

Merging Web Tables for Relation Extraction with Knowledge Graphs

Jhomara Luzuriaga, Emir Muñoz, Henry Rosales-Méndez, Aidan Hogan

Abstract—We propose methods for extracting triples from Wikipedia’s HTML tables using a reference knowledge graph. Our methods use a distant-supervision approach to find existing triples in the knowledge graph for pairs of entities on the same row of a table, postulating the corresponding relation for pairs of entities from other rows in the corresponding columns, thus extracting novel candidate triples. Binary classifiers are applied on these candidates to detect correct triples and thus increase the precision of the output triples. We extend this approach with a preliminary step where we first group and merge similar tables, thereafter applying extraction on the larger merged tables. More specifically, we propose an *observed schema* for individual tables, which is used to group and merge tables. We compare the precision and number of triples extracted with and without table merging, where we show that with merging, we can extract a larger number of triples at a similar precision. Ultimately, from the tables of English Wikipedia, we extract 5.9 million novel and unique triples for Wikidata at an estimated precision of 0.718.

Index Terms—Web tables, relation extraction, information extraction, knowledge graphs, distant supervision, Wikidata, Wikipedia

1 INTRODUCTION

MOST Web content is published as semi-structured HTML documents intended for human – rather than machine – consumption. Automated integration of information from multiple sources on the Web thus remains a challenging problem [1]. Addressing these limitations, various research initiatives have fostered the publication of structured content (data) on the Web, including the Semantic Web [2], Linked Data [3], etc. Thousands of graph-structured knowledge bases (aka. *knowledge graphs*) have been published using related standards [4]. Novel applications using such knowledge graphs continue to emerge; for example, the Wikidata [5] knowledge graph is used not only to structure and curate the data underlying Wikipedia and related projects, but has also become an important source for a variety of data-intensive applications, such as Apple’s Siri [6], the WikiGenomes project [7], etc.

Despite these initiatives, the majority of Web content remains unstructured or semi-structured [8, 9]. Automatically extracting structured data from unstructured content (i.e., text) with high precision and recall remains a major challenge [10]. Hence a variety of research works have rather targeted extraction from semi-structured elements of Web documents, particularly Web tables, which offer a rich source of factual content [11, 12, 13]. These works have addressed the extraction of various structured elements from Web tables, including entities, column types, and relations, with the latter being the most challenging task [10].

Building upon our previous work [14], in this paper we propose an approach for extracting triples from Web tables. We evaluate our approach for extracting triples from the tables of Wikipedia using Wikidata as a reference knowledge graph. Our method is based on distantly-supervised relation extraction that involves mapping table entities to the reference knowledge graph, detecting existing triples between pairs of entities in the same row, and then propos-

ing the corresponding relation between pairs of entities in the corresponding columns of other rows. This process produces a set of candidate triples that are then refined using binary classifiers that, based on selected features, output triples assigned to a “correct” class denoting claims that are verifiably true. Though initial results were promising [14], they revealed a key limitation: *Web tables tend to be small* [15, 16], where the tables of Wikipedia have around 10–15 rows each, on average [14], implying that limited information is available in individual tables for extraction and classification purposes. Our previous work [14] and most related approaches (e.g., [12, 17, 18], etc.) apply extraction with respect to individual tables, thus facing the same fundamental limitation of processing small tables.

Inspired by recent works [15, 16, 19], we extend our previous work [14] with a technique for grouping and merging similar tables prior to applying relation extraction. The goal is to increase the local information available, which we hypothesise will allow for the extraction of more triples at a similar precision assuming a corpus of tables with many “similar” (small) tables [15, 16]. However, as Lehmberg and Bizer [16] observe, simply merging tables with the same headers is not effective for Wikipedia’s tables, which are highly diverse in nature. We thus propose an *observed schema*, which can represent hierarchical headers, column data-types, among other information. The input tables are partitioned according to their observed schemata and refined to avoid conflicting relations. The tables in each partition are then merged (unioned) and used for extraction.

We evaluate relation extraction from Wikipedia tables with respect to the Wikidata knowledge graph, comparing the precision of triples extracted and the number of triples extracted with and without *a priori* table merging.

MOTIVATING EXAMPLE: Figure 1 shows HTML tables from two Wikipedia articles detailing information about awards. The tables indicate the year (**Year**) of the awards (**Award(s)**)

for which the given work (**Nominated work**) was awarded or nominated (**Result**) in a given category (**Category**). Most cells offer links that point to another Wikipedia article. These links can be mapped directly to entities in the Wikidata knowledge graph (e.g., `wk:Breaking_Bad` ↔ `wd:Q1079`).¹ Unlike the upper table, the first column (**Year**) of the lower table links to the specific annual event for the award.

Mapping Wikipedia links to Wikidata entities, we can find pairs of entities on the same row of the tables with existing relations. Consider the table in Figure 1a; in Wikidata we can find the triple (`wd:Q1079`, `wdt:P1411`, `wd:Q1357770`) indicating that *Breaking Bad* was *nominated for* Best Guest Starring Role on Television, or the triple (`wd:Q1357770`, `wdt:P31`, `wd:Q105447`), indicating that Best Guest Starring Role on Television is an *instance of* Saturn Award. We can then propose the Wikidata property `wdt:P1411` as holding from each entity in the **Nominated work** column to the entity in the **Category** column of the same row; we can likewise propose the property `wdt:P31` as holding between the **Category** and **Awards** columns.

With this process, we may miss triples between the main entity of the article – often not appearing in the table – and the entities of the table. For example, in Wikidata we find the triple (`wd:Q1079`, `wdt:P161`, `wd:Q726142`), indicating that *Breaking Bad* has *cast member* Giancarlo Esposito. As a pre-processing step, we thus extend each table with an additional column copying the main entity on each row, allowing us to detect triples involving the main entity.

Associating properties to pairs of table columns, we extract triples that do not already exist in Wikidata, which are proposed as *candidate triples*. Such triples may be incorrect. Consider, for example, the triple (`wd:Q132351`, `wdt:P166`, `wd:Q697007`) in Wikidata indicating that *The Usual Suspects* was *awarded* Best Cast. Within the candidate set, we will then find proposed triples such as (`wd:Q1079`, `wdt:P166`, `wd:Q1357770`) indicating that *Breaking Bad* was *awarded* Best Guest Starring Role on Television, when in reality it was only nominated. To increase precision, we associate each candidate triple with features extracted from the table and apply binary classification with the goal of partitioning correct and incorrect triples; an example of a key feature is the ratio of rows for which the proposed relation holds, where the fact that `wdt:P166` holds for few rows indicates that the relation may be incorrect.

Conversely, the table of Figure 1a has much fewer rows. The only existing triple we will find in Wikidata is (`wd:Q19798734`, `wdt:P161`, `wdt:Q27452406`), indicating that *Stranger Things* has *cast member* Sadie Sink. Hence we miss candidate triples for `wdt:P1411` (*nominated for*), `wdt:P31` (*instance of*), etc. Furthermore, features extracted for the candidate triples of this table will be based on information from only two rows, making classification more prone to noise for smaller tables. The main extension we propose in this work is to partition the table corpus into groups according to an observed schema (defined later), merging the tables in each partition before applying the aforementioned relation extraction process. Our hypothesis is that this

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

(a) `wk:Giancarlo_Esposito`

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

(b) `wk:Sadie_Sink`

Fig. 1. Two example HTML tables from Wikipedia (© ⓘ ⓘ ⓘ)

merging process will allow for more triples to be extracted with similar precision due to (1) triples being proposed across small tables that are merged together; (2) more robust statistical features being computed from larger tables. For example, if we merge the two tables of Figure 1, then the `wdt:P31` relation found on rows of the table of Figure 1a across the columns **Category** and **Awards** can be used to propose triples from the rows of Figure 1b between the same columns. Also the features extracted from Figure 1b for the triples extracted would be based on more rows.

CONTRIBUTIONS AND OUTLINE: We first discuss related works (§2) and the setting for our approach (§3). Subsequently, our main contributions are as follows: we propose an approach using distant supervision for extracting triples from Web tables at large scale based on a reference knowledge graph (§4); we propose a notion of observed schema, by which we can merge the tables in a large corpus prior to applying relation extraction (§5); we evaluate these proposals for extracting triples from Wikipedia's tables using Wikidata as a reference knowledge graph (§6). We finally summarise the main conclusions of the work, discussing current limitations and future directions (§7).

With respect to our previous work [14], the most important extension is the method for merging tables based on observed schemata. Other extensions include: the definition of novel property-based features for classification capturing information about multiplicities and domain/range type compatibility; evaluation with additional classification models and methods; as well as migrating from DBpedia to Wikidata as a reference knowledge graph.

2 RELATED WORK

As mentioned in the introduction, a number of previous works have addressed a variety of Information Extraction tasks over Web tables. We first discuss works that propose extraction methods for individual tables before discussing proposals that merge multiple tables.

2.1 Information Extraction from Individual Tables

We discuss works that extract structure from tables, divided into various sub-tasks that can be considered [10].

1. In this paper, we use the following prefixes:

`wk:` <http://en.wikipedia.org/wiki/>
`wd:` <http://www.wikidata.org/entity/>
`wdt:` <http://www.wikidata.org/prop/direct/>

Fresh		Event											
		2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	
Directed by	Boaz Yakin												
Produced by	Lawrence Bender Randy Ostrow												
Written by	Boaz Yakin												
Starring	Sean Nelson Giancarlo Esposito Samuel L. Jackson N'Bushe Wright												
Music by	Stewart Copeland												
Distributed by	Miramax Films												
Release date	August 24, 1994												
Running time	114 min.												
🏅 Olympic Games	Time trial	Not held				34	Not held			Not held			
	Road race	Not held				34	Not held			Not held			
	Cross-country	Not held				—	Not held			35	Not held		
🌐 World Championships	Team Time trial	Not held				4	7	9	27	—			
	Time trial	—	—	—	—	—	—	—	—	—	—	—	
🇪🇺 European Championships	Road race	—	DNF	12	14	6	43	1	1	1	DNF	—	
	Time trial	Not held							—	—	—	—	—
🇺🇸 National Championships	Road race	—	—	—	—	—	—	1	—	—	—	—	
	Time trial	10	—	1	1	1	1	1	2	2	1	4	

(a) wk:Fresh_(1994_film)

(b) wk:List_of_career_achievements_by_Peter_Sagan

Fig. 2. An example of (a) an attribute–value (aka. infobox) table and (b) a matrix table from Wikipedia (©️🌐)

TABLE EXTRACTION AND NORMALISATION: A straightforward approach to extract tables from webpages is to simply use the corresponding HTML tags, such as `<table>`, `<th>` (header), `<tr>` (row), `<td>` (data/cell), etc., to identify the table and extract its structure. However, HTML tables are often defined with visual aesthetic, rather than machine readability, in mind, meaning that tables may contain spanned rows or columns (see, e.g., *Breaking Bad* in Figure 1), may contain blank columns for spacing, may contain nested or split tables, may use bold font instead of explicit header tags, may contain variable-length rows, etc. The goal of table normalisation is then to extract a matrix from the table [20], optionally denoting headers. This process may involve duplicating spanned content into individual cells, unnesting tables, detecting non-standard header representations, etc.

TABLE CLASSIFICATION: Various works address the classification of web tables, which allows for filtering irrelevant tables, or adapting the extraction process for different types of tables. A coarse categorisation separates *genuine* tables (with factual content) from *non-genuine* tables (used for forms, navigation, etc.) [12, 21]. Crestan and Pantel [13] propose a more detailed classification scheme based on twelve table types, including *listing*, *attribute-value*, *matrix*, *enumeration*, etc. Diverse tables are likewise found in Wikipedia, where Figure 2a gives an example of an attribute–value (aka. infobox) table, while Figure 2b gives an example of a complex matrix with both horizontal and vertical headings. This diversity of tables complicates automated interpretation. Most works tend to focus on one type of table or another, with info-box tables (e.g., [22, 23]) and relational tables (e.g., [17, 18, 24, 25, 26]) – tables with a single vertical header as the first row(s) of the table along the lines of Figure 1 – being the most common targets.

ENTITY LINKING: In the table of Figure 1a, while some cells are linked to Wikipedia articles, others (such as *Primetime Emmy Awards*) are not. Given a target knowledge base (e.g., Wikipedia, Wikidata, etc.), the goal of Entity Linking is to propose links for the cells of Web tables where not present. While Entity Linking has mostly been explored for text [27], specialised techniques have been proposed for tables, exploiting the additional structure available in the latter case. One such approach is *TaBEL* [28], which uses a co-citation measure for disambiguating (related) entities in the same table. Other approaches adopt methods based on probabilistic graphical models [29], semantic embeddings [25, 30],

ontology matching [30], neural networks [31], etc.

COLUMN TYPING: Annotating the columns of tables with types can be used for retrieving tables or in further extraction steps. Types may vary from generic datatypes – entity, numeric, date, etc. – to classes extracted from a knowledge-base – person, movie, etc. Wang et al. [32] propose to use *Hearst* patterns to extract a taxonomy of types from text, used to classify entities and thereafter columns. Other approaches rely on the classes of entities detected in the columns taken from an existing knowledge base [17, 18, 24, 25, 26], possibly combined with string matching between column and class labels [26].

ATTRIBUTE EXTRACTION: Attribute extraction identifies a *subject column* (or columns) that identify the primary entities described by the table. The other columns are then considered to be *attributes* of the subject entities and can be matched – for example – to properties in a knowledge base that denote the given relation. A common heuristic to detect the subject column is to select the leftmost column without repeated entities [12, 18, 33]; other approaches find the textual column in any position with the most unique entities [34], or use a classifier based on Support Vector Machines trained on labelled data [18]. Though attribute extraction is quite a popular approach, in many web tables there is no clear subject column(s); for example, it is not clear what the subject column(s) of the tables in Figure 1 would be. Even with a clear subject, a table may denote relations of interest not involving that subject; for example, in Figure 1, there is a relevant *part-of* relation between the **Category** and the **Awards** columns though neither appear to be subjects.

RELATION EXTRACTION: A more general approach is to extract (typically binary) relations between any pair of columns (rather than identifying a subject column). The result is typically a set of pairs of columns associated with a property that denotes the relation. Common approaches involve distant supervision, where the existing triples in a knowledge base for pairs of entities in the same row of a table are used to generate candidates; and/or string matching, which compares column names and property labels in the knowledge base [17, 18, 24, 35].

2.2 Merging Tables

A number of works have recently proposed techniques to merge web tables for a variety of applications including

table enrichment, data consolidation, visualisation, and indeed, information extraction. We now discuss works representing three methods for merging tables, namely *table joins*, *table unions*, and *hybrid* approaches that combine both.

JOINED TABLES: Joining tables refers to a horizontal form of merging, which, in general, involves computing relational joins over source tables, thus creating a joined table that combines the source tables' columns. A main challenge is to identify columns in different tables that can be joined, and will produce non-empty tables. An early work was Google Fusion Tables [36], which gathers, joins and visualises tables for data exploration and management. Yakout et al. [37] proposed to use table joins to enrich tables generated by users, proposing additional sets of columns that may be of interest. Given a Wikipedia table, Bhagavatula et al. [38] address the problem of identifying columns in other tables upon which the current table can be joined. The system proposed by Lehmborg et al. [34] rather applies left-outer joins, further consolidating (non-join) columns with compatible data.

UNION TABLES: Other works merge tables by applying a relational *union*, thus combining the source tables' rows. Such approaches are motivated by observations of tables with similar headers appearing on a given website due to copying-and-pasting habits of editors, the use of templates, etc. (see, e.g., Figure 1). Yoshida et al. [39] apply table clustering based on columns that are particular to a given cluster (e.g., a **BirthDay** might indicate a cluster about people), and then produce a union table for each cluster by also matching and combining columns. Ling et al. [15] propose to merge Web tables by taking the union of all tables with the same header, further adding columns that identify the contextual entities of individual tables. Nargesian et al. [40] find tables in open data with headers that can be unioned based on having matching columns with high overlap of similar values from similar classes. Wang and He [41] aim to synthesise mapping relations, which are binary relations with a functional dependency; they also apply unions on binary relations with many overlapping pairs and avoid conflicts by not unioning relations that break the functional dependency (i.e., giving two states for a single city). Cannaviccio et al. [19] compute union tables from Wikipedia but extract and merge binary relations; e.g., rather than unioning the tables in Figure 1, they may extract and union the column pair [**Award(s)**, **Category**].

HYBRID MERGES: One could also consider combining joins and unions. Lehmborg and Bizer [16] first compute union tables from a large corpus of web tables. Thereafter, they apply outer joins on the union tables and apply schema-matching techniques to consolidate compatible columns identified based on column names and the overlap in values appearing in the columns. They further propose to use the same entities in different tables with the same values in different columns to identify compatible columns. The authors show that these techniques improve results for table typing, column typing and attribute extraction.

2.3 Novelty

Our goal is to apply relation extraction over the tables of Wikipedia in order to extract novel triples for a knowledge

graph (namely Wikidata). We extend upon our previous work [14], which provides a novel technique for extracting triples from any pair of table columns, with finer-grained triple-wise feature extraction and classification (as opposed to labelling pairs of columns in the table with a relation). The method further considers triples involving the protagonist of the table: the entity of the article containing the table. We extend upon our previous work by exploring new features and new methods for unioning tables prior to extraction.

The approach of computing union tables has been studied previously [15, 16, 19, 40, 41]. Ling et al. [15] does not consider relation extraction, rather focusing on extracting contextual columns from the surrounding text. Nargesian et al. [40] and Wang and He [41] rely on overlapping rows to merge tables, which is too restrictive for Wikipedia tables, which tend to be small and focussed on a particular entity (see Figure 1 for two typical tables that we wish to merge with no overlap), where we thus rather rely on matching headers. Wang and He [41] further assume functional dependencies for conflict resolution that do not apply in the general case, though we will use a similar feature (amongst many others) for classifying correct triples. Wang and He [41] and Cannaviccio et al. [19] only apply unions on pairs of columns, not full tables; we merge full tables as all columns are useful to decide when to merge or not, where, for example, merging pairs of columns like [**Year**, **Category**] from Figure 1 extracted from all tables of Wikipedia would merge information from unrelated tables. Lehmborg and Bizer [16] apply both the union table approach of Ling et al. [15] and further methods for stitching tables, but focus on predefined schema matching tasks, such as attribute extraction; while their approach is well-suited to tables generated from databases, the authors explicitly mention that it does not work well for Wikipedia's diverse tables (which are filtered from their corpus).

Ultimately our method extracts 5.9 million novel triples using the Wikidata vocabulary (its native entity and property identifiers) from Wikipedia tables at a precision of 0.718. The aforementioned works do not present similar results.

3 SETTING

We assume as input a *knowledge graph*, defined to be a directed-edge labelled graph $G := (V, E, P)$, where V is a set of *vertices*, P a set of *edge labels* (aka. *properties*), and E a set of *edges* (aka. *triples*) such that $E \subseteq V \times P \times V$. A triple $(s, p, o) \in E$ denotes a labelled edge $s \xrightarrow{p} o$ in the graph, where we call s a subject, p a predicate, and o an object. The vertices s and o denote entities while a triple denotes a binary relation $p(s, o)$ with the predicate p between them.

We further assume as input a set of tables. We define a table T to be an $m \times n$ matrix, where $T(i, j)$ denotes the value in the i^{th} column of the j^{th} row of the table ($1 \leq i \leq m$, $1 \leq j \leq n$). We assume that the first h rows ($1 \leq h < n$) of the table T form a *header*, where the rest of the table forms the *body*. For convenience, we denote the header of the table T by H_T : the $m \times h$ matrix formed from the first h rows of T (such that $H_T(i, j) = T(i, j)$ for $1 \leq i \leq m$, $1 \leq j \leq h$). We denote the body of the table T by B_T : the $m \times (n - h)$ matrix formed from the other rows of T (such

that $B_T(i, j) = T(i, j + h)$ for $1 \leq i \leq m, 1 \leq j \leq (n - h)$. We assume the body of the table to be non-empty ($h < n$).

Web tables can be parsed into this normalised representation using HTML tags. We assume that standard techniques for table normalisation have been applied [14, 20] in order to (1) duplicate the value of spanning cells into each cell that they span; (2) un-nest tables; (3) consider vertical headers beyond the horizontal header as data cells;² (4) add a blank header row if no header is provided. The result is a table of the form T . If parsing and normalisation fails (e.g., due to syntax errors), the table is simply dropped.

We consider all tables of the class `wikitable` as input to our approach; we thus exclude info-boxes (see Figure 2a) for which dedicated approaches exist [22, 42]. Though our approach works best for relational tables with (only) horizontal headers, triples can be extracted from any table with binary relations present in the rows; for example, from the first two columns of the matrix of Figure 2b, we may extract triples such as (`wd:Q506424`, `wdt:P527`, `wd:Q13603535`), indicating that UCI Road World Championships *has part* UCI Road World Championships — Men’s road race.

Sometimes relations will extend from elements inside tables to elements outside the table in the same web document. Tables may thus be extended with columns to represent such elements [15]. In our case, we add a column representing the *protagonist* of the table: the principal entity that the web-page containing the table describes. Specifically, we add a column to each table T with a fixed and unique header value (***Protagonist***, leaving subsequent header rows blank), repeating the name of the protagonist in each row of the column’s body.³ Given that the result is again a table as before, we will henceforth implicitly assume that the protagonist column has been added to the tables.

Finally, we consider that the content of the cells in the tables can be linked to vertices in the knowledge graph. More specifically, let \mathbf{T} denote the set of all tables. Given a table T and a knowledge graph $G = (V, E, P)$, we define a *table link-set* as a mapping $\lambda: \mathbf{T} \rightarrow \mathbf{T}$ such that $H_T = H_{\lambda(T)}$ and for all $1 \leq i \leq m, 1 \leq j \leq (n - h)$, $B_{\lambda(T)}(i, j) := \bigvee_{i,j} V_{i,j}$ where $\bigvee_{i,j}$ is an ordered (possibly empty) list of vertices from V that are mentioned in the content of the cell $B_T(i, j)$. This definition allows for multiple vertices of the knowledge graph to be linked from a single cell; see the **Category** value of the last row in Table 1a for a real-world example. We call $\lambda(T)$ the *linked table* of T . Given that we will work with Wikipedia tables and the Wikidata knowledge graph, which are interlinked, we can simply compute $\lambda(T)$ based on the links to Wikipedia articles embedded in cells (otherwise Entity Linking could be applied [25, 28, 29, 30, 31]).

Example 1. In Figure 3, we show the normalised and linked versions of the table from Figure 1b. In the normalised

version, we highlight the duplication of the **Stranger Things** value into the cells that the original cell spanned, and also the addition of the protagonist column. In the linked version, we highlight that cells may have zero or more links to vertices of the knowledge graph, and that links are ordered in terms of appearance.

4 RELATION EXTRACTION

Given a knowledge graph $G = (V, E, P)$, an $m \times n$ input table T with k header rows, and its linked version $\lambda(T)$, we now describe the process of extracting triples from T with respect to G . The result of this process is a novel set of triples E' in the signature of G ; in other words, we output $E' \subseteq V \times P \times V$ such that $E' \cap E = \emptyset$. We begin by extracting an initial set of candidate triples using a distant supervision approach. We then associate each candidate triple with a set of features and train a binary classifier over those features to predict if a given candidate triple is correct or not. We now discuss each of these steps in turn.

4.1 Extracting Candidate Triples

We identify candidate triples by finding existing triples from the knowledge graph holding between pairs of entities in different columns of the same row, and propose novel triples with the same property for other pairs of vertices in different rows. More precisely, for each triple $(s, p, o) \in E$ such that there exist two column indexes i and i' ($1 \leq i \leq m, 1 \leq i' \leq m, i \neq i'$) and a row index j ($1 \leq j \leq (n - h)$) such that $s \in B_{\lambda(T)}(i, j)$ and $o \in B_{\lambda(T)}(i', j)$, we propose the triples $\{(s', p, o') \mid s' \in B_{\lambda(T)}(i, j'), o' \in B_{\lambda(T)}(i', j')\} \setminus E$ ($1 \leq j' \leq (n - h), j \neq j'$) as candidate triples.

Example 2. With reference to the normalised and linked tables T and $\lambda(T)$ shown in Figure 3, given that the triple (`wd:Q2530270`, `wdt:P31`, `wd:Q268200`) exists in the Wikidata knowledge graph – stating that Outstanding Performance by an Ensemble in a Drama Series is an *instance of* Screen Actor Guild Award – and given that both entities appear in different columns of the first row of the table body – we will propose the corresponding triples for the second row as candidate triples, namely:

- (`wd:Q4220900`, `wdt:P31`, `wd:Q110145`),
- (`wd:Q26704332`, `wdt:P31`, `wd:Q110145`),
- (`wd:Q26308335`, `wdt:P31`, `wd:Q110145`), etc.

denoting that Best On-Screen Team, G. Matarazoo and F. Wolfhard are *instance of* MTV Movie & TV Award, respectively. Of these, only the first triple is correct.

4.2 Feature Extraction

As seen in the previous example, many of the candidate triples may be incorrect. Hence we extract features for individual triples that will serve as signals for classifying these triples as correct or incorrect, with the goal of boosting the precision of the extracted triples. The features we consider have different levels of granularity, as follows:

2. For example, in Figure 2b, we consider H_T to be the first row of the table, while other vertical header cells such as Olympic Games and Time trial will form part of B_T .

3. Wikipedia contains list articles that do not directly refer to a “real-world” protagonist. These cases could be detected based on filtering article URLs containing `List_of`; alternatively, Wikidata provides entities corresponding to such articles, typed accordingly, that could be used to avoid false positives (e.g., books whose title start with “List of . . .”). However, these Wikidata entities for list articles have relations in the knowledge graph, and such cases are not common overall. Hence we choose to include such articles/entities as protagonists.

Year	Award	Category	Nominated work	Result	*Protagonist*
2018	Screen Actors Guild Awards	Outstanding Performance by an Ensemble in a Drama Series	Stranger Things	Nominated	Sadie Sink
2018	MTV Movie & TV Awards	Best On-Screen Team (with G. Matarazoo, F. Wolfhard ...)	Stranger Things	Nominated	Sadie Sink

(a) Normalised table with hyperlinks underlined

Year	Award	Category	Nominated work	Result	*Protagonist*
[wd:Q37998098]	[wd:Q268200]	[wd:Q2530270]	[wd:Q19798734]		[wd:Q27452406]
[wd:Q43379479]	[wd:Q110145]	[wd:Q4220900, wd:Q26704332, Q26308335, ...]	[wd:Q19798734]		[wd:Q27452406]

(b) Linked table

Fig. 3. Normalised and linked versions of the table shown in Figure 1b

- TABLE-LEVEL: (1) ordinal of table in the article,⁴ (2) number of rows, (3) number of columns, (4) ratio $\frac{(2)}{(3)}$, (5) number of candidate triples extracted;
- COLUMN-LEVEL: (6|7) ordinal of subject|object column, (8|9) number of entities in subject|object column, (10) ratio $\frac{(8)}{(9)}$, (11|12) number of unique entities in subject|object column, (13) ratio $\frac{(11)}{(12)}$, (14) number of potential triples for a single property, (15) number of unique potential triples for a single property, (16) is the subject or object column the protagonist;
- PROPERTY-LEVEL: (17) number of triples in the knowledge graph with the property, (18|19) number of unique subjects|objects in the knowledge graph for the property, (20⁺) ratio $\frac{(19)}{(17)}$, (21⁺|22⁺) maximum number of subjects|objects with the same object|subject for the given property, (23⁺|24⁺) compatibility with domain|range;
- CELL-LEVEL: (25|26) number of entities in the subject|object cell, (27) ratio $\frac{(25)}{(26)}$, (28|29) total string length of subject|object cell, (30|31) presence of formatting tags in subject|object cell, (32⁺|33⁺) number of links in the subject|object cell (includes links not resolved to entities); (34⁺|35⁺) string length of subject|object cell without links, (36⁺|37⁺) is the subject|object cell part of a colspan or rowspan;
- PROPERTY/COLUMN-LEVEL: (38|39) string similarity for property and subject|object header, (40) maximum of (38) and (39), (41) number of rows for which the corresponding relation holds in knowledge graph, (42) ratio $\frac{(41)}{(2)}$, (43) number of triples for which the corresponding relation holds in the knowledge graph, (44) ratio $\frac{(43)}{(14)}$, (45) number of unique triples for which the corresponding relation holds in the knowledge graph, (46) ratio $\frac{(45)}{(15)}$;
- TRIPLE-LEVEL: (47⁺|48⁺) for the given triple (s, p, o) , the number of values for x such that $(s, p, x)|(x, p, o)$ already appears in the knowledge graph.

Cell-level and triple-level features allow for labelling individual triples from each row as correct or incorrect. For example, a cell may have additional text that invalidates or questions the validity of the particular value of a triple (e.g., “disputed”, “later rescinded”, “citation needed”) on a particular row; or a cell may contain multiple links, where a relation may sometimes only hold for one such link

4. By *ordinal*, we refer to the position of an element in a list; e.g., if it is the second table in an article, the ordinal is 2.

(see Figure 1b). Having fine-grained features allows us to identify correct triples on a row-by-row or triple-by-triple basis, unlike related approaches that rather aim at labelling pairs of columns with a given property, which are then interpreted uniformly for all rows [12, 17, 18, 24, 33, 34, 35].

Features marked “.+” are new to this work (we refer to our previous work for more discussion on the original features [14]). These new features were identified based on a study of the false positives generated using the original features only. This study identified two common issues.

The first issue related to triples classified as correct even though they are incompatible with the domain/range of the property. Referring to Figure 3, a triple $(wd:Q19798734, wdt:P1441, wd:26704332)$ – *Stranger Things nominated for Gaten Matarazoo* – has an object $(wd:26704332/Gaten\ Matarazoo)$ whose class $(wd:Q5/Human)$ is incompatible with the range of the property $(wdt:P1441/nominated\ for)$. We add features (23⁺) and (24⁺) to identify such cases: given a triple (s, p, o) , we say that it is incompatible with the domain of p if and only if there does not exist a triple (x, p, y) in the knowledge graph such that x shares at least one class with s ; conversely, we define it to be incompatible with the range of p if and only if there does not exist a triple (v, p, w) in the knowledge graph such that w shares at least one class with o .⁵

The second issue related to triples classified as correct even though the property has a low multiplicity and a value already exists in the knowledge graph; for example, we may find a triple $(wd:Q16, wdt:P36, wd:Q172)$ – *Canada has the capital Toronto* – which would seem unlikely given that *capital* tends to be a (nearly) functional property (i.e., that countries tend to have one capital), and *Canada* already has a known capital in the knowledge graph. Features (21⁺) and (22⁺) are used to capture statistics of a property’s multiplicity in both directions, while (47⁺) and (48⁺) are used to indicate if the property already has a corresponding value in the knowledge graph, again in both directions.

4.3 Binary Classification

We apply binary classifiers to predict whether each candidate triple is correct or incorrect. We consider six classification models (two of which are new to this work, marked “.+”): Bagging Decision Trees (BDT), Extreme Gradient Boost (XGB)⁺, K-Nearest Neighbours (KNN)⁺, Logistic Regression (LR), Naive Bayes (NB) and Random Forests (RF).

5. One could alternatively consider using explicit domain/range axioms or constraints if available for the knowledge graph.

Area	Men				Women			
	Time (s)	Wind (m/s)	Athlete	Nation	Time (s)	Wind (m/s)	Athlete	Nation
Africa (records)	9.85	+1.7	Olusoji Fasuba	 Nigeria	10.78	+1.6	Murielle Ahouré	 Ivory Coast

Fig. 4. Example of a table with a hierarchical header (with first data row only) from Wikipedia (© – wk:100_metres)

5 MERGING TABLES

We now discuss how tables are merged prior to extraction. The desiderata we identify for such a merge are as follows. (R1) BINARY RELATION COLLATION: our use-case for merging is to improve the extraction of triples, and in particular, to create larger tables that collate more rows for binary relations. (R2) CAUTIOUS MERGING: merging unrelated tables (false positives) will generate noise in the extracted triples compared with the baseline of applying relation extraction to individual tables; not merging related tables will tend towards the baseline output. Hence we prefer cautious merging that prioritises avoiding false positives over false negatives. (R3) SCALABILITY. Given that we work with millions of tables, we require a merging method with $O(t)$ or $O(t \log t)$ runtime for t the number of input tables.

Given (R1), we apply relational unions (rather than joins) over compatible tables. Given (R2), we propose an observed schema that normalises and enriches the information available in the table header and split merged tables having incompatible properties. With respect to (R3), we can group the tables to be merged according to having the same observed schema (which can be hashed or sorted). We now describe this process in more detail.

5.1 Merging Tables under Observed Schemata

A natural method to merge the related input tables \mathcal{T} would be to first partition \mathcal{T} according to the table headers: $\mathcal{T}/\sim_H := \{\{T' \mid H_T = H_{T'}\} \mid T \in \mathcal{T}\}$, and then apply a relational union over the tables in each part $\mathcal{T}' \in \mathcal{T}/\sim_H$ to merge them into a single large table under the respective header. However, this would not merge tables with minor variations in header names (see **Award** vs. **Awards** in Figure 1), nor would it merge tables with compatible columns in a different order. Hence Ling et al. [15] propose to relax column order and also to match column names with the same meaning when performing a union of tables.

However, we found numerous cases in Wikipedia for which the method proposed by Ling et al. [15] is not well-defined. First, some tables have multiple columns with the same name, often resulting from cell spans; for example, in Figure 2b, normalising the **Event** column span gives two columns of the same name. Another common case was the use of hierarchical headers, where it is not immediately clear how they should be handled; Figure 4 provides an example. We also found false matches on generic column names carrying incompatible types of data; for example, a column named **From** may contain dates, strings (e.g. place names), numbers, etc., depending on the table.

To address such issues, we propose the notion of an *observed schema*, which is a set of *column signatures* computed for each column of an input table. In the following, we

assume a normalisation function $\nu(\cdot)$ for cell values, which we instantiate by applying lower-case and word stemming. By $\#S$, we denote the cardinality of the set S .

Definition 1. Given an $m \times n$ input table T with h header rows, the *column signature* for the i^{th} column of T ($1 \leq i \leq m$) is a tuple $\sigma_i = (l_i, d_i, o_i)$, containing a *label* $l_i := (\nu(H_T(i, 1)), \dots, \nu(H_T(i, h)))$ encoding the normalised content of the h cells of the i^{th} column of the header; a *datatype* $d_i \in \{\text{NUM}, \text{DATE}, \text{STR}, \text{EMPTY}\}$ denoting the observed type of values in the i^{th} column of the body; and an *ordinal* $o_i := \#\{i' \mid 1 \leq i' \leq i, (l_i, d_i) = (l_{i'}, d_{i'})\}$ indicating the order of the i^{th} column with respect to other columns with identical keys and datatypes.

Definition 2. Given an $m \times n$ input table T , its *observed schema* is defined as $\Sigma(T) := \{\sigma_1, \dots, \sigma_m\}$, where σ_i denotes the column signature of the i^{th} column ($1 \leq i \leq m$) of T .

Example 3. In Table 1, we provide examples of the observed schemata for four of the tables shown previously. Please note that the sets are unordered. In practice, a ***protagonist*** column would also be present, with the same column signature in each input table.

We then partition the input tables \mathcal{T} per their observed schemata: $\mathcal{T}/\sim_\Sigma := \{\{T' \mid \Sigma(T) = \Sigma(T')\} \mid T \in \mathcal{T}\}$. To compute this partition in practice, a canonical form of each observed schema is computed by ordering the column signatures it contains lexicographically; this serves as a partitioning key, where partitioning can then be trivially conducted in $O(t)$ using (ideal) hashing on the keys, or $O(t \log t)$ using sorting on the keys (where $t = \#\mathcal{T}$). The tables in each part can then be unioned straightforwardly.

5.2 Refining Merged Tables

Initial results revealed that merging tables under observed schemata sometimes led to unrelated tables (often with generic column names) being merged; for example, considering tables with column labels **{name, country}**, if the **name** column lists movies or other works of art, wdt:P495 (*country of origin*) is the most appropriate property; if it lists cities or rivers, wdt:P17 (*country*) is the most appropriate choice; if it lists people, then wdt:P27 (*country of citizenship*) or wdt:P1532 (*country for sport*) might be appropriate choices. Merging all such tables into one may lead to many erroneous triples such as (wd:Q676203, wdt:P1532, wd:Q419) – Machu Picchu has *country for sport* Peru – being proposed.

After partitioning but prior to merging, we use triples from the knowledge graph appearing in the tables to detect conflicts and resolve such cases. In particular, given a table T with n rows in its body B_T , let $\Lambda_{i,j}(T) := \bigcup_{1 \leq k \leq n} (B_{\lambda(T)}(i, k) \times B_{\lambda(T)}(j, k))$ denote the pairs from columns i and j on each row of $B_{\lambda(T)}$. Given a knowledge graph $G = (V, E, P)$, let $\text{so}(G, p) := \{(s, o) \mid (s, p, o) \in E\}$ denote the pairs of vertices related by an edge labelled with the property p in G . For each pair of column indexes (i, j) of each table $T \in \mathcal{T}'$ ($i \neq j$), we compute the set of properties $P_{i,j}^{\max}(T) := \arg \max_{p \in P} \#(\text{so}(G, p) \cap \Lambda_{i,j}(T))$; in the special case that $\#(\text{so}(G, p) \cap \Lambda_{i,j}(T)) = 0$ for all $p \in P$, we define $P_{i,j}^{\max}(T) := \{\}$. Intuitively $P_{i,j}^{\max}(T)$ gives us the set of properties tied for having the most triples in the knowledge graph whose subjects and objects can be found

TABLE 1
Observed schemata for example input tables

INPUT TABLE(S)	OBSERVED SCHEMA
Tables 1a and 1b	{{((year), DATE, 1), ((award), STR, 1), ((category), STR, 1), ((nominat work), STR, 1), ((result), STR, 1)}
Table 2b	{{(event), STR, 1), ((event), STR, 2), ((2009), STR, 1), . . . , ((2019), STR, 1)}
Table 4	{{(area), STR, 1), ((men, time (s)), NUM, 1), . . . , ((women, time (s)), NUM, 1), . . . , ((women, nation), STR, 1), }

on the same row of T for the indicated pair of columns (being empty if no triples are found for those columns). We further define *incompatible properties* as follows, where α denotes a compatibility threshold for properties:⁶

Definition 3. Given a knowledge graph $G = (V, E, P)$, we call $p_1 \in P$ and $p_2 \in P$ *incompatible properties*, denoted $p_1 \not\asymp p_2$ if and only if $\frac{\#(\text{so}(G, p_1) \cap \text{so}(G, p_2))}{\min(\#(\text{so}(G, p_1)), \#(\text{so}(G, p_2)))} < \alpha$; otherwise we call p_1 and p_2 *compatible*, denoted $p_1 \asymp p_2$.

Example 4. Given two properties `wdt:P27` (*country of citizenship*) and `wdt:P1532` (*country for sport*), let us assume that `wdt:P27` relates 10,000 vertex pairs in the knowledge graph (including, e.g., (wd:Q633, wd:Q16) aka. (Neil Young, Canada)) while `wdt:P1532` relates 1,000 vertex pairs (including, e.g., (wd:Q615, wd:Q414) aka. (Lionel Messi, Argentina)). Let us assume that we set $\alpha = 0.5$. Since `wdt:P1532` relates fewer pairs, for `wdt:P27` and `wdt:P1532` to be considered compatible, `wdt:P27` must relate at least half (i.e., 500) of the pairs in the knowledge graph that `wdt:P1532` relates.

Analogously, we call two sets of properties $P_1 \not\asymp P_2$ incompatible if and only if there exists $p_1 \in P_1$ and $p_2 \in P_2$ such that $p_1 \not\asymp p_2$ (observe that the empty set is thus compatible with any set of properties).

Finally, for each part $\mathcal{T}' \in \mathcal{T}/\sim_\Sigma$, we separate tables $T_1 \in \mathcal{T}'$ and $T_2 \in \mathcal{T}'$ where there exists a pair of column indexes (i, j) such that $P_{i,j}^{\max}(T_1) \not\asymp P_{i,j}^{\max}(T_2)$.⁷ Let $P^{\max}(T) := \{(P_{i,j}^{\max}(T), i, j) \mid 1 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$ capture the most frequent properties for all pairs of columns in the table T . We first subdivide \mathcal{T}' into the partition $\mathcal{T}'/\sim_P := \{\{T' \in \mathcal{T}' \mid P^{\max}(T) = P^{\max}(T')\} \mid T' \in \mathcal{T}'\}$. We then iterate over the parts of \mathcal{T}'/\sim_P in ascending order of their cardinality, merging each successive part with the largest compatible part found, maintaining the part if no larger compatible part is found. This is repeated for all parts of \mathcal{T}/\sim_Σ , producing the final partition of tables.

The refinement of each part \mathcal{T}' requires $O(t^2)$ compatibility checks, for $t = \#\mathcal{T}'$, potentially affecting scalability. However, in practice, relatively few unique values for $P^{\max}(\cdot)$ are identified within each part.

5.3 Extracting Candidate Triples from Merged Tables

With a set of refined/merged tables in hand, linked tables can be merged, or alternatively merged tables can be linked, in an analogous manner to that previously discussed. The same process as described for individual tables in Section 4 can subsequently be applied for extracting candidate triples

from merged tables. Rather than replace the features associated with individual tables in Section 4.2, we extend them with analogous features computed for the merged table in those cases where the value of the feature may differ; for example, with respect to feature 3, we will maintain the count of rows in the original table and the merged table as two separate features; on the other hand, for feature 2, the number of columns does not change and hence we keep one feature. We thus include merged features for (2), (8), (9), (11), (12), (14), (15), (41), (42), (43), (44), (45), and (46). We distinguish the merged versions of features with \cdot^m ; e.g., (9^m) . Though these features are defined analogously as for individual tables, the merged variants can take very different values for a given candidate triple, as they are based on merged tables that in some cases have thousands of times more rows than an individual table. We also add two more features: (49^m) is the triple proposed from the individual table or (only) from the merged table; (50^m) the number of individual tables forming the merged table. We expect both of these features to negatively correlate with the positive class: merged tables (particularly large ones) should exhibit more noise since a single relation observed in one row of a merged table (with potentially thousands of rows) will generate candidate triples for all rows. A positive or high value (respectively) for these features should demand stronger evidence for the positive class from elsewhere.

6 EVALUATION

We consider four key research questions. (Q1) How many triples can be extracted from individual Wikipedia tables using the proposed methods and at what level of precision? (Q2) Do the novel features proposed in this work increase the number of triples that can be extracted from individual tables at a given level of precision? (Q3) Do the additional features extracted from merged tables increase the number of triples that can be extracted at a given level of precision? (Q4) How many more novel, correct triples can be extracted by merging tables overall, and at what level of precision?

MATERIALS AND FURTHER DETAILS: The Wikipedia corpus is available online [43]. Code and other materials can be found at <https://wikitables.github.io/>. Further details for experiments and their results are also available in [44].

6.1 Setting

We discuss the table corpus, knowledge graph, classifiers and relation extraction methods considered.

TABLE CORPUS: Our tables are sourced from Wikipedia. Specifically, we downloaded 5,582,225 articles in HTML format from English Wikipedia using the REST API.⁸ From

6. Where available, property disjointness axioms could also be leveraged for determining incompatibility.

7. For brevity, we assume that column indexes have been aligned for analogous columns in different tables of each part of \mathcal{T}/\sim_Σ .

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

these articles we extract 17,553,469 raw tables. Of these tables, 52.5% are layout tables (not containing data), 21.7% are info-boxes (not targeted by this work, being targeted by dedicated methods [22, 23]), 3.2% are smaller than 2×2 (and thus cannot have triples extracted), while 0.002% are tables-of-content (not containing data). Filtering all of these categories of tables, we input the remaining 3,957,549 (22.5%) tables into the normalisation process, which resulted in 3,967,412 tables (the number increases slightly due to unnesting inner tables). We then discard tables without headers, leaving 3,631,229 (90.31%) of the normalised tables.

KNOWLEDGE GRAPH: We use the Wikidata knowledge graph [5] for our experiments. Specifically we use the RDF truthy dump from 2018-07-30. Since our method currently extracts triples between entities, we filter triples with literals to make the knowledge graph more concise, leaving 470,573,942 edges, 51,488,027 vertices and 4,936 properties. Entities in table cells are linked to Wikidata using the native Wikidata \leftrightarrow Wikipedia mapping provided by Wikimedia. Of the 3,631,229 tables in the corpus, at least one link was found to Wikidata for 3,050,328 (85%) tables (not including the protagonist, which is always linked). We found on average ~ 5 Wikidata links per column, and an average of ~ 5 columns and ~ 13 rows per table with such entities. Links to 3,175,548 unique vertices/entities of Wikidata were found.

LABELLING DATA: Developing a training set for triples requires labelling data manually. Where this is necessary, we define four labels: CORRECT, INCORRECT, CONTEXTUAL, or UNKNOWN. The CONTEXTUAL label is used, e.g., for triples that were true in the past, but not now, such as (wd:30,wdt:P35,wd:Q76): United States has *head of state* Barack Obama. The UNKNOWN label is used in cases where the judge could not find information to verify or reject a triple, or in subjective cases, such as (wd:Q38222,wdt:P800,wd:Q165713): George Lucas has *notable work* Star Wars: Episode I – The Phantom Menace.

6.2 Results: Extracting Candidate Relations

We now discuss results up to and including the extraction of candidate triples from individual and merged tables. In the following, we define candidate triples as novel triples (not appearing in Wikidata) with their associated features; for example, the same triple appearing multiple times with different features will be counted multiple times. At the end of this section we will present numbers for unique triples.

CANDIDATES FROM INDIVIDUAL TABLES: A total of 14,670,522 (3,810,714 unique) existing triples in Wikidata were identified in the table rows. Using the method described in Section 4.1, we extracted 62,516,679 million candidate triples. As an initial evaluation of the precision of the candidate set, we sampled 100 unique triples, which were then labelled independently by two judges as CORRECT, INCORRECT, CONTEXTUAL and UNKNOWN. The judges agreed in 73/100 cases. Of these 73 cases, 10 (13.7%) were CORRECT and 63 (86.3%) were INCORRECT. We can thus estimate that there are approximately 8.6 million CORRECT candidates

MERGING TABLES: Merging 3,631,229 input tables by observed schema resulted in 1,135,977 merged tables, with on average ~ 3.2 individual tables forming each merged

table. Approximately 30% of the input tables were not merged with other tables, while approximately 30% were merged with at least one thousand other tables. The largest merged table consisted of 81,277 individual tables relating to political election results, with column labels {**party**, **candidate**, **vote**, **percentage**}. We remark that 273,341 (7.5%), 577,856 (15.9%) and 776,798 (21.4%) of the input tables, respectively, had hierarchical headers, multiple columns with the same name and columns without a name; i.e., we found a considerable number of tables for which the approach of Ling et al. [15] is not well-defined but that are supported by our approach based on observed schemata.

CANDIDATES FROM MERGED TABLES: From 1,135,977 tables merged by the observed schema, we extract 461,360,797 candidate triples: 398,844,124 more than for individual tables. To estimate precision, 100 unique triples are randomly sampled from those extracted from merged tables only, and labelled by two judges, who agreed on 88 cases, of which $\sim 4.5\%$ were CORRECT and $\sim 95.5\%$ INCORRECT. As a baseline, we performed the same experiment merging tables with the same schema incorporating hierarchical headers and ordinals but without datatypes (more formally, using partial column signatures $\sigma'_i(l_i, d_i)$ per Definition 1) where the two judges agreed on 89 cases, of which $\sim 2.2\%$ were CORRECT and $\sim 97.8\%$ were INCORRECT. Tables with columns of the same label (e.g., **rank**) but with different datatypes (e.g., NUM for numeric rank vs. STR for military rank) are often – but not always – unrelated. The results that follow are based on the more precise set of candidate triples extracted based on observed schemata with full signatures.

FILTERING MULTI-LINK CELLS: Based on the preliminary results, we observed that for individual tables, 59/100 candidate triples are extracted from a cell (subject, object or both) with more than one entity, of which 53 (89.8%) are INCORRECT. For the triples extracted (only) from merged tables, 35/100 come from such a cell, of which 32 (91.4%) are INCORRECT. We conclude that triples extracted from such cells add a lot of noise. While we could delegate the task of filtering such noise to the binary classifiers, in practice, randomly sampling and labelling triples from these sets would yield too few correct examples for training purposes. We thus decided to remove all candidate triples generated from a cell with multiple entities, leaving 18,587,101 and 134,058,618 million candidate triples in each set. Sampling 100 candidate triples from each filtered set, we find consensus on 84 cases for individual tables: 31 (36.9%) CORRECT, 51 (60.7%) INCORRECT, 2 (2.4%) CONTEXTUAL; we find consensus on 91 cases for merged tables: 4 (4.4%) CORRECT, 87 (95.6%) INCORRECT. The precision of candidate triples for individual tables increases considerably, but not so for merged tables; however, we have yet to apply refinement of merged tables based on incompatible properties.

CANDIDATES AFTER REFINEMENT: In experiments varying the compatibility threshold, we found that the refinement is not sensitive to a choice of value $\alpha \in [0.1, 0.9]$ since only 0.02% of the (non-reflexive) property pairs considered fall within this range of compatibility scores, with the vast majority of cases having no overlapping vertex pairs (a compatibility of 0), being clearly incompatible (a compatibility

TABLE 2
Training and test set

	Training set		Test set	
	CORRECT	INCORRECT	CORRECT	INCORRECT
I	191 (38%)	309 (62%)	38 (38%)	62 (62%)
M	74 (15%)	426 (85%)	13 (13%)	87 (87%)

in $(0, 0.1)$ or clearly compatible (a compatibility in $(0.9, 1]$). We thus choose $\alpha = 0.5$ as the threshold. Refining the 1,135,977 merged tables to separate incompatible properties resulted in 1,169,682 merged tables: 33,705 more merged tables ($\sim +2.9\%$) than before refinement. After refinement, we extract 112,180,480 candidate triples (83.7% of those extracted without refinement). To initially estimate precision, 100 triples are again randomly sampled and labelled by two judges. The judges agreed in 84 of cases: 6 (7.1%) CORRECT, 77 (91.7%) INCORRECT and 1 (1.2%) CONTEXTUAL, indicating a slight improvement in precision (4.4% to 7.1%).

FINAL CANDIDATES: To summarise, from tables that are not merged, we extract 18,587,101 unique candidate triples (considering features; not appearing in Wikidata), corresponding to 9,431,565 unique triples (not considering features) at an estimated precision of 36.9%. For merged tables after refinement, we extract 112,180,480 unique candidates, 62,109,161 of which correspond to unique triples at an estimated precision of 7.1%. Note that we do not remove candidates with duplicate triples as they have different feature values, and thus may yield different classification results: a single positive classification will be sufficient to include a triple in the final output. Though the precision of the candidates from merged tables is far lower, it contains many more (unique) correct triples, where we now see if binary classification succeeds in isolating these triples.

6.3 Results: Binary Classification

We now apply binary classifiers to identify correct triples from both sets of candidate triples and thus boost precision.

EXPERIMENTS: To address our core research questions (Q1–4), we perform experiments with respect to both sets of candidate triples: I is extracted from individual tables, while M is extracted from refined, merged tables (triples from multi-link cells are filtered in both cases). We associate these triples with three sets of features: \cdot^i denotes the original features for individual tables proposed in our previous work [14], \cdot^{i+} extends \cdot^i by adding the new features for individual tables proposed herein (see Section 4.2), while \cdot^m extends \cdot^i by adding features extracted from merged tables (see Section 5.3). We will then perform experiments over I^i (for Q1), I^{i+} (for Q2), I^m (for Q3) and M^m (for Q4).

LABELLED DATA: We randomly sampled 700 unique triples from I and 700 from M for training and testing. These 1,400 triples were labelled by two judges. From each of these sets, we take 600 triples classified by both judges as CORRECT or INCORRECT (discarding triples classified as CONTEXTUAL or UNKNOWN, or triples with disagreement). We split each dataset into 500 triples for training, and 100 triples for testing. Table 2 provides the number of triples per class.

TABLE 3

Top-10 features positively correlated and negatively correlated with the CORRECT label in training sets (Pearson’s ρ , $n = 500$) for I^m and M^m , where \cdot^+ indicates a new feature and \cdot^m indicates a group feature

Nº	I^m (pos.)		I^m (neg.)		M^m (pos.)		M^m (neg.)	
1	(46 ^m)	0.26	(35 ⁺)	-0.16	(46 ^m)	0.36	(49 ^m)	-0.25
2	(46)	0.24	(29)	-0.13	(44 ^m)	0.35	(31)	-0.17
3	(42 ^m)	0.21	(6)	-0.12	(46)	0.30	(15 ^m)	-0.16
4	(44 ^m)	0.21	(34 ⁺)	-0.10	(44)	0.29	(2 ^m)	-0.16
5	(44)	0.19	(31)	-0.10	(42 ^m)	0.28	(12 ^m)	-0.16
6	(40)	0.19	(37 ⁺)	-0.09	(45)	0.23	(11 ^m)	-0.15
7	(45)	0.18	(3)	-0.09	(48 ⁺)	0.22	(14 ^m)	-0.14
8	(42)	0.18	(48 ⁺)	-0.08	(42)	0.20	(8 ^m)	-0.14
9	(38)	0.17	(20 ⁺)	-0.08	(41 ^m)	0.18	(9 ^m)	-0.14
10	(45 ^m)	0.17	(2)	-0.07	(43 ^m)	0.18	(16)	-0.12

FEATURE CORRELATION: In Table 3 we present the most useful features in terms of being positively or negatively correlated with the CORRECT label (using Pearson’s ρ).⁹ First looking at individual tables, among the most positively correlated features with the CORRECT class were, respectively, features (41–46), relating to the number/ratio of rows for which the corresponding relation held; and features (38–40), relating to the string similarity between column names and property labels. The features most negatively correlated with the CORRECT class include: features (29,34,35,37) relating to cells having long text, formatting, etc.; feature (6) relating to the subject column ordinal; and features (2–3) indicating the number of table rows and columns. The most positively correlated features were similar for merged tables, but had higher correlation; conversely among the negatively correlated features, we see features (8,9,11,12,14,15,49) which all relate to having merged tables with many rows, indicating more INCORRECT triples in larger merged tables.

MODEL CONFIGURATION: We experiment with the six binary classifiers mentioned in Section 4.3. Given the large number of features, the class imbalance in the labelled data, as well as the presence of hyper-parameters in some classifiers, we consider four configurations for each model: (1) *default*: we apply classifiers with default hyper-parameters on the unmodified data; (2) *feature-selection*: we prune features that are highly intercorrelated and features not (inversely) correlated with the labelled classes; (3) *balancing*: we add positive triples that already exist in Wikidata to the training set until the classes are balanced; (4) *tuning*: we use Grid search to optimise the models’ hyper-parameters for the AUC metric using cross-validation on the training set. Of these four configurations, only *tuning* had a notable positive effect; *feature-selection* and *balancing* had a neutral or negative effect versus baseline results. Henceforth we present results with *tuning* as the most competitive configuration.

CLASSIFIER COMPARISON: In order to select the best-performing classifier of the six considered, in Table 4 we compare their results over the I dataset with three different feature sets. Though other classifiers sometimes provide better precision or recall for individual experiments, Extreme Gradient Boost (XGB) provided the best F_1 score for all three experiments (tied with Random Forests in one case). The

⁹ Features with positive or negative correlations are useful: high values for the former features suggest the positive class and for the latter features suggest the negative class, and vice versa for low values.

TABLE 4

Precision, recall and F_1 results for tuned models using Bagging Decision Trees (BDT), k -Nearest Neighbours (KNN), Linear Regression (LR), Naive Bayes (NB), Random Forests (RF) and Extreme Gradient Boost (XGB) classifiers on candidate triples extracted from individual (I) tables, using original features (\cdot^i), extended features (\cdot^{i+}) and merged features (\cdot^m), with best results per column in bold

Classifier	I^i			I^{i+}			I^m		
	P	R	F_1	P	R	F_1	P	R	F_1
BDT	0.71	0.66	0.68	0.74	0.68	0.71	0.70	0.55	0.62
KNN	0.77	0.18	0.30	0.75	0.32	0.44	0.50	0.16	0.24
LR	0.78	0.18	0.30	1.00	0.02	0.05	0.60	0.08	0.14
NB	0.44	0.87	0.58	0.41	0.92	0.59	0.67	0.11	0.18
RF	0.72	0.55	0.62	0.84	0.68	0.75	0.76	0.34	0.47
XGB	0.73	0.71	0.72	0.74	0.76	0.75	0.85	0.76	0.81

different feature sets have varying effects on the different models, which may be due to overfitting, but for XGB both precision and recall improve as more features are added, and less variance is seen. Applying 5-fold cross validation on the training set yielded similar results, with XGB displaying better results overall and less variance across the folds. We also obtained the AUC for each model, where XGB achieves 0.89 (I^i), 0.90 (I^{i+}) and 0.88 (I^m).

In further experiments with the M^m dataset, XGB had by far the best performance ($P = 0.85$, $R = 0.46$, $F_1 = 0.60$), but had a notable drop in recall and overall F_1 scores compared with the I dataset. One possible reason is the lower ratio of CORRECT examples in the M candidate set. We tried both oversampling and undersampling to improve these results, but found that they reduced the F_1 score, where XGB already provides good baseline results for imbalanced settings. Another approach to improve the results would be to annotate more data in order to have more CORRECT examples, but learning curves for XGB (in all cases) show that performance plateaus after 250–300 training examples (we provide 500) [43], which suggests that adding more training data will not improve performance. Lower performance in the M dataset may rather be attributable to a higher volume of difficult cases, i.e., triples with similar or even indistinguishable features that are correct in some cases and incorrect in others. These can occur from relations that are intermittently correct, such as proposing `wdt:P57 (director)` triples between two columns of a large merged table whose relation is better described by the `wdt:P161 (cast member)` property. We would expect the director relation to hold for some fraction of rows – since people sometimes act in the movies they direct – but the correct or incorrect cases would not be easily distinguishable by the features generated. Such cases occur more frequently in the M dataset as the `wdt:P57 (director)` need only hold for one row of the larger merged table in order for candidates to be proposed for all rows.

From these experiments, we conclude that the XGB classifier offers the best performance in our scenario.

6.4 Results: End-to-End

We now discuss results for the end-to-end process, which takes the Wikipedia tables and Wikidata knowledge graph as input, and outputs a set of novel triples not in Wikidata.

CLASSIFYING ALL CANDIDATES: We use XGB to classify the full sets of candidate triples (I and M), with the 600 labelled

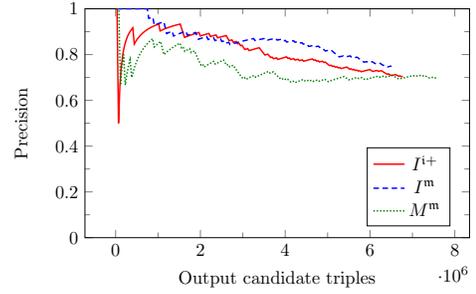


Fig. 5. Precision vs. output candidate triples for the three models

triples for I and M used, respectively, for training. Having already shown the benefits of the novel features for XGB, we train models over I^{i+} , I^m and M^m . Comparing I^{i+} and I^m will allow us to see the effect of adding features extracted from merged tables. Comparing I^{i+} and M^m will allow us to see the effect of adding features and candidates extracted from merged tables. Comparing I^m and M^m will allow us to see the effect of adding candidates extracted from merged tables when features from merged tables are already considered. We classify and output CORRECT candidate triples.

The classifiers for I^{i+} , I^m and M^m output 4,809,323, 4,881,566 and 5,313,880 candidate triples (with features), respectively. These correspond to 3,565,603, 3,396,017 and 3,990,832 unique triples, respectively. Table 5 enumerates the top-10 properties corresponding to the output of I^{i+} , I^m and M^m . Amongst others, we see properties relating to geography (P17, P131, P150, P276), sports (P54, P118, P1532) and cinema (P161), reflecting common themes for tables on Wikipedia. While the properties for I^{i+} are similar to I^m , those for M^m are quite different (we will return to this later).

In the following results, we consider candidate triples as the result for a unique triple can differ due to having different features; later we will analyse unique triples.

We found that 17.7%, 17.7% and 49.8% of the candidates from I^{i+} , I^m and M^m , respectively, use the protagonist column (0.7%, 0.7% and 2.9%, respectively, involve list articles).

To perform a validation of the output candidate triples, we sample 200 from each set (for I^{i+} , I^m and M^m), which are pooled into a set of 600 triples labelled by two judges without knowing to which set they belong; this time, cases with disagreement were discussed and a consensus reached.

In Figure 5, we show the precision (based on the 600 manually labelled candidates) versus the number of output candidate triples (labelled CORRECT by the classifier) when varying the models' threshold. We see that I^m maintains a higher precision than I^{i+} , while M^m maintains lower precision than I^{i+} early on, but plateaus to a similar precision level (~ 0.7) as I^{i+} and continues to output further triples.

Turning to unique triples, Table 6 summarises the number of unique (novel) triples (removing features) output by the classifiers, along with their corresponding validation results, estimated precision, precision error bound, and an estimate of the number of correct triples in the output. Comparing I^{i+} and I^m , we see that I^m offers a higher precision (0.705 vs. 0.750), and outputs 1.3% more correct triples than I^{i+} . Comparing I^{i+} and M^m , M^m shows a slight drop in precision (0.705 vs. 0.700), but outputs 11.1%

TABLE 5

Top-10 properties by number of (output) candidate triples classified as CORRECT by I^{i+} , I^m , M^m

№	I^{i+}		I^m		M^m	
1	P131	731,444	P131	723,321	P131	751,680
2	P1532	467,984	P1532	462,402	P361	625,855
3	P361	423,480	P361	431,735	P710	608,648
4	P161	419,449	P161	409,990	P17	568,984
5	P150	347,913	P150	337,432	P1344	356,513
6	P54	279,483	P54	311,662	P161	296,085
7	P276	254,622	P276	246,795	P166	276,335
8	P1344	193,085	P1344	185,312	P118	258,144
9	P102	176,179	P102	182,646	P527	252,495
10	P166	168,721	P166	166,702	P1532	192,940

P17 country, P54 member of sports team, P102 member of political party, P118 league, P131 located in the administrative territorial entity, P150 contains administrative territorial entity, P161 cast member, P166 award received, P276 location, P361 part of, P527 has part, P710 participant, P1344 participant of, P1532 country for sport

TABLE 6

Final results for three models and their combinations using XGB including unique output triples classified CORRECT, and the result of validating $n = 200$ sample output triples, indicating the number of triples labelled CORRECT (CR.), INCORRECT (IN.), CONTEXTUAL (CN.), and UNKNOWN (UN.), as well as precision ($P = \frac{CR.}{n}$), standard error of the mean ($\sigma_P = \sqrt{\frac{P(1-P)}{n}}$), and the estimated correct triples

Model	Output (Unique)	Validation				P	σ_P	Correct (Unique)
		CR.	IN.	CN.	UN.			
I^{i+}	3,565,603	141	53	4	2	0.705	± 0.032	2,513,750
I^m	3,396,017	150	49	0	1	0.750	± 0.031	2,547,013
M^m	3,990,832	140	57	1	2	0.700	± 0.031	2,793,582
$I^{i+} \cup I^m$	3,899,785	291	102	4	3	0.728	± 0.022	2,837,094
$I^{i+} \cup M^m$	5,719,275	281	110	6	4	0.703	± 0.023	4,017,791
$I^m \cup M^m$	5,525,730	290	106	1	3	0.725	± 0.022	4,006,154
$I^{i+} \cup I^m \cup M^m$	5,936,091	431	159	5	5	0.718	± 0.018	4,262,113

more correct triples than I^{i+} . Comparing I^m and M^m , M^m shows a notable drop in precision (0.750 vs. 0.700), but outputs 9.6% more correct triples than I^m .

Combining the output of the three classifiers, $I^{i+} \cup I^m$ yields a modest increase (11.4% more triples than I^m). However, combining the results from M^m yields a significant increase (with $I^{i+} \cup I^m \cup M^m$ yielding 69.6%, 67.3% and 52.6% more correct triples than output by I^{i+} , I^m and M^m). Though one might expect the output for M^m to subsume that of the other models (as M^m contains all the candidate triples and features of the other models), this is not the case. The candidate sets for I and M are sufficiently different – only one sixth of the candidates of M appear in I – such that they benefit from having separately trained models.

Ultimately we can extract 5.9 million unique triples that do not exist in Wikidata with a precision of 0.718 ± 0.018 .

RECALL: We have not included recall measures for the end-to-end process since it is unclear how we should define the set of true outputs – i.e., the set of all correct Wikidata triples that can be extracted from Wikipedia tables – and whether or not it should include, for example, triples extracted vertically from columns, from text within cells, from datatype values, from imprecise values, etc. It is further unclear how to handle true triples that are not directly stated, e.g., capital of triples extracted from a table indicating the largest cities of different countries. However, although we cannot precisely define absolute recall for the end-to-end process, if we

assume the set of true outputs to be fixed, then we can say that approaches extracting more (correct) triples have higher recall since the denominator (the number of true outputs) is fixed across all approaches. Thus we can estimate (e.g.) that the recall of $I^{i+} \cup I^m \cup M^m$ is $1.696 \times$ that of I^{i+} , even if we do not know the absolute recall values.

SUMMARY: We now reflect back on our research questions:

- Q1: We extract 3.6 million novel triples from the individual tables at a precision of 0.705 (Table 4).
- Q2: The novel features extracted from individual tables enable slight improvements for five of the six binary classifiers considered, where the F_1 -score for the best classifier, XGB, increases from 0.72 to 0.75 (Table 4).
- Q3: For candidate triples from individual tables, features from merged tables improve precision by 4–8 percentage points (Figure 5).
- Q4: Approximately 280 thousand (+11.1%) more unique, novel, correct triples are extracted from merged tables (M^m) versus individual tables (I^{i+}) at the cost of a slight drop in precision (from 0.705 to 0.700). Considering only the additional features, and not the additional candidates, from merged tables (I^m), the number of correct triples output increases slightly (+1.3%) versus individual tables (I^{i+}), but precision increases significantly (from 0.705 to 0.700); thus the results for I^m are strictly better than I^{i+} , producing more correct triples at a higher precision. Comparing triples extracted from merged tables (M^m) versus individual tables with merged features (I^m), approximately 247 thousand (+9.6%) more unique, novel, correct triples are extracted from merged tables (M^m) at the cost of a notable drop in precision (from 0.750 to 0.700). If we combine all results for individual and merged tables ($I^{i+} \cup I^m \cup M^m$), we can extract approximately 1.75 million (+69.6%) more unique, novel, correct triples as a result of merging (versus I^{i+}), with a slightly better precision of 0.718 vs. 0.705 (Table 6). These combined results are again strictly better than those for individual tables alone.

7 CONCLUSIONS

We have proposed a distantly supervised method to extract triples from Wikipedia tables using Wikidata as a reference knowledge graph. When applied to individual tables, this process extracts 3.6 million triples with a precision of 0.705 ± 0.032 . Adding features from merged tables to the process, we extract 3.4 million triples with a precision of 0.750 ± 0.031 (with more correct triples output than before). Adding features and candidate triples from merged tables, we extract 4.0 million triples with a precision of 0.700 ± 0.031 . Combining the results for all three configurations outputs 5.9 million triples with a precision of 0.718 ± 0.018 . These results show that merging tables facilitates extracting more triples at similar or higher precision.

In terms of future work, it would be interesting to explore how methods for fact checking [45] or knowledge graph refinement [46] could be leveraged to refine candidate sets. Another interesting direction would be to apply knowledge fusion techniques, for example to choose a single

value from various extracted values for a given functional or inverse-functional property on a subject/object [1]. Other techniques that may enable improvements are word embeddings (for matching properties and column names), graph embeddings (for plausibility scores), etc. Generating merged tables of thousands or hundreds of thousands of rows also suggests that semi-supervised methods may become practical, where input from an expert could help to extract large batches of triples with relatively little cost. It would also be of interest to adapt and evaluate the proposed methods for HTML tables taken from the broader Web.

ACKNOWLEDGMENTS

This work was funded by Fondecyt Grant No. 1181896, CONICYT-PCHA/Doctorado Nacional/2016-21160017, and ANID Millennium Science Initiative Program ICN17_002.

REFERENCES

- [1] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang, "From Data Fusion to Knowledge Fusion," *PVLDB*, vol. 7, no. 10, pp. 881–892, 2014.
- [2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Sci. Am.*, vol. 284, no. 5, pp. 34–43, 2001.
- [3] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.
- [4] M. Schmachtenberg, C. Bizer, and H. Paulheim, "Adoption of the Linked Data Best Practices in Different Topical Domains," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2014, pp. 245–260.
- [5] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, pp. 78–85, 2014.
- [6] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, and A. Bielefeldt, "Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2018, pp. 376–394.
- [7] T. E. Putman, S. Lelong, S. Burgstaller-Muehlbacher, A. Waagmeester, C. M. Diesh, N. A. Dunn, M. C. Muñoz-Torres, G. S. Stupp, C. Wu, A. I. Su, and B. M. Good, "WikiGenomes: an open web application for community consumption and curation of gene annotation data in Wikidata," *Database*, vol. 2017, p. bax025, 2017.
- [8] P. Mika, E. Meij, and H. Zaragoza, "Investigating the Semantic Gap through Query Log Analysis," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2009, pp. 441–455.
- [9] R. Meusel, P. Petrovski, and C. Bizer, "The WebData-Commons Microdata, RDFa and Microformat dataset series," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2014, pp. 277–292.
- [10] J. Martínez-Rodríguez, A. Hogan, and I. López-Arévalo, "Information extraction meets the Semantic Web: A survey," *Semantic Web*, vol. 11, no. 2, pp. 255–335, 2020.
- [11] H. Chen, S. Tsai, and J. Tsai, "Mining tables from large scale HTML texts," in *Int. Conf. on Computational Linguistics (COLING)*. Morgan Kaufmann, 2000, pp. 166–172.
- [12] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "WebTables: exploring the power of tables on the Web," *PVLDB*, vol. 1, no. 1, pp. 538–549, 2008.
- [13] E. Crestan and P. Pantel, "Web-scale table census and classification," in *Web Search and Web Data Mining (WSDM)*. ACM, 2011, pp. 545–554.
- [14] E. Muñoz, A. Hogan, and A. Mileo, "Using Linked Data to mine RDF from Wikipedia's tables," in *Web Search and Web Data Mining (WSDM)*. ACM, 2014, pp. 533–542.
- [15] X. Ling, A. Y. Halevy, F. Wu, and C. Yu, "Synthesizing Union Tables from the Web," in *Int. Joint Conf. on Artificial Intelligence (IJCAI)*. IJCAI/AAAI, 2013, pp. 2677–2683.
- [16] O. Lehmsberg and C. Bizer, "Stitching Web Tables for Improving Matching Quality," *PVLDB*, vol. 10, no. 11, pp. 1502–1513, 2017.
- [17] G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and Searching Web Tables Using Entities, Types and Relationships," *PVLDB*, vol. 3, no. 1, pp. 1338–1347, 2010.
- [18] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu, "Recovering Semantics of Tables on the Web," *PVLDB*, vol. 4, no. 9, pp. 528–538, 2011.
- [19] M. Cannaviccio, L. Ariemma, D. Barbosa, and P. Meriardo, "Leveraging Wikipedia Table Schemas for Knowledge Graph Augmentation," in *Int. Workshop on the Web and Databases (WebDB)*. ACM, 2018, p. 5.
- [20] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovic, and R. Studer, "Transforming arbitrary tables into logical form with TARTAR," *Data Knowl. Eng.*, vol. 60, no. 3, pp. 567–595, 2007.
- [21] Y. Wang and J. Hu, "A machine learning based approach for table detection on the web," in *World Wide Web Conf. (WWW)*. ACM, 2002, pp. 242–250.
- [22] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.
- [23] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia," *Artif. Intell.*, vol. 194, pp. 28–61, 2013.
- [24] V. Mulwad, T. Finin, and A. Joshi, "Semantic message passing for generating Linked Data from tables," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2013, pp. 363–378.
- [25] S. Zwicklbauer, C. Seifert, and M. Granitzer, "DoSeR – A knowledge-base-agnostic framework for entity disambiguation using semantic embeddings," in *Extended Semantic Web Conf. (ESWC)*. Springer, 2016, pp. 182–198.
- [26] P. Buche, J. Dibia-Barthélemy, L. Ibanescu, and L. Soler, "Fuzzy Web data tables integration guided by an ontological and terminological resource," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 805–819, 2013.

- [27] H.-J. Dai, C.-Y. Wu, R. Tsai, W.-L. Hsu *et al.*, "From Entity Recognition to Entity Linking: a survey of advanced Entity Linking techniques," in *Japanese Society for Artificial Intelligence (JSAI)*, 2012.
- [28] C. S. Bhagavatula, T. Noraset, and D. Downey, "TabEL: Entity Linking in Web tables," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2015, pp. 425–441.
- [29] Y. Ibrahim, M. Riedewald, and G. Weikum, "Making Sense of Entities and Quantities in Web Tables," in *Int. Conf. on Information and Knowledge Management (CIKM)*. ACM, 2016, pp. 1703–1712.
- [30] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, and V. Christophides, "Matching Web Tables with Knowledge Base Entities: From Entity Lookups to Entity Embeddings," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2017, pp. 260–277.
- [31] X. Luo, K. Luo, X. Chen, and K. Q. Zhu, "Cross-Lingual Entity Linking for Web Tables," in *AAAI Conf. on Artificial Intelligence*. AAAI Press, 2018, pp. 362–369.
- [32] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu, "Understanding Tables on the Web," in *Int. Conf. on Conceptual Modeling (ER)*. Springer, 2012, pp. 141–155.
- [33] Z. Zhang, "Effective and efficient semantic table interpretation using Tableminer⁺," *Semantic Web*, vol. 8, no. 6, pp. 921–957, 2017.
- [34] O. Lehmborg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer, "The Mannheim Search Join Engine," *J. Web Sem.*, vol. 35, pp. 159–166, 2015.
- [35] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmman, S. Sun, and W. Zhang, "Knowledge vault: a Web-scale approach to probabilistic knowledge fusion," in *Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2014, pp. 601–610.
- [36] H. Gonzalez, A. Y. Halevy, A. Langen, J. Madhavan, R. McChesney, R. Shapley, W. Shen, and J. Goldberg-Kidon, "Socialising Data with Google Fusion Tables," *IEEE Data Eng. Bull.*, vol. 33, no. 3, pp. 25–32, 2010.
- [37] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "InfoGather: entity augmentation and attribute discovery by holistic matching with web tables," in *Int. Conf. on Management of Data (SIGMOD)*. ACM, 2012, pp. 97–108.
- [38] C. S. Bhagavatula, T. Noraset, and D. Downey, "Methods for exploring and mining tables on Wikipedia," in *ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA@KDD)*. ACM, 2013, pp. 18–26.
- [39] M. Yoshida, K. Torisawa, and J. Tsujii, "Extracting ontologies from World Wide Web via HTML tables," in *Pacific Association for Computational Linguistics (PACLING)*, 2001, pp. 332–341.
- [40] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller, "Table Union Search on Open Data," *PVLDB*, vol. 11, no. 7, pp. 813–825, 2018.
- [41] Y. Wang and Y. He, "Synthesizing Mapping Relationships Using Table Corpus," in *Int. Conf. on Management of Data (SIGMOD)*. ACM, 2017, pp. 1117–1132.
- [42] F. Mahdisoltani, J. Biega, and F. Suchanek, "Yago3: A knowledge base from multilingual Wikipedias," in *Conf. on Innovative Data Systems Research (CIDR)*. CIDR Conf., 2014.
- [43] J. Luzuriaga, A. Hogan, E. M. noz, and H. Rosales, "Wikitablets," Oct. 2019. [Online]. Available: <https://doi.org/10.5281/zenodo.3483254>
- [44] J. Luzuriaga, "Merging HTML Tables for Extracting Relations," Master's thesis, University of Chile, 2019.
- [45] G. L. Ciampaglia, P. Shiralkar, L. M. Rocha, J. Bollen, F. Menczer, and A. Flammini, "Computational fact checking from knowledge networks," *PLOS ONE*, vol. 10, no. 6, pp. 1–13, 06 2015.
- [46] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web J.*, vol. 8, no. 3, pp. 489–508, 2017.



Jhomara Luzuriaga received a B.Eng. from the Universidad Nacional de Loja, Ecuador in 2012, after which she worked as a software developer. She received an M.Sc. in Computer Science from Universidad de Chile in 2019, working on Information Extraction and Machine Learning topics. She is interested in the integration of client applications with Machine Learning and Knowledge Graphs. She is currently working as a Data Analyst for Promerica Produbanco-Ecuador, integrating data with web services.



Emir Muñoz received a B.Eng. and M.Sc. in Computer Engineering from Universidad de Santiago, Chile, in 2009 and 2011, respectively. He is pursuing a Ph.D. in Computer Science at the National University of Ireland, Galway, on the topic of Knowledge Graph mining. He is also a Senior Machine Learning Engineer in the AI Group at Genesys, where he works on applying ML for improving customer experience.



Henry Rosales-Méndez received his B.Sc. from the Universidad de Oriente, Cuba, in 2012, where he continued as a lecturer until 2014. He began his Ph.D., focussing on Information Extraction, at the Universidad de Chile in 2016, under the supervision of Aidan Hogan and Barbara Poblete. He has been a PC Member in the International Workshops/Conferences: SOTICS'19, OM'19, ECAI'20. He is also a member of the Chilean Association of Pattern Recognition.



Aidan Hogan received a B.Eng. and Ph.D. from the National University of Ireland, Galway, in 2006 and 2011, respectively. He is currently an Associate Professor at the Department of Computer Science, Universidad de Chile, where he also holds the position of Associate Researcher in the Millennium Institute for Foundational Research on Data (IMFD). His primary research interests centre on the Semantic Web, Information Extraction and Graph Databases.