the input table, using exact term matching. Then, the table matching score is computed by (1) building an edit distance similarity matrix between the input and candidate tables' column headings, and (2) calculating the Jaccard similarity of the two tables using the matrix's maximum weighted bipartite matching score. Similar to the Mannheim Search Join Engine that is based on table headings, Nargesian et al. [51] search tables that are likely unifiable with the seed table, which is called *attribute union ability*. Nargesian et al. [51] formalize three statistical models to estimate the likelihood that two attributes contain values that are in the same domain. The simplest case, named *set domains*, uses the size of the intersection of values between two columns. The second case, called *semantic domains*, measures the semantic similarity between the values by mapping the columns to classes, e.g., entities. For values that are expressed in natural language, the third case of *natural language domains* measures semantics based on natural langue rather than on ontologies. They use word embeddings trained based on Wikipedia documents to define natural language domains and statistical tests between the vectors are used to evaluate the likelihood that two attributes are from the same domain. Das Sarma et al. [20] aim to find related tables for augmenting the input table with additional rows or columns, referred to as *entity complement* and *schema complement*, respectively. Entity complement considers the relatedness between entity sets of the input and candidate tables. Relatedness between two entities is estimated by representing entities as weighed label sets (from a knowledge base or from a table corpus) and taking their dot product. Das Sarma et al. [20] propose multiple methods to aggregate pairwise entity relatedness scores for computing relatedness between two sets of entities. Schema complement combines two element-wise similarities: table entities and column headings. The former considers the overlap between table entities. The latter estimates the benefits of adding a column from the candidate table to the input table by determining the consistency between the new column and the existing columns of the input table. Yakout et al. [76] propose InfoGather, a holistic method for matching tables to support three core operations: augmentation by column headings, augmentation by example, and column heading discovery. They consider element-wise similarities, including table context, URL, tuples, column headings, column

values, and table data, as well as cross-element similarity between table and context. Similarity is measured using the vector product of TF/IDF-weighted term vectors. Then, element-level similarity scores are combined as features in a machine learned model. In follow-up work, InfoGather is extended as InfoGather+ [78] to incorporate tables with numeric and time-varying attributes. Zhang and Balog [85] perform table matching by representing table elements in multiple semantic spaces, and then combining element-level similarities using a discriminative learning model.

Nguyen et al. [53] consider the diversity of the returned tables. They focus on two table elements: column headings and table data. The former is similar in spirit to the Mannheim Search Join Engine [41]. The latter works by measuring the similarity between table columns, which are represented as term frequency vectors.

Unlike the above methods, which consider tables as the unit of retrieval, Limaye et al. [44] return a ranked list of cells as result. They train a machine learning method for annotating (1) entities in tables cells, (2) columns with types, and (3) relations between columns. Then, search is performed by issuing an automatically generated structured query.

6 QUESTION ANSWERING ON TABLES

Tables are a rich source of knowledge that can be utilized for answering natural language questions. This problem has been investigated in two main flavors: (1) where the table, which contains the answer to the input question, is given beforehand [55], and (2) where a collection of tables are to be considered [65]. The latter variant shares many similarities with traditional question answering (QA), while the former requires different techniques. One of the main challenges of QA on tables, shared by both scenarios, is how to match the unstructured query with the (semi-)structured information in tables. Question answering on tables is also closely related to work on natural language interfaces to databases, where the idea is that users can issue natural language queries, instead of using formal structured query languages (like SQL), for accessing databases [3, 42, 43, 57]. *Semantic parsing* is the task of parsing natural language queries into a formal representation. Semantic parsing is often used in question answering, by generating logical expressions that are executable on knowledge bases [6, 25].

6.1 Using a Single Table

We first discuss approaches that take a single table as input (sometimes referred to as *knowledge base table* [77]), and seek the answer to the input question in that table. The basic idea is to regard the input table as a knowledge base, which poses a number of challenges. First, knowledge bases contain a canonicalized set of relations, while tabular data is much more noisy. Second, traditional semantic parsing sequentially parses natural language queries into logical forms and executes them against a knowledge base. To make them executable on tables, special logical forms are required. Last, semantic parsing and query execution become complicated for complex questions as carefully designed rules are needed to be able to parse them into logical forms. Pasupat and Liang [55] propose to answer complex questions, involving operation such as comparison, superlatives, aggregation, and arithmetics to address above problems. They convert the input table into a knowledge graph by taking table rows as row nodes, strings as entity nodes, and columns as directed edges. The column headings are used as predicates. Numbers and strings are normalized following a set of manual rules. Being one of the earliest works addressing this task, Pasupat and Liang [55] follow a traditional parser design strategy. A semantic parser is trained based on a set of question-answer pairs. Given a table and a question, a set of candidate logical forms is generated by parsing the question. Then, logical forms are ranked using a feature-based representation, and the highest ranked one is applied on the knowledge graph to obtain the answer. Pasupat and Liang [55] develop a dataset, called WikiTableQuestion, which consists of a random sample of

2,100 tables from Wikipedia and 22,000 question-answer pairs. The proposed approach is found to suffer from a coverage issue, i.e., it is able to answer only 20% of the queries that have answers in Freebase.

Different from semantic parsing methods that require predefined logical operations, Yin et al. [77] propose a neural network architecture, named Neural Enquirer, for semantic parsing with a specific table. Neural Enquirer is a fully neural system that generates distributed representations of queries and tables (called query encoder and table encoder, respectively). Then, the question is executed through a series of query operations, called executors, with intermediate execution results computed in the form of table annotations at different levels. Training can be performed in an end-to-end fashion or carried out using step-by-step supervision (for complex queries). They use query answers as indirect supervision, but jointly perform semantic parsing and query execution in distributional spaces. The distributed representations of logical forms are learned in end-to-end tasks, which is based on the idea of adopting the results of the query execution as indirect supervision to train the parser. It is worth pointing out that this model makes a number of strong assumptions. For example, they consider four specific types of queries, provide the logical form template for each, and carefully and manually select a table that is supplied as part of the input. Similar to Reference [77], Neelakantan et al. [52] attempt to solve the task of question answering on tables using neural networks, a system called Neural Programmer. Neural Programmer runs for T steps and the final result is formed step by step. The model adopts a Recurrent Neural Network (RNN) architecture to process the input question, a selector to assign probabilities to a set of possible operations and data segments at each step, and a history RNN to remember the previous operation and data selections. Providing a small set of basic operations, they also take the result of correct executions as indirect supervision. During the training, adding random noise to the gradient greatly improves performance as the operations and data selections are quite heterogenous.

6.2 Using a Collection of Tables

Another line of work focuses on answering questions using a collection of tables. These approaches are more similar to traditional question answering on text, comprising of candidate identification, query-type prediction, and ranking components. The main differences are twofold. One is schema matching, which is the same as before (in Section 6.1), but there is an additional normalization issue across tables here. The other is the need for extracting quantity values from tables. Sarawagi and Chakrabarti [62] show that over 40% of table columns contain numeric quantities, and propose a collective extraction framework to extract quantities from raw web tables based on a consensus model. Their system, called QEWT, extends the work of Banerjee et al. [5] to tables. They employ keyword-based table ranking to fetch table candidates. This corresponds to the candidate snippet/answer identification step in traditional QA. QEWT can answer quantity-target queries with a ranked list of quantity distributions, which are taken from the tables. It uses a table column unit annotator based on probabilistic context free grammars for easily extracting quantities from table columns to deal with ambiguity (both for headings and for values). From an information retrieval perspective, quantity queries on web tables is the task of returning a ranked list of quantities for a query. QEWT employs a quantity response model for this task.

Inspired by classical textual QA, Sun et al. [65] decompose table cells into relational chains, where each relational chain is a two-node graph connecting two entities. Specifically, each row of a table represents relations among the cells. They construct a pseudo-node to connect all the cells and take the headings to label relationships. Any pair of cells in the same row form a directional relational chain. The input query is also represented as a two-node graph question chain, by identifying the entities using an entity linking method. The task then boils down to finding the relational chains that best match the question chain. This matching is performed using deep

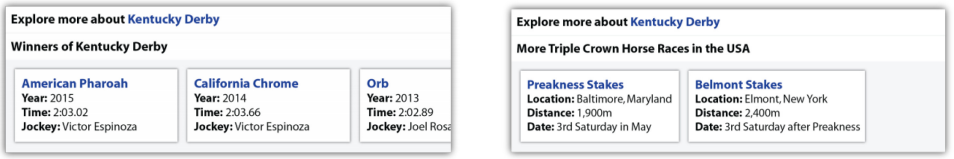

Fig. 5. Illustration in Reference [17], showing an example of knowledge exploration for the query of "kentucky derby" through Knowledge Carousels: (a) a downward showing the winners of Kentucky Derby; (b) a sideway representing the famous Triple Crown horse races in the US, of which Kentucky Derby is a member.

neural networks, in particular the Convolutional Deep Structured Semantic Model (C-DSSM) [64], to overcome the vocabulary gap limitation of bag-of-words models. They find that combining the deep features with some shallow features, like term-level similarity between query and table chains, achieve the best performance. Sun et al. [65] conclude that their method can complement KB-based QA methods by improving their coverage.

7 KNOWLEDGE BASE AUGMENTATION

The knowledge extracted from tabular data can be used for enriching knowledge bases. First, we present an approach that is devised for exploring the knowledge contained in web tables. Then, we discuss methods for knowledge base augmentation using tabular data.

7.1 Tables for Knowledge Exploration

The knowledge contained in web tables can be harnessed for knowledge exploration. Knowledge Carousels [17] is the first system addressing this, by providing support for exploring "is-A" and "has-A" relationships. These correspond to two kinds of entity-seeking queries (queries searching for attributes and relationships of entities), called "sideways" and "downwards," respectively. Given an input entity, Chirigati et al. [17] utilize web tables to select the carousel type, create a set entities of this carousel, generate human-readable titles, and rank carousels based on popularity and relatedness extracted from tables. See Figure 5 for an illustration.

7.2 Knowledge Base Augmentation and Construction

Tabular data on the Web can be used to construct new or augment existing knowledge bases.

7.2.1 Knowledge Base Augmentation. In Section 4, we have presented techniques for interpreting tables with the help of knowledge bases. The obtained annotations, in turn, can contribute to extending those knowledge bases. *Knowledge base augmentation*, also known as *knowledge base extension*, is concerned with generating new instances of relations using tabular data and updating knowledge bases with the extracted information.

Knowledge bases need to be complete, correct, and up-to-date. A precondition of extending knowledge bases using web tables is matching them to those existing knowledge bases. Specific matching problems include *table-to-class matching*, *row-to-instance matching*, and *attribute-to-property matching*. Ritze et al. [60] propose an iterative matching method, T2K, to match web tables to DBpedia for augmenting knowledge bases. They also develop and make publicly available the T2D dataset for matching, consisting of 8,700 schema-level and 26,100 entity-level correspondences between web tables and DBpedia, which are extracted and annotated manually. The T2K method utilizes the T2D dataset to execute iterative steps between candidate matching and property matching, to find proper entities/schemas in DBpedia for table rows/columns. However, T2D mainly focuses on large tables and does not work that well for small-sized tables [39]. To counter

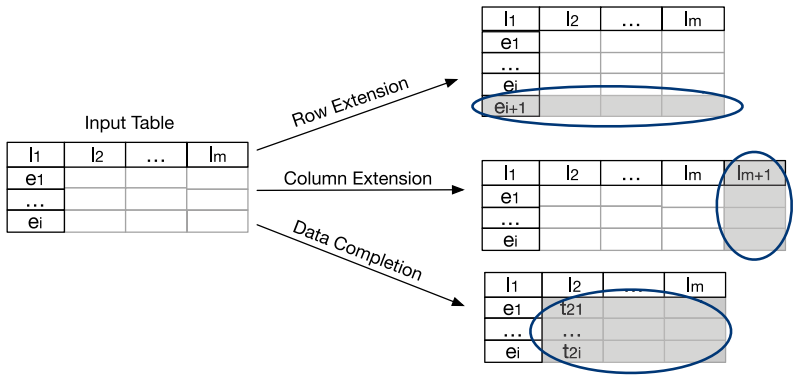

Fig. 6. Illustration of three table augmentation tasks: row extension, column extension, and data completion.

this problem, Lehmberg and Bizer [39] propose to combine tables from each website into larger tables for table matching, building on the intuition that tables from the same website are created in a similar fashion.

Strictly speaking, we classify the work in Reference [69] as *row extension*. Nevertheless, since they map table entities to a knowledge base with the purpose of collecting more entities from other tables that belong to the same concept in the knowledge base, their work can also be regarded as a knowledge base augmentation approach.

7.2.2 Knowledge Base Construction. Instead of augmenting existing knowledge bases, web tables contain abundant information to be turned into knowledge bases themselves.

Even though there exists a number of large-scale knowledge bases, they are still far from complete [22]. Therefore, Dong et al. [22] introduce a web-scale probabilistic knowledge base named Knowledge Vault (KV) that fuses different information sources. For web tables, Dong et al. [22] firstly identify the relation that is expressed in a table column by checking the column's entities, and reason about which predicate each column could correspond to. This latter task is approached using a standard schema matching method [68], with Freebase as the knowledge base. The extracted relations, together with relational data from other sources, are converted into RDF triples, along with associated confidence scores. The confidence scores are computed based on a graph-based method. Specifically, the triples are fused by machine learning methods from multiple sources, including an existing knowledge base, (i.e., Freebase) and web tables. Consequently, 1.6B triples are generated, of which 324M have a confidence score above 0.7 and 271M have a confidence score above 0.9.

8 TABLE AUGMENTATION

Table augmentation refers to the task of extending a seed table with more data. Specifically, we discuss three tasks in this section: row extension (Section 8.1), column extension (Section 8.2), and data completion (Section 8.3). See Figure 6 for an illustration. One might envisage these functionalities being offered by an intelligent agent that aims to provide assistance for people working with tables [81].

8.1 Row Extension

Row extension aims to extend a given table with more rows or row elements (see Figure 7). It mainly focuses on a particular type of table, namely, relational tables. More specifically, row extension primarily targets horizontal relational tables, where rows represent entities and columns describe

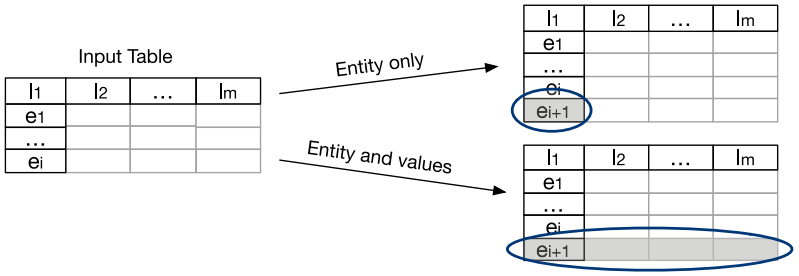

Fig. 7. Illustration of row extension by adding only an entity (Upper) or an entire row, i.e., an entity as well as cell values (Lower). Existing work has focused on the former task.

Table 12. Overview of Row Population Methods

	Data		Tasks	
Reference	**KB**	**Tables**	**Table search**	**Row population**
Das Sarma et al. [20]		✓	✓	
Wang et al. [69]		✓	✓	✓*
Yakout et al. [76]		✓	✓	✓
Zhang and Balog [81]	✓	✓	✓	✓

* Originally developed for concept expansion, but can be used for row population.
Notice that table search is inherently involved.

the attributes of those entities. In such tables there usually exists a *core column* (or *key column*) containing mostly entities [8, 68]. Instead of directly providing a complete tuple (row), existing work has focused on identifying entities for populating such core columns (i.e., the Upper scenario in Figure 7). Table 12 provides an overview of methods that will be covered below. As we shall see, table search is inherently involved here.

Populating entities in the core column of a table is similar to the problem of *concept expansion*, also known as *entity set expansion*, where a given set of seed entities is to be completed with additional entities [12, 34, 46, 47]. Existing methods for concept expansion suffer from two main issues: input ambiguity and semantic drift (i.e., entities belonging to different concepts are mixed during expansion). Motivated by the intuition that tables tend to group entities that belong to a coherent concept, Wang et al. [69] leverage web tables for the concept expansion task, thereby aiming to prevent semantic drift. They provide both the seed entities as well as a concept name as input. First, they retrieve tables related to the seed entities. Then, they use a graph-based ranking method to rank candidate entities that co-occur with the seed entities in those tables. Specifically, they first expand the set by iteratively adding the most relevant tables based on concept likelihood, and collecting entities there. Then, they refine the earlier estimation and remove less relevant tables based on more complete information. Wang et al. [69] find that adding an input concept can address the semantic drift problem for tail concepts. While this method is developed for concept expansion, it is directly applicable to the problem of populating entities in a core column.

Das Sarma et al. [20] search for *entity complement* tables that are semantically related to entities in the input table (as we have already discussed in Section 5.2). Then, the top-k related tables are used for populating the input table. Das Sarma et al. [20], however, stop at the table search task. A similar approach is taken in InfoGather [76], where this task is referred to as the *augmentation by example* operation. There, they first search for related tables (cf. Section 5.2), and then consider entities from these tables, weighted by the table relatedness scores. Yakout et al. [76] build a schema matching graph among web tables, based on pairwise table similarity. Despite the use of

Table 13. Overview of Column Population Methods

Reference	Task	Data			Output	
		Web tables	WP tables	KBs	T_H	$T_H + T_{[:,j]}$
Bhagavatula et al. [7]	Relevant join		✓			✓
Cafarella et al. [14]	Schema auto-completion	✓			✓	
Das Sarma et al. [20]	Schema complement	✓				
Lehmberg et al. [41]	Search join	✓	✓	✓		✓
Zhang and Balog [81]	Column population		✓		✓	

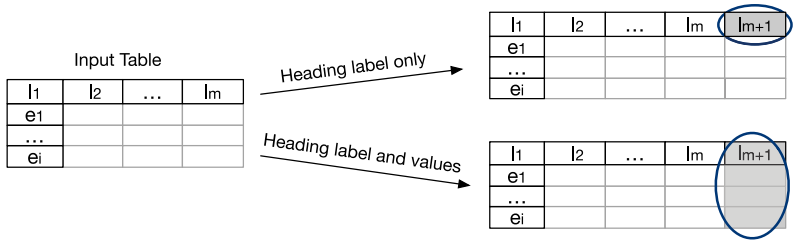

Fig. 8. Illustration of column extension by adding only a heading label (Upper) or an entire column, i.e., heading label and cell values (Lower).

scalable techniques, this remains to be computationally very expensive, which is a main limitation of the approach. Instead of relying only on related tables from a table corpus, Zhang and Balog [81] also consider a knowledge base (DBpedia) for identifying candidate entities. Specifically, they collect entities sharing the same types or categories with the input entities from DBpedia, and entities from similar tables (i.e., tables sharing seed entities, having similar captions, or including the same headings) as candidates. They find that entity-type information in DBpedia is too general to help identify relevant candidates, and end up using only category information when extracting candidates from DBpedia. It is also shown that using related tables and using a knowledge base are complementary when identifying candidate entities. They develop a generative probabilistic model for the subsequent ranking of candidate entities based on their similarity to (1) other entities in the table, (2) column headings, and (3) the caption of the input table. Among the three table elements, seed entities are the most important component for entity ranking, followed by table headings and caption. A combination of the three table elements performs the best in the end. In recent work, Deng et al. [21] utilize Word2vec to train table embeddings for core column entities. Combining the embedding-based similarity scores with the probability-based scores from [81] results in further performance improvements.

8.2 Column Extension

The most widely studied subtask in table augmentation is *column extension*: extending a table with additional columns. This task roughly corresponds to the *join* operation in databases. In this context, the set of column heading labels is also often referred to as the table *schema*. Commonly, column extension is approached by first locating similar tables and then considering the column headings/values in those tables. Table 13 provides an overview of the methods discussed in this section.

One particular variant of column extension aims to identify additional column heading labels (see Figure 8 (Upper)). As table columns often correspond to entity attributes, this task is also referred to as *attribute discovery* [76] or *schema auto-complete* Cafarella et al. [14]. The WebTables

system [14] implements this functionality based on the *attribute correlation statistics database* (ACSDb). ACSDb contains frequency statistics of attributes and co-occurring attribute pairs in a table corpus. ACSDb comprises 5.4M unique attribute names and 2.6M unique schemas. With these statistics at hand, the next probable attribute can be chosen using a greedy algorithm. The statistics-based method in Reference [14] was the first approach to column extension, and was found to be able to provide coherent heading suggestions. However, later research has proven that considering additional features can further improve performance. Das Sarma et al. [20] focus on finding related tables, with the aim of schema complement. For ranking tables, they consider two factors: (1) the coverage of entities, and (2) the potential benefits of adding additional attributes from those tables (we discussed the table search method in Section 5.2). Again, they stop at the table search task. The task of identifying potential attributes or column labels is also known as *schema matching* [41] or *column population* [81]. Zhang and Balog [81] try to find the headings that can be placed as the next column in an input table. They first find candidate headings from similar tables (the same strategy that they also use for row population). Zhang and Balog [81] observe that input entities and table caption contribute comparably to the identification of relevant candidates, while table headings are the least important component. However, similar to row population, all these sources are complementary, i.e., each source can identify candidate headings that none of the others could. In a subsequent ranking step, the candidates are scored based on table similarity, by aggregating element-wise similarities between (corresponding elements of) the input table and related tables. In Reference [21], they utilize Word2vec to train embeddings for table headings. Similar to row population, combining the embedding similarity scores with the probabilities from [81] yields further performance improvements. The above approaches differ in what they use as input, i.e., whether they use only table headings [14, 41] or the entire table [20, 81].

Another variant attempts to augment the input table with entire columns, that is, including both the heading label as well as the corresponding cell values for each row within that column (see Figure 8 (Lower)). Bhagavatula et al. [7] present the *relevant join* task, which returns a ranked list of column triplets for a given input table. Each triplet consists of *SourceColumn*, *MatchedColumn*, and *CandidateColumn*. *SourceColumn* is from the query table, while *MatchedColumn* and *CandidateColumn* are from the candidate tables. They propose a semantic relatedness measure to find candidate tables from related Wikipedia pages, where page relatedness is estimated based on in-link intersections. Their idea is to compute similarity between columns, such that if *SourceColumn* and *MatchedColumn* share largely similar values, then the input table may be extended with *CandidateColumn*. These candidate columns are classified as relevant or non-relevant, using a linear ranking model, before performing the actual join. To reduce the number of candidate columns, some are filtered out in a pre-processing stage using simple heuristics. Columns that are kept are required to be non-numeric, have more than four rows, and an average string length larger than four. Bhagavatula et al. [7] find that columns containing numeric data make more relevant additions than non-numeric ones. Additionally, more distinct values in the *SourceColumn* and a higher match percentage lead to better quality joins. The join operation is also supported by the Mannheim Search Join Engine [41]. It first searches for related tables based on column headings (cf. Section 5.2), then applies a series of left outer joins between the query table and the returned tables. Afterwards, a consolidation operation is performed to combine attributes. Specifically, they employ a matching operator that relies on data from knowledge bases. Given two columns, similar match (Levenshtein distance) and exact match are used for matching headings. Lehmberg et al. [41] observe that similar match returns on average 3.4 times more tables than exact match. Among different table corpora, web tables provide the largest number of relevant tables, and Wikipedia tables tend to be populous on certain topics, such as countries and films.

Table 14. Overview of Data Completion Methods

Reference	Data		Output	
	Tables	Web	$T_{[:,j]}$	$T_{[i,j]}$
Ahmadov et al. [2]	✓	✓	✓	✓
Cafarella et al. [13]	✓		✓	
Yakout et al. [76]	✓		✓	
Zhang and Chakrabarti [78]	✓		✓	
Zhang and Balog [84]	✓	✓		✓

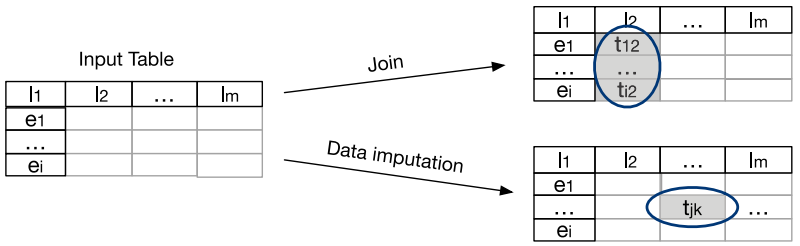

Fig. 9. Illustration of data completion tasks: join (Upper) and data imputation (Lower).

8.3 Data Completion

Data completion for tables refers to the task of filling in the empty table cells. Table 14 summarizes the methods we discuss here. One variant of this task attempts to find the cell values for an entire column (see Figure 9 (Upper)). This is known as the *augmentation by attribute name* operation in the InfoGather system [76]. This is typical of a scenario where the core entity column as well as the column headings are given in a relational table, and the values for the corresponding attributes (*augmenting attributes*) are to be filled in. The system in Reference [76] takes the incomplete table as input to search for matching tables, then extracts attribute values from those tables. It is worthwhile to point out that InfoGather focuses on finding values that are entities. An extended version of the system, InfoGather+ [78], focuses on numerical and time-varying attributes. They use undirected graphical models and build a semantic graph that labels columns with units, scales, and timestamps, and computes semantic matches between columns. Their experiments are conducted on three types of tables: company (revenue and profit), country (area and tax rate) and city (population). Zhang and Chakrabarti [78] find that the conversion rules (manually designed unit conversion mapping) achieve higher coverage than string-based schema matching methods. Similar to InfoGather's *augmentation by attribute name* operation, the *extend* operation in the OCTOPUS systems [13] enables the user to add more columns to a table by performing a join. It takes a keyword query and a given (existing) table column as input, where the keyword describes the newly added column. Different from a regular join, the added column is not necessarily an existing column. It may be formed row-by-row by combining information from multiple related tables (see Section 5.1 for the table search operation). However, Cafarella et al. [13] rely on simple methods like edit distance for schema matching, which leaves room for improvement.

Another flavor of the data completion task focuses on filling in missing values for individual cells, referred to as *data imputation* (see Figure 9 (Lower)). Ahmadov et al. [2] present a hybrid imputation method that combines a lookup-based approach, based on a corpus of web tables, and a model-based approach that uses machine learning (e.g., k-nearest neighbors or linear regression) to predict the value for a missing cell. It is worth noting that all the above methods rely only on

tables and ignore the cases where no similar tables can be found. The method in Reference [2] is shown to be able to improve coverage. However, being able to automatically decide when to do simple lookup and when to employ a machine learned model remains an open challenge. CellAutoComplete is a recent framework proposed by Zhang and Balog [84] to tackle several novel aspects of this problem, including: (1) enabling a cell to have multiple, possibly conflicting values, (2) supplementing the predicted values with supporting evidence, (3) combining evidence from multiple sources, and (4) handling the case where a cell should be left empty. This framework makes use of a large table corpus and a knowledge base as data sources, and consists of preprocessing, candidate value finding, and value ranking components.

9 CONCLUSIONS AND FUTURE DIRECTIONS

Tables are a powerful and popular tool for organizing and manipulating data. Research on web tables has seen nearly two decades of development. During this long period, the research focus has evolved considerably, from low-level tasks (table extraction) to tapping more and more into the actual knowledge contained in tables (for search and for augmenting existing knowledge bases). Below, we review past progress and identify open research directions for each of the six main categories of tasks.

9.1 Table Extraction

In the early years, research was mainly focused on detecting, identifying, and extracting tables from webpages, and classifying them according to some type taxonomy. Gradually, spreadsheet documents were also considered for table extraction, and type taxonomies became more fine-grained. With the advancement of table extraction and classification methods, large-scale table corpora were constructed, which became available as resources to be utilized in other tasks. One open issue is that the available table corpora are all a result of a one-off extraction effort. As such, these collections get quickly outdated.

9.2 Table Interpretation

The problem of uncovering table semantics, including but not limited to identifying table column types, linking entities in tables, and extracting relational data from tables, represents an active research area. It is also an important one, as the resulting semantic annotations are heavily utilized in other table-related tasks, such as knowledge base augmentation and question answering. While there exist methods for high-precision extraction, there is plenty of room for improvement in terms of recall, as most existing methods can only interpret a small portion of tables. For instance, Ritze et al. [61] find that only 2.85% of web tables can be matched to DBpedia. Further, most of the emphasis has been on relational tables; other table types (e.g., entity tables) bring about a different set of challenges. Another line of future research concerns the development of user interfaces and tools for facilitating and visualizing the annotations [45].

9.3 Table Search

The task of retrieving relevant tables from a table corpus for an input query is one of the core tasks that was started in the early days and remains to be an active research topic ever since. One limitation of existing work is that it often makes assumptions about underlying query intent and the preferred answer table types. For example, Zhang and Balog [82] assume that queries follow a class-property pattern, which can be successfully answered by relational tables. As a result, relational tables with this pattern are preferred, which might therefore result in lower coverage. TableNet [27], a recent study on the interlinking of tables with subPartOf and equivalent

relations, can provide a better understanding of table patterns. In the future, it would be desirable if an automatic query intent classifier were to identify the type of result table sought, which does not need to be limited to relational tables. Another topic that deserves attention in our opinion, but has not been explored yet, is the presentation of table search results. For example, for large tables, how should appropriate snippets (summaries) be generated for search result pages? Following the two main lines of approaches to automatic summarization, summaries could be extraction-based, by selecting relevant portions of the table to be displayed, or abstraction-based, by generating a natural language summary of its contents [30].

9.4 Question Answering

Facts extracted from tabular data can be used for answering natural language questions. Previous work has looked at answering questions using a single table or multiple tables. Much of the research emphasis has been directed to parsing the questions and on extracting the facts from tables. While studying these, certain simplifications were made concerning other aspects of the problem. For example, works that address QA on a single table all take a carefully selected table (which is to be treated as a knowledge base) for granted. Locating a proper KB table is a challenging table search task that remains to be solved. There also seems to be a lack of understanding of when tables can actually aid QA. Existing research has found that even though QA on tables suffers from low coverage, it can complement QA on text. Yet, there has not been any systematic study on understanding what are the types of questions where tables can help or what is the scope of facts or relations where web tables have sufficient coverage. Finally, the heterogeneity of web tables limits the applicability of current methods to a small portion of tables. In the future, more generic methods would need to be developed to be able to deal with heterogeneous tables.

9.5 Knowledge Base Augmentation

The knowledge mined from tables can (also) be utilized for knowledge base (KB) population/construction. Existing methods address one of two main problems: (1) matching tables to a knowledge base or (2) discovering new facts/relations from tables by utilizing these "table-to-KB" matching results. Yet, all these approaches seem to ignore "out-of-KB" data, that is, entities or properties that are not linked to a knowledge base. Wikipedia tables are one specific example that contain many unlinked entity mentions. Those entities/properties could also be potentially useful for populating KBs with new information. Shortcomings of current approaches include (1) the lack of consideration for temporal information and (2) identifying entities at the right level of granularity (e.g., location may be given as a city or as a state or country) [61]. The former is especially important, as it may promote further utilization of tables to help keep KBs up-to-date.

9.6 Table Augmentation

There is a solid body of work on augmenting existing tables with additional data, extracted either from other tables or from knowledge bases. However, there are at least two issues that remain. One is tapping into the large volumes of unstructured sources (e.g., webpages). The other is combining data from multiple sources, which brings about a need for techniques to draw users' attention to conflicting information and help them to deal with those cases. Existing work on augmentation assumes the presence of an input table that the user needs helps with completing. An and ambitious research direction is to automatically generate a table, which can answer the user's information need, from scratch [83].

REFERENCES

[1] Yanif Ahmad, Tudor Antoniu, Sharon Goldwater, and Shriram Krishnamurthi. 2003. A type system for statically detecting spreadsheet errors. In *Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE'03)*. 174–183.

[2] Ahmad Ahmadov, Maik Thiele, Julian Eberius, Wolfgang Lehner, and Robert Wrembel. 2015. Towards a hybrid imputation approach using web tables. In *Proceedings of the IEEE 2nd International Symposium on Big Data Computing (BDC'15)*. 21–30.

[3] Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases—An introduction. *CoRR* cmp-lg/9503016.

[4] Sreeram Balakrishnan, Alon Y. Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying WebTables in practice. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR'15)*.

[5] Somnath Banerjee, Soumen Chakrabarti, and Ganesh Ramakrishnan. 2009. Learning to rank for quantity consensus queries. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*. 243–250.

[6] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*. 1533–1544.

[7] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2013. Methods for exploring and mining tables on Wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA'13)*. 18–26.

[8] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: Entity linking in web tables. In *Proceedings of the 14th International Conference on The Semantic Web (ISWC'15)*. 425–441.

[9] Katrin Braunschweig, Maik Thiele, Julian Eberius, and Wolfgang Lehner. 2015. Column-specific context extraction for web tables. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC'15)*. 1072–1077.

[10] Katrin Braunschweig, Maik Thiele, Elvis Koci, and Wolfgang Lehner. 2016. Putting web tables into context. In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K'16)*. 158–165.

[11] Katrin Braunschweig, Maik Thiele, and Wolfgang Lehner. 2015. From web tables to concepts: A semantic normalization approach. In *Proceedings of the Conceptual Modeling (IC3K'16)*. 247–260.

[12] Marc Bron, Krisztian Balog, and Maarten de Rijke. 2013. Example-based entity search in the web of data. In *Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR'13)*. 392–403.

[13] Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data integration for the relational web. *Proc. VLDB Endow.* 2, 1 (Aug. 2009), 1090–1101.

[14] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: Exploring the power of tables on the web. *Proc. VLDB Endow.* 1, 1 (Aug. 2008), 538–549.

[15] Michael J. Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu 0002. 2008. Uncovering the relational web. In *Proceedings of the 11th International Workshop on the Web and Databases (WebDB'08)*.

[16] Zhe Chen and Michael Cafarella. 2013. Automatic web spreadsheet data extraction. In *Proceedings of the 3rd International Workshop on Semantic Search Over the Web (SS'13)*. 1–8.

[17] Fernando Chirigati, Jialu Liu, Flip Korn, You (Will) Wu, Cong Yu, and Hao Zhang. 2016. Knowledge exploration using tables on the web. *Proc. VLDB Endow.* 10, 3 (Nov. 2016), 193–204.

[18] E. F. Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM* 13, 6 (June 1970), 377–387.

[19] Eric Crestan and Patrick Pantel. 2011. Web-scale table census and classification. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM'11)*. 545–554.

[20] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. 817–828.

[21] Li Deng, Shuo Zhang, and Krisztian Balog. 2019. Table2Vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 1029–1032.

[22] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. 601–610.

[23] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. 2015. Building the dresden web table corpus: A classification approach. In *Proceedings of the 2nd IEEE/ACM International Symposium on Big Data Computing (BDC'15)*. 41–50.

[24] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. 2017. Matching web tables with knowledge base entities: From entity lookups to entity embeddings. In *Proceedings of the 16th International Semantic Web Conference (ISWC'17)*. 260–277.

[25] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. 1156–1165.

[26] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. 2014. A hybrid machine-crowdsourcing system for matching web tables. In *Proceedings of the IEEE 30th International Conference on Data Engineering (ICDE'14)*. 976–987.

[27] Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. TableNet: An approach for determining fine-grained relations for wikipedia tables. In *Proceedings of the World Wide Web Conference (WWW'19)*. 2736–2742.

[28] Vidhya Govindaraju, Ce Zhang, and Christopher Ré. 2013. Understanding tables in context using standard NLP toolkits. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*. 658–664.

[29] Rahul Gupta and Sunita Sarawagi. 2009. Answering table augmentation queries from unstructured lists on the web. *Proc. VLDB Endow.* 2, 1 (Aug. 2009), 289–300.

[30] Braden Hancock, Hongrae Lee, and Cong Yu. 2019. Generating titles for web tables. In *Proceedings of the World Wide Web Conference (WWW'19)*. 638–647.

[31] Faegheh Hasibi, Krisztian Balog, Darío Garigliotti, and Shuo Zhang. 2017. Nordlys: A toolkit for entity-oriented and semantic search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17)*. 1289–1292.

[32] Oktie Hassanzadeh, Michael J. Ward, Mariano Rodriguez-Muro, and Kavitha Srinivas. 2015. Understanding a large corpus of web tables through matching with knowledge bases: An empirical study. In *Proceedings of the Workshop on Linked Data on the Web Co-located with the International World Wide Web Conference (CEUR'15)*, Vol. 1545. 25–34.

[33] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek Narasayya, and Surajit Chaudhuri. 2018. Transform-data-by-example (TDE): An extensible search engine for data transformations. *Proc. VLDB Endow.* 11, 10 (June 2018), 1165–1177.

[34] Yeye He and Dong Xin. 2011. SEISA: Set expansion by iterative similarity aggregation. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*. 427–436.

[35] Vu Hung, Boualem Benatallah, and Regis Saint-Paul. 2011. Spreadsheet-based complex data transformation. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM'11)*. 1749–1754.

[36] Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. 2016. Making sense of entities and quantities in web tables. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM'16)*. 1703–1712.

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

[38] Oliver Lehmberg and Christian Bizer. 2016. Web table column categorisation and profiling. In *Proceedings of the 19th International Workshop on Web and Databases (WebDB'16)*. 4:1–4:7.

[39] Oliver Lehmberg and Christian Bizer. 2017. Stitching web tables for improving matching quality. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1502–1513.

[40] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web (WWW'16 Companion)*. 75–76.

[41] Oliver Lehmberg, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, and Christian Bizer. 2015. The mannheim search join engine. *Web Semant.* 35, P3 (Dec. 2015), 159–166.

[42] Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.* 8, 1 (Sept. 2014), 73–84.

[43] Yunyao Li, Huahai Yang, and H. V. Jagadish. 2005. NaLIX: An interactive natural language interface for querying XML. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*. 900–902.

[44] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types, and relationships. *Proc. VLDB Endow.* 3, 1–2 (Sept. 2010), 1338–1347.

[45] Suvodeep Mazumdar and Ziqi Zhang. [n.d.]. A tool for creating and visualizing semantic annotations on relational tables. In *Proceedings of the 4th International Workshop on Linked Data for Information Extraction Co-located with 15th International Semantic Web Conference (ISWC'19)*.

[46] Steffen Metzger, Ralf Schenkel, and Marcin Sydow. 2013. QBEES: Query by entity examples. In *Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management (CIKM'13)*. 1829–1832.

[47] Steffen Metzger, Ralf Schenkel, and Marcin Sydow. 2014. Aspect-based similar entity search in semantic knowledge graphs with diversity-awareness and relaxation. In *Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Volume 1 (WI-IAT'14)*. 60–69.

[48] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. 2014. Using linked data to mine RDF from Wikipedia's tables. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM'14)*. 533–542.
[49] Varish Mulwad, Tim Finin, and Anupam Joshi. 2013. Semantic message passing for generating linked data from tables. In *Proceedings of the 12th International Semantic Web Conference, Part I (ISWC'13)*. 363–378.
[50] Varish Mulwad, Tim Finin, Zareen Syed, and Anupam Joshi. 2010. Using linked data to interpret tables. In *Proceedings of the First International Conference on Consuming Linked Data, Volume 665 (COLD'10)*. 109–120.
[51] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table union search on open data. *Proc. VLDB Endow.* 11, 7 (March 2018), 813–825.
[52] Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *CoRR* abs/1511.04834 (2015).
[53] Thanh Tam Nguyen, Quoc Viet Hung Nguyen, Weidlich Matthias, and Aberer Karl. 2015. Result selection and summarization for web table search. In *Proceedings of the 31st International Conference on Data Engineering (ISDE'15)*. 231–242.
[54] Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. 168–174.
[55] Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL'15)*. 1470–1480.
[56] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *Proc. VLDB Endow.* 5, 10 (June 2012), 908–919.
[57] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI'03)*. 149–157.
[58] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Info. Retr.* 13, 4 (2010), 346–374.
[59] Dominique Ritze and Christian Bizer. 2017. Matching web tables to DBpedia - A feature utility study. In *Proceedings of the 20th International Conference on Extending Database Technology (EDBT'17)*. 210–221.
[60] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching HTML tables to DBpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics (WIMS'15)*. Article 10, 6 pages.
[61] Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. 2016. Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *Proceedings of the 25th International Conference on World Wide Web (WWW'16)*. 251–261.
[62] Sunita Sarawagi and Soumen Chakrabarti. 2014. Open-domain quantity queries on web tables: Annotation, response, and consensus models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. 711–720.
[63] Yoones A. Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. 2014. Knowledge base augmentation using tabular data. In *Proceedings of the Workshop on Linked Data on the Web Co-located with the 23rd International World Wide Web Conference (CEUR'14)*.
[64] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW'14 Companion)*. 373–374.
[65] Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web (WWW'16)*. 771–782.
[66] Zareen Saba Syed. 2010. *Wikitology: A Novel Hybrid Knowledge Base Derived from Wikipedia*. Ph.D. Dissertation. Advisor(s) Finin, Timothy W.
[67] Stephen Tyree, Kilian Q. Weinberger, Kunal Agrawal, and Jennifer Paykin. 2011. Parallel boosted regression trees for web search ranking. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*. 387–396.
[68] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering semantics of tables on the web. *Proc. VLDB Endow.* 4, 9 (June 2011), 528–538.
[69] Chi Wang, Kaushik Chakrabarti, Yeye He, Kris Ganjam, Zhimin Chen, and Philip A. Bernstein. 2015. Concept expansion using web tables. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*. 1198–1208.
[70] Hong Wang, Anqi Liu, Jing Wang, Brian D. Ziebart, Clement T. Yu, and Warren Shen. 2015. Context retrieval for web tables. In *Proceedings of the International Conference on The Theory of Information Retrieval (ICTIR'15)*. 251–260.
[71] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q. Zhu. 2012. Understanding tables on the web. In *Proceedings of the 31st International Conference on Conceptual Modeling (ER'12)*. 141–155.
[72] Yue Wang and Yeye He. 2017. Synthesizing mapping relationships using table corpus. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD'17)*. 1117–1132.

[73] Yalin Wang and Jianying Hu. 2002. Detecting tables in HTML documents. In *Proceedings of the 5th International Workshop on Document Analysis Systems V (DAS'02)*. 249–260.
[74] Yalin Wang and Jianying Hu. 2002. A machine learning-based approach for table detection on the web. In *Proceedings of the 11th International Conference on World Wide Web (WWW'02)*. 242–250.
[75] Tianxing Wu, Shengjia Yan, Zhixin Piao, Liang Xu, Ruiming Wang, and Guilin Qi. 2016. Entity linking in web tables with multiple linked knowledge bases. In *Semant. Technol.* 239–253.
[76] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. 2012. InfoGather: Entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. 97–108.
[77] Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables in natural language. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*. 2308–2314.
[78] Meihui Zhang and Kaushik Chakrabarti. 2013. InfoGather+: Semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'13)*. 145–156.
[79] Shuo Zhang. 2018. SmartTable: Equipping spreadsheets with intelligent AssistanceFunctionalities. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'18)*. 1447–1447.
[80] Shuo Zhang and Krisztian Balog. 2017. Design patterns for fusion-based object retrieval. In *Proceedings of the 39th European Conference on Advances in Information Retrieval (ECIR'17)*. 684–690.
[81] Shuo Zhang and Krisztian Balog. 2017. EntiTables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17)*. 255–264.
[82] Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the World Wide Web Conference (WWW'18)*. 1553–1562.
[83] Shuo Zhang and Krisztian Balog. 2018. On-the-fly table generation. In *Proceedings of 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'18)*. 595–604.
[84] Shuo Zhang and Krisztian Balog. 2019. Auto-completion for data cells in relational tables. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*. 761–770.
[85] Shuo Zhang and Krisztian Balog. 2019. Recommending related tables. Retrieved from http://arxiv.org/abs/1907.03595.
[86] X. Zhang, Y. Chen, X. Du, and L. Zou. 2013. Mapping entity-attribute web tables to web-scale knowledge bases. *Database Syst. Adv. Appl.* (2013), 108–122.
[87] Ziqi Zhang. 2017. Effective and efficient semantic table interpretation using TableMiner+. *Semantic Web* 8 (2017), 921–957.

Received December 2018; revised August 2019; accepted November 2019