Extracting Novel Facts from Tables for Knowledge Graph Completion (Extended version)

Benno Kruit[1][2], Peter Boncz[1], and Jacopo Urbani[2]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
[2] Dept. of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
{kruit, p.boncz}@cwi.nl; jacopo@cs.vu.nl

Abstract. We propose a new end-to-end method for extending a Knowledge Graph (KG) from tables. Existing techniques tend to interpret tables by focusing on information that is already in the KG, and therefore tend to extract many redundant facts. Our method aims to find more novel facts. We introduce a new technique for table interpretation based on a scalable graphical model using entity similarities. Our method further disambiguates cell values using KG embeddings as additional ranking method. Other distinctive features are the lack of assumptions about the underlying KG and the enabling of a fine-grained tuning of the precision/recall trade-off of extracted facts. Our experiments show that our approach has a higher recall during the interpretation process than the state-of-the-art, and is more resistant against the bias observed in extracting mostly redundant facts since it produces more novel extractions.

1 Introduction

Motivation. Much of the world's information exists as tabular data. These are available as HTML tables on web pages, as spreadsheets, or as publicly available datasets in many different formats. There has been more than a decade of research in recognizing, cleaning and capturing these so-called *web tables* [4]. Because of their relational nature, such large collections of web tables are suitable for supporting table search [32] or for answering specific factual queries [29]. In certain web tables, the rows describe attributes or relationships of entities. This makes them suitable sources for extending the coverage of Knowledge Graphs (KGs), which is a task known as *KG completion*.

Problem. In order to perform KG completion from web tables, we must first align their structure and content with the KG, a problem broadly referred to as *table interpretation*. Table interpretation has been the subject of several prior works [16,27,33,17,18,28,9,12]. Similar to our research, these works primarily focus on the interpretation of entity tables, i.e., tables where each row describes one entity and columns represent attributes. In this case, the interpretation process consists of two operations. First, each row is linked with an entity in the

KG, and optionally the entire table is linked to a class. Then, each column is associated to a KG relation.

After the table is correctly interpreted, we can extract novel triples from the table and add them to the KG. This last operation is also known as *slot-filling*, as the empty 'slots' in the KG are filled with new facts [27]. Table interpretation strongly affects the quality of slot-filling, since errors in the former can no longer be corrected. Because of this, state-of-the-art table interpretation techniques (an overview is given in Section 6) aim for high precision by pruning out many potential assignments already at early stages. While high precision is desirable in some contexts (e.g., table search), it has been observed [14] that this strategy leads to a high number of redundant extractions during slot-filling, since only the assignments to entities that are well-covered in the KG are retained.

Contribution. With the goal of maximizing the number of novel extractions without sacrificing precision, we present a new method for KG completion from web tables. In contrast to existing approaches, our method does not prune out row-entity assignments, but performs the interpretation by performing inference over all possible assignments using a Probabilistic Graphical Model (PGM). The PGM uses label similarities as priors, and then updates its likelihood scoring to maximise the *coherence* of entity assignments across the rows using Loopy Belief Propagation (LBP). Coherence is not computed using a predefined metric (such as class membership) but is automatically selected as a combination of properties that are shared by the entities in the table. This is a novel feature of our method which makes it capable of working with KGs with different topologies and/or relations. Since we use both label similarities and coherence based on salient common attributes, our method is able to maintain a high accuracy for the row-entity assignments. At the same time, it is also able to return many more novel extractions since we did not prune out any assignments.

We also propose an approach to perform slot-filling by disambiguating attribute cells in a novel link-prediction framework. Our approach makes use of embeddings of KG entities and relations to improve the quality of the disambiguation whenever label matching is not sufficient. This furthers our aim to find novel facts for KG completion.

We compared our method to several state-of-the-art systems. Additionally, we evaluated the performance of these systems with regard to the redundancy of the facts that they extract from the tables. Our experiments on popular benchmark datasets show that our approach yields slightly lower precision, but significantly higher recall on entity predictions. This leads to many more novel extractions than what is possible with existing methods. Finally, to test the scalability of our method we perform a large-scale evaluation on 786K tables from Wikipedia. An extended version of this paper is available at [15].

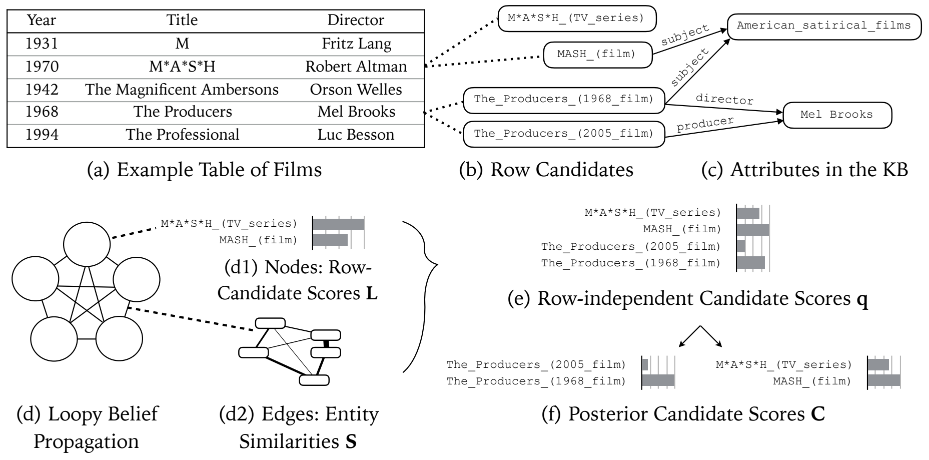
2 Background

KGs. A KG $\mathcal{K}$ is a repository of factual knowledge that can be seen as a directed labeled graph where the nodes are entities and the edges represent semantic relations. We define $\mathcal{K}$ as a tuple $(\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$ is the set of entities (nodes), $\mathcal{R}$ is the set of relations, and $\mathcal{F}$ is the set of facts (edges) in the graph. Each entity is associated to a finite set of labels $\mathsf{Labels}(e)$. We use the notation $\langle s, r, o\rangle$ to denote a fact in $\mathcal{F}$ where $s, o \in \mathcal{E}$ and $r \in \mathcal{R}$. Realistic KGs contain facts of various types: For instance, they either indicate type memberships (e.g., ⟨`Netherlands`, `type`, `Country`⟩), or encode more generic binary relations (e.g., ⟨`Amsterdam`, `capitalOf`, `Netherlands`⟩), and are normally encoded in RDF [11].

Table Interpretation. Tables represent an important source of knowledge that is not yet in the KG. A class of tables that is particularly useful for enriching KGs is the one that contains *entity tables*, i.e., tables where one column contains the name of entities (the *key-column*) and all others contain the entity attributes. While these tables ostensibly contain structured data, the textual content of cells and identifiers is created more with the aim of human interpretation than automatic processing. To capture the semantics of these tables in a coherent and structured way, it is useful to link their content to concepts in the KG. We refer to this task as *table interpretation*, mapping each row and attribute column to entities and relations respectively. These mappings can be computed by determining (1) which entities in the KG are mentioned in the table, (2) which are the types of those entities, and (3) which relations are expressed between columns (if any) [16,30,17,26,33]. After the interpretation is finished, we can use the mappings to construct facts for the KG. We call this operation *slot-filling*.

Example 1. Consider an example KG with five entities {`Netherlands`, `Country`, `Amsterdam`, `City`, `capitalOf`} and a table X which contains a row r with cell values $r[1]$ = "Holland" and $r[2]$ = "A'dam". The first cell value should be mapped to the entity `Netherlands`, while the second should be mapped to the entity `Amsterdam`. The mapping is not trivial because a string can map to multiple entities, e.g., "Holland" can refer either to the county or to 19 different cities in the U.S. The task of table interpretation consists of disambiguating the correct meaning intended in the table. Furthermore, if the other rows in the table also contain countries and their capital cities, then the system should infer that all these entities are instances of classes such as `Country` and `City`, and the relation between the columns should be `capitalOf`. With slot-filling, our goal is to extract statements like ⟨`Amsterdam`, `capitalOf`, `Netherlands`⟩ from the table so that they can be added to the KG.

PGMs. In this paper, we employ Probabilistic Graphical Models (PGMs) to perform the interpretation. PGMs are a well-known formalism for computing joint predictions [21]. For a given set of random variables, conditional dependences between pairs of variables are expressed as edges in a graph. In these graphs, variables are connected if the value of one influences the value of another. The connection is directed if the influence is one-way, and undirected if both variables influence each other. The behaviour of the influence on every edge is expressed by a function known as the *potential function*. When performing inference in a PGM, information from the nodes is propagated through the



Year	Title	Director
1931	M	Fritz Lang
1970	M*A*S*H	Robert Altman
1942	The Magnificent Ambersons	Orson Welles
1968	The Producers	Mel Brooks
1994	The Professional	Luc Besson

(a) Example Table of Films
M*A*S*H_(TV_series)
MASH_(film)
The_Producers_(1968_film)
The_Producers_(2005_film)
American_satirical_films
Mel Brooks
subject
subject
director
producer
(b) Row Candidates
(c) Attributes in the KB
M*A*S*H_(TV_series)
MASH_(film)
(d1) Nodes: Row-Candidate Scores L
(d2) Edges: Entity Similarities S
(d) Loopy Belief Propagation
M*A*S*H_(TV_series)
MASH_(film)
The_Producers_(2005_film)
The_Producers_(1968_film)
(e) Row-independent Candidate Scores q
The_Producers_(2005_film)
The_Producers_(1968_film)
M*A*S*H_(TV_series)
MASH_(film)
(f) Posterior Candidate Scores C

Fig. 1: Schematic representation of our method.

network using the potential functions in order to determine the final distribution of the random variables.

KG Embeddings. We also make use of latent representations of the KG [20] to filter out incorrect extractions. In particular, we consider *TransE* [3], one of the most popular methods in this category. The main idea of TransE is to "embed" each entity and relation into a real-valued d-dimensional vector (where $d > 0$ is a given hyperparameter). The set of all vectors constitutes a model Θ of $|\mathcal{E}|d + |\mathcal{R}|d$ parameters which is trained so that the distance between the vectors of entities which are connected in $\mathcal{K}$ is smaller than the distance between the ones of entities which are not connected.

Training Θ is done by minimizing the loss function

$$\mathcal{L}_\Theta = \sum_{\langle s,r,o\rangle \in \mathcal{F}} \sum_{\langle s',r,o'\rangle \in S_{\langle s,r,o\rangle}} [\gamma + d(\mathbf{s}+\mathbf{r},\mathbf{o}) - d(\mathbf{s}'+\mathbf{r},\mathbf{o}')]_+ \tag{1}$$

where: $\mathbf{s}, \mathbf{s}', \mathbf{o}, \mathbf{o}', \mathbf{r}$ are the vectors associated to the entities s, s', o, o' and type r respectively; $\gamma \geq 0$ is an hyperparameter that defines the minimum acceptable margin; $d(\cdot)$ is a distance function (typically the L_1 norm), $[x]_+$ returns the positive part of x, and $S_{\langle s,r,o\rangle} = \{\langle s,r,o'\rangle \mid \langle s,r,o'\rangle \notin \mathcal{F}\} \cup \{\langle s',r,o\rangle \mid \langle s',r,o\rangle \notin \mathcal{F}\}$, i.e., it is a set of "corrupted" facts which are not in $\mathcal{K}$. Once training is completed, the model can be used to perform link prediction, i.e., to estimate the likelihood of unseen facts: if the distance of these facts is small, then these are more likely to be true.

3 Table Interpretation

We introduce our method for performing table interpretation. Fig. 1 shows the computation that takes place during the interpretation, using table (a) as a motivating example. In this case, the key-column is the second one ("title") but its content is ambiguous since the values can refer to movies, TV series, or books. For instance, the second row can refer to the TV serial `M*A*S*H` or to the movie `MASH`, as is shown in Fig. 1b. The goal of this task is to map as many rows ρ as possible to corresponding entities in $\mathcal{E}$ and each column c to one relation in $\mathcal{R}$. To this end, we perform a sequence of five operations, described below.

3.1 Step 1: Candidate Entity Selection

First, we identify the key-column (if any) using the heuristics proposed by [26], which consists of selecting the column with most unique non-numeric values breaking ties by choosing the leftmost one. This heuristics works well in practice so we apply it without modifications. Only the tables with valid key columns are considered since these are the only ones for which we can (potentially) extract factual knowledge.

For every cell in the key column, we then select a set of entity candidates. We represent this computation with the function $\mathsf{Cand}(\rho)$ which takes in input a generic row ρ and returns all entities in $\mathcal{E}$ which are potential candidates with ρ. This function is implemented by 1) indexing all the labels in $\mathcal{K}$, 2) retrieving the labels which contain the cell value of the key column, 3) returning the entities associated to the labels. Let $e \in \mathsf{Cand}(\rho)$ be a potential entity candidate for row ρ. We call the tuple (ρ, e) a *row-entity assignment.* If $\mathsf{Cand}(\rho)$ is empty, then ρ is ignored. Otherwise, the table interpretation process will determine which row-entity assignment should be selected.

Example 2. In the table (a) of figure 1, we assume that the first column is the key column, because it is the leftmost column with non-numeric unique values. The label index returns a set of scored candidate entities per row (b).

The label matches are ranked using length-normalised smoothed TF-IDF. In our case, the query corresponds to the cell value of the key column, while the documents are all labels in $\mathcal{K}$. Identically to [26], we (1) take only the first result if it is much better than the next and (2) take the top three labels otherwise. The final set of candidates consists of all entities associated with these labels.

Typically entities are explicitly linked to labels with direct relations (e.g., `rdfs:label` [11]). However, more links can be retrieved if we also consider titles and disambiguation pages. In our approach, we add also these labels to the index because we observed that this leads to a substantial increase of the recall. At this stage, it is important to have a high recall because the subsequent operations cannot recover in case we fail to retrieve the correct mapping. In the definitions below, we denote these sets of labels for each entity as $\mathsf{Labels}(e)$.

3.2 Step 2: Computation of the Priors

In this step, we compute a score of the row-entity assignments by comparing all cell values in the row with all the labels of entities that are connected to the candidate entities. To this end, we first define attribute links, and related labels of an entity e as

$$\mathsf{Links}(e) = \{\langle r, v\rangle \mid \langle e, r, v\rangle \in \mathcal{F}\} \tag{2}$$

$$\mathsf{LinkLabels}(e, r) = \{l \mid \langle r, v\rangle \in \mathsf{Links}(e), l \in \mathsf{Labels}(v)\} \tag{3}$$

Intuitively, $\mathsf{Links}(e)$ contains all links of e while $\mathsf{LinkLabels}(e, r)$ represents the labels at the other end of the r-links from e. Then, we introduce the function

$$\mathsf{Match}(c, \rho, e, r) = \max_{s \in \mathsf{Cell}(c,\rho)} \max_{l \in \mathsf{LinkLabels}(e,r)} \mathsf{TokenJaccard}(s, l) \tag{4}$$

to compute the highest attainable string similarity between the cell at column c and row ρ and the values of the r-links from e. Here, $\mathsf{Cell}(i, j)$ returns the content of the cell at row i and column j in a table with n rows and m columns, while $\mathsf{TokenJaccard}$ is the Jaccard index $J(A, B) = \frac{|A\cap B|}{|A\cup B|}$ of the tokens in each string. For instance, in the table in Fig. 1 each cell is matched to each attribute of the corresponding row-entity candidates, e.g., $\mathsf{Match}(3, 4, \texttt{The\_Producers\_(1968\_film)}, \texttt{director})$ is the score that quantifies to what extent the content of the cell at coordinates $(3, 4)$ matches the string "Mel Brooks", which is the label of the director of the film. Note that we treat the content of every cell as a string. There are some approaches that use type-specific cell and column matching methods [26,22,33,16], but a combination of our method with these techniques should be seen as future work.

We can now compute likelihood scores for mapping cells to relations (Eq. 5), and for mapping columns to relations (Eq. 6) to aggregate and normalise these scores on the row and column levels respectively:

$$\mathsf{CellScore}(c, \rho, r) = \frac{1}{|\mathsf{Cand}(\rho)|} \sum_{e \in \mathsf{Cand}(\rho)} \mathsf{Match}(c, \rho, e, r) \tag{5}$$

$$\mathsf{ColScore}(c, r) = \frac{\sum_{i=0}^{n} \mathsf{CellScore}(c, \rho_i, r)}{\sum_{i=0}^{n} \sum_{r' \in \mathcal{R}} \mathsf{CellScore}(c, \rho_i, r')} \tag{6}$$

For instance, in Fig. 1a, $\mathsf{CellScore}(4, 3, \texttt{director})$ returns the likelihood that the cell (4,3) matches the relation $\texttt{director}$, while $\mathsf{ColScore}(3, \texttt{director})$ returns the aggregated scores for column 3 considering all rows in the table.

Since $\mathsf{ColScore}(c, r)$ is the likelihood score that column c maps to relation r, we can use this value to construct the prior distribution of all assignments to c. Furthermore, we can use these scores to refine the likelihood of the possible row-entity matchings. We compute such likelihood as

$$\mathsf{RowScore}(\rho, e) = \frac{1}{m} \sum_{i=0}^{m} \max_{r \in \mathcal{R}} \mathsf{ColScore}(c_i, r) \times \mathsf{Match}(c_i, \rho, e, r) \tag{7}$$

In essence, Eq. 7 computes the likelihood of an entity-row matching as the average best product that each cell matches to a certain attribute (r, e) ($\mathsf{Match}(\cdot)$) with the general likelihood that the column matches to r ($\mathsf{ColScore}(\cdot)$). We use the values of $\mathsf{RowScore}$ to build a prior distribution for all entity-row matches.

3.3 Step 3: Entity Similarity Scores

Both prior distributions computed with Eqs. 5 and 6 rely on the Jaccard Index. Thus, they are distributions which are ultimately built on the string similarities between the strings in the cells and the entities' labels. We use these scores to compute similarity scores between pairs of candidate entities across the rows. In the next step, we will use these similarities to compute better entity-row likelihood scores than the ones of Eq. 7.

First, we weigh all links $\langle r, v\rangle$ depending on their popularities across the entities in the table and the corresponding prior of the assignments that use them. To this end, we define the function $\mathsf{LinkTotal}$ as

$$\mathsf{LinkTotal}(r, v) = \sum_{i=0}^{n} \max_{e \in \mathsf{Cand}(\rho_i)} \mathsf{RowScore}(\rho_i, e)[\langle r, v\rangle \in \mathsf{Links}(e)] \tag{8}$$

where $[x]$ returns 1 if x is true or 0 otherwise. Note that since $\mathsf{RowScore}$ returns a value between 0 and 1, $\mathsf{LinkTotal}(\cdot)$ returns n in the best case.

Then, we represent the coverage and saliency of $\langle r, v\rangle$ by normalising the value $\mathsf{LinkTotal}(r, v)$ with respect to the table and the KG:

$$\mathsf{Cover}(r, v) = \frac{\mathsf{LinkTotal}(r, v)}{\sum_{i=1}^{n}[\langle r, v\rangle \in \cup_{e \in \mathsf{Cand}(\rho_i)} \mathsf{Links}(e)]} \tag{9}$$

$$\mathsf{Salience}(r, v) = \frac{\mathsf{LinkTotal}(r, v)}{|\{e \in \mathcal{E} \mid \langle r, v\rangle \in \mathsf{Links}(e)\}|} \tag{10}$$

Intuitively, $\mathsf{Cover}(\cdot)$ computes the popularity of $\langle r, v\rangle$ among the rows of the table, while $\mathsf{Salience}(\cdot)$ considers all entities in $\mathcal{K}$. We combine them as

$$\mathsf{LinkScore}(r, v) = \mathsf{Cover}(r, v) \times \mathsf{Salience}(r, v) \tag{11}$$

so that we can rank the attributes depending both on their coverage within the table and popularity in the KG. This combination allows us to give low ranks to attributes, like $\langle$`isA,Resource`$\rangle$, which should not be considered despite their high coverage since they are not informative. In contrast, it can boost up the score of attributes with a medium coverage in case they have a high saliency.

Finally, we use the scores from Eq. 11 to compute a similarity score between pairs of entities. We compute the similarity between entities e_1 and e_2 as

$$\mathsf{EntitySimilarity}(e_1, e_2) = \sum_{\langle r, v\rangle \in \mathsf{Links}(e_1) \cap \mathsf{Links}(e_2)} \mathsf{LinkScore}(r, v) \tag{12}$$

3.4 Step 4: Disambiguation

Now, we compute which are the row-entity assignments which maximise the coherence in the table, i.e., maximise the similarity between the entities. These assignments are determined using Loopy Belief Propagation (LBP) [21].

We model each row-entity prediction as a categorical random variable, for which the label score $\mathsf{RowScore}(\rho, e)$ is the prior distribution (Fig. 1d1). For convenience, we can view these scores as a sparse matrix $\boldsymbol{L}$ of size $n \times |\mathcal{E}|$. The variables are connected to each other with the edge potentials being defined by entity-entity similarities $\mathsf{EntitySimilarity}(e_1, e_2)$ (Fig. 1d2; equivalently represented by a matrix $\boldsymbol{S}$), which forms a complete graph. Since this graph has loops it is not possible to perform exact inference. Therefore we approximate it by executing LBP. Additionally, all our edge potentials are identical. This causes all nodes to receive identical information from each other. Instead of having separate messages for each node, we thus have a single vector-valued message that provides the belief updates for our nodes:

$$q_e = \prod_{\rho=0}^{n} \sum_{e' \in \mathsf{Cand}(\rho)} L_{\rho,e'} \times S_{e,e'} = \prod_{\rho=0}^{n} (\boldsymbol{LS})_{\rho,e} \tag{13}$$

$$C_{\rho,e} = L_{\rho,e} \times q_e \tag{14}$$

where q_e indicates how similar entity e is to all weighted candidates of all rows, and $C_{\rho,e}$ is the coherence score of entity e for row ρ (Figs. 1e and 1f respectively). Because the main operation consists of a single matrix multiplication, computation is fast and can be parallelized by standard matrix processing libraries.

LBP can be run for multiple iterations (in our case, replacing $L_{\rho,e'}$ by $C_{\rho,e'}$), but is not guaranteed to converge [21]. In fact, we observed that sometimes an excessive number of iterations led to suboptimal assignments. This occurred when the entity similarity scores (Eq. 12) were not accurate due to missing attributes in the KG and ended up "overriding" the more accurate priors that were computed considering only label similarities (Eq. 7) when they are combined in the following step. From our experimental analysis, we observed that in the overwhelming majority of the cases a single iteration of LBP was enough to converge. Therefore, we apply Eq. 14 only once without further iterations.

As we can see from Eq. 14, the selection of the entity for row ρ relies on two components, $\boldsymbol{L}$ and $\boldsymbol{q}$: The first takes into account to what extent the entity label matches the label of candidate entities and to what extent the labels of the attributes matches with the remaining cell values. The second considers the coherence, i.e., the mappings that maximise the similarity between the entities.

Finally, we disambiguate rows by choosing the highest-rated candidate $\hat{e}_\rho = \text{argmax}_e\ C_{\rho,e}$. Then, we re-calculate $\mathsf{ColScore}(c, r)$ with the updated set of candidates containing only the predicted entity $\mathsf{Cand}(\rho) = \{\hat{e}_\rho\}$ and disambiguate columns by choosing the highest scoring relation $\hat{r}_c = \text{argmax}_r\ \mathsf{ColScore}(c, r)$. After this last step is computed, our procedure has selected one entity per row and one relation per attribute column. In the next section, we discuss how we can extract triples from the table.

4 Slot-Filling

After the table is interpreted, we can extract partial triples of the form $\langle s, r, ?\rangle$ where s are the entities mapped to rows and r are the relations associated to columns. If the cell contains numbers or other datatypes (e.g., dates) that we can add the cell value to the KG as-is, but this is inappropriate if the content of the cell refers to an entity. In this case, we need to map the content of the cell to an entity in the KG.

The disambiguation of the content of a cell could be done by querying our label index precisely the same way as done in Sec. 3.1. However, this extraction is suboptimal since now we have available some context, i.e., $\langle s, r, ? \rangle$ that we can leverage to refine our search space. To this end, we can exploit techniques for predicting the likelihood of triples given the KG's structure, namely KG embeddings provided by the TransE algorithm [3]. Given in input e_i, i.e., the entity associated to row i and r_j, i.e., the relation associated to column j, our goal is to extract a fact of the form $\langle e_i, r_j, x \rangle$ where entity x is unknown. We proceed as follows:

1. We query the label index with the content of $\mathsf{Cell}(i, j)$ as done for the computation of $Cand(\cdot)$ in Sec. 3.1. This computation returns a list of entity candidates $\langle e_1, \ldots, e_n \rangle$ ranked based on label string similarities.
2. For each candidate $e_k \in \langle e_1, \ldots, e_n \rangle$, we compute $\mathsf{Rank}(k) = d(\boldsymbol{e_i} + \boldsymbol{r_j}, \boldsymbol{e_k})$ where d is the distance measure used to compute the TransE embeddings (we use the L_1 norm), and $\boldsymbol{e_i}, \boldsymbol{r_j}, \boldsymbol{e_k}$ are the TransE vectors of e_k, r_j, e_i respectively.
3. We return $\langle e_i, r_j, e_k \rangle$ where e_k is the entity with the lowest $\mathsf{Rank}(k)$, i.e, has the closest distance hence it is the triple with the highest likelihood score.

5 Evaluation

We implemented our method into a system called `TAKCO` (TAble-driven KG COmpleter). The code is available online[3].

Our implementation uses two additional systems: *Trident*[4], an in-house triple store to query the KG; and *Elasticsearch 6.4.2*, a well-known system used for building and querying the label index. Moreover, we reimplemented the TransE algorithm for creating the KG embeddings. Since our KGs contain millions of nodes and edges, we parallelized the learning using Hogwild! [24] (we empirically verified that this form of parallelism does not affect the quality of the embeddings).

Baselines. Since our goal is to extract novel facts from tables, we considered existing systems that perform slot-filling as baselines. In particular, we considered the systems T2K MATCH [26] and TABLEMINER+ [33] because of their state-of-the-art results. There are other systems that implement only parts of the pipeline, for instance entity disambiguation (see Section 6 for an overview). An important system in this category is TabEL [2], which exploits co-occurrences of anchor links to entity candidates on Wikipedia pages for predicting a coherent set of entities. Although such system can potentially return better performance

[3] `https://github.com/karmaresearch/takco`
[4] `https://github.com/karmaresearch/trident`

on entity disambiguation, we did not include it in our analysis due its reliance on additional inputs. A comparison between the performance of our method for the subtask of entity disambiguation, and more specialized frameworks like TabEL should be seen as future work.

The system T2K Match implements a series of matching steps that match table rows to entities, using similarities between entity property values and the table columns. The TableMiner+ system consists of two phases that are alternated until a certain confidence level has been reached. Note that these approaches primarily focus on table interpretation. In contrast, we provide an end-to-end system which considers also the operation of slot-filling.

The first system is designed to work with a specific subselection of DBpedia [1] while the second system was originally built to use the Freebase API. We have performed some slight modifications to their source code so that we could perform a fair comparison. For T2K Match, we modified the system to be able to use an augmented set of candidates so that in some experiments we could measure precisely the performance of table interpretation. For TableMiner+, we modified the system so that we could use different KGs without API access.

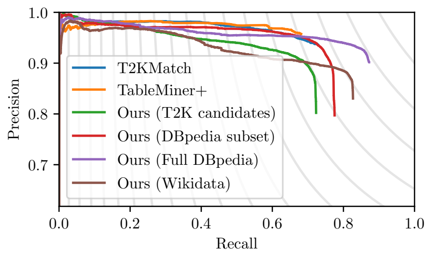
Knowledge Graphs. Our method can work with any arbitrary KG. We consider DBpedia (so that we could compare against T2K Match) which is a popular KGs created from Wikipedia and other sources. We use two versions of DBpedia: The first is the triple-based version of the tabular subset used by T2K Match. This is a subset of DBpedia from 2014 and we consider it so that we can perform an exact comparison. It contains 3.4M entities and 28M facts. Additionally, we also use the latest version of the full KG (version 2016-10). The full DBpedia contains 15M entities (including entities without labels and redirected entities) and 110M facts. Finally, we compare our performance using Wikidata ("truthy" RDF export, acquired on Oct 2018), which has 106M entities and 1B facts. For evaluation, we map the gold standard to Wikidata using `owl:sameAs` links from DBpedia.

Testsets. To the best of our knowledge, there are two openly available datasets of tables that have been annotated for the purpose of table interpretation. The first one is the *T2D* dataset [26], which contains a subset of the WDC Web Tables Corpus – a set of tables extracted from the CommonCrawl web scrape[5]. We use the latest available version of this dataset (v2, released 2017/02). In our experiments, we disregarded tables without any annotation. The resulting dataset contains 238 entity tables with 659 column annotations and 26106 row-entity annotations. Throughout, we refer to this dataset as *T2D-v2*.

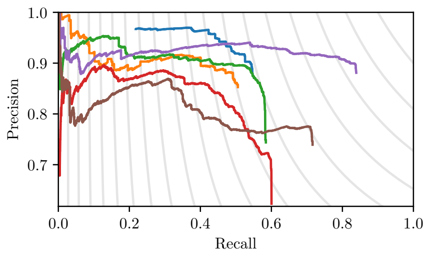
The second dataset is *Webaroo*, proposed by [16]. Tables in this dataset were annotated with entities and relations in YAGO. While these tables are a less varied sample of the ones in the T2D, they allow us to study the behaviour of the systems on a dataset with different annotations. This dataset contains 429 entity tables with 389 and 4447 column and row-entity annotations respectively. In order to test the performance of T2K Match with this dataset, we "ported"

[5] `http://webdatacommons.org/webtables/`

the YAGO annotations to DBpedia using the Wikipedia links they refer to. Finally, we tested the scalability of our system by running it on a large set of Wikipedia tables [2]. Instructions to obtain these datasets are available in the code repository of our system.



(a) Performance tradeoff, T2D-v2



(b) Performance tradeoff, Webaroo

System	Pr.	Re.	F_1
T2KMatch	.94	.73	.82
TableMiner+	**.96**	.68	.80
Ours (T2K candidates)	.88	.72	.79
Ours (DBpedia subset)	.90	.76	.83
Ours (Full DBpedia)	.92	**.86**	**.89**
Ours (Wikidata)	.87	.82	.84

(c) Row-entity evaluation, T2D-v2

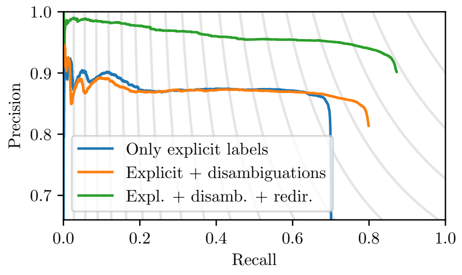
System	Pr.	Re.	F_1
T2KMatch	.88	.55	.67
TableMiner+	.85	.51	.63
Ours (T2K candidates)	.74	.58	.65
Ours (DBpedia subset)	.72	.59	.65
Ours (Full DBpedia)	**.88**	**.84**	**.86**
Ours (Wikidata)	.77	.71	.74

(d) Row-entity evaluation, Webaroo

Fig. 2: Row-entity evaluation scores and precision-recall tradeoff for the T2D-v2 and Webaroo datasets (the isolines of constant F_1 score are shown in grey). Precision, recall, and F_1 are calculated at the threshold of maximum F_1.

5.1 Table Interpretation

We evaluate the performance of determining the correct row-entity assignments, which are the key output for table interpretation. Fig. 2b,d and Fig. 2a,c report a comparison of the performance of our method against the baselines. We measure the precision/recall tradeoff (obtained by altering the threshold value for accepting mappings), and precision, recall, and F_1 (shown at the threshold of maximum F_1) on all predictions. The precision decreases whenever a system makes a wrong prediction while the recall is affected when no entity is selected. Not predicting a match for a row can have several causes: the candidate set for that row might have been empty, the annotated entity might not have been in the KG (this occurs when we use a subset), or when all candidates have been pruned away during the interpretation (this occurs with the baselines).



System	Pr.	Re.	F_1
Only explicit labels	.85	.69	.76
Explicit + disambiguations	.84	.79	.81
Expl. + disamb. + redir.	**.92**	**.86**	**.89**

Fig. 3: Row-entity evaluation scores and precision-recall tradeoff of our approach given different label sources, on T2D-v2.

In these experiments, we configured our system with three different settings: First, we use the same KG and the candidates (i.e., the output of $\mathsf{Cand}(\cdot)$) used by the other two systems. We refer to this setting as "T2K candidates". Then, we use the KG subset used by T2K Match in our own label index and disambiguation ("DBpedia subset"). Finally, we use our own candidate set generation and full KG ("Full DBpedia"). By evaluating the performance of our method with these settings, we can compare the performance of our approach given the limitations of the inputs that the other systems face.

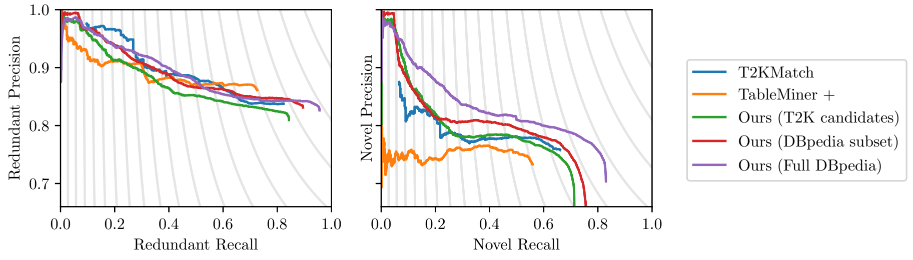
From the results reported in the figures, we can make a few considerations. First, our method returns a comparable recall but an inferior precision than the baselines if we use the set of candidates from T2K Match, but is able to match its performance in terms of F_1 when using the same KG. However, the baselines are limited with respect to KGs. In fact, T2K Match requires that DBpedia is translated into a tabular format while our method does not have this restriction. If our method is configured to use the full DBpedia KG, then it returns the highest recall with only a small loss in terms of precision. This translates in a significantly higher F_1 score than the best of the baselines. These are positive results since a high recall is important for extracting novel facts.

While the precision of our system is low in the limited-input setting, many of the errors that it makes are due to problems with the candidate set and the KG. Therefore, we evaluated a scenario (not shown in the figures of this paper) in which we artificially expanded the candidate set to always include the gold standard. This means we are artificially making use of a "perfect" candidate index. Even with this addition, T2K Match is unable to use these candidates for prediction and returns the same results. In contrast, manually adding them to our system leads to both a notably higher recall and precision.

This indicates that our method is sensitive to the candidate generation, i.e., to the very first selection of candidates using the index label. To evaluate how our system behaves with richer label indices, we evaluated our method on T2D-v2 with three different label indices. The first index only uses the explicit labels of the entities. The second one includes also the labels that we obtain from redirect pages in Wikipedia. The third one adds also the labels we obtain from the disambiguation pages. The results of this experiment are reported in Fig. 3. As we can see from these results, including more labels per entity significantly improves both the precision and recall of our system.

System	Redundant			Novel		
	Pr.	Re.	F_1	Pr.	Re.	F_1
T2KMatch	.84	.82	.83	**.76**	.66	.71
TableMiner+	**.86**	.73	.79	.73	.56	.63
Ours (T2K candidates)	.81	.84	.83	.61	.71	.66
Ours (DBpedia subset)	.83	.90	.86	.59	.76	.66
Ours (Full DBpedia)	.83	**.96**	**.89**	.70	**.83**	**.76**

(a) The scores for extracting novel and redundant triples from T2D-v2, measured at the acceptance threshold of maximum F_1.



(b) The precision-recall tradeoff curve on T2D-v2.

Fig. 4: The novel and redundant precision-recall tradeoff for the T2D-v2 dataset (in gray, the isolines of constant F_1 score). Unlike the experiments in the previous figures, here the bias towards extracting known (redundant) facts is made explicit and we focus on finding novel KG facts in web tables.

5.2 Measuring Redundancy

Current systems (e.g., [26,33]) were evaluated against a set of manual annotations, and scored on the individual subtasks of table interpretation. Such evaluation did not consider the novelty of facts that the system has extracted. In other words, no difference was made between predictions of already known facts or new knowledge, but this difference is important in our context. In order to fill this gap, we need to distinguish between these cases when measuring performance.

Given in input a KG $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$, an extraction technique like ours is expected to yield a new set of predicted facts $\mathcal{F}_P$ over $\mathcal{E}$ and $\mathcal{R}$ from an input source like web tables. If we have gold standard table annotations, we can generate another set of facts $\mathcal{F}_G$ and use them for evaluating how many facts in $\mathcal{F}_P$ are correct. Note that both $\mathcal{F}_P$ and $\mathcal{F}_G$ might contain facts that are either in

$\mathcal{F}$ or not. So far, current techniques have been evaluated w.r.t. the set of true positives $\mathcal{F}_G \cap \mathcal{F}_P$ (correctly extracted facts) and false negatives as $\mathcal{F}_G \setminus \mathcal{F}_P$ (valid facts that were missed). These measures do not take the redundancy of the extracted facts into account, while the redundant information exceeds the novel information for benchmark datasets [14].

In Fig. 4, we show the evaluation of the correctness of *novel* and *redundant* facts separately. Crucially, our system significantly outperforms the baselines with respect to the recall of novel facts, which is paramount to KG completion. In the tradeoff curve for novel triples (Fig. 4b), we also outperform the state-of-the-art regarding precision for most threshold values.

5.3 Slot-filling

To test the scalability of our system, we have run it on all 1.6M tables in the Wikitable dataset. The first step concerns detecting entity tables with key columns that contain entity labels. This process returned 786K tables. Then, we proceeded with the retrieval of entity candidates. About 288K tables did not contain any entity in DBpedia, thus were discarded. The table interpretation process was launched on the remaining 498K tables. Our approach is trivially parallelizable, and runs in 391 ms per table on average.

Ranking	Dataset	Prec@1	Prec@3
Only Label Index (TF-IDF score)	Wikitable	0.37	0.42
	T2D-v2	0.24	0.31
Labels + Embeddings (TransE)	Wikitable	**0.61**	**0.72**
	T2D-v2	**0.62**	**0.74**

Table 1: Precision of slot-filling with/out KG embeddings, calculated on redundant extractions.

From these tables, we extracted 2.818.205 unique facts for 1.880.808 unique slots of the form $\langle s, r, ? \rangle$. Of those slots, 823.806 already contain at least one entity o in the KG. However, we do not know whether our extractions are redundant, or t represents a new extraction that should be added to the existing ones in the KG. To determine the novelty, we queried the label index for every extracted fact and discovered that in 307.729 cases the labels were matching. We can assume these extracted facts to be redundant. From these numbers, we conclude that our extraction process has produced about 1.6M extractions for which we have no evidence of redundancy and thus can be considered as novel. A manual analysis over a sample confirmed this conclusion.

Finally, we evaluated the effectiveness of our procedure for re-ranking the extractions using the KG embeddings on the Wikitable and T2D-v2 datasets. To this end, we compare the naïve index-based ranking obtained by simply picking the top result returned from the label index against our strategy or re-ranking considering the distance of the corresponding embeddings (Sec. 4). We chose to measure the precision for the first or top-3 ranked candidates since this is a standard metric used to evaluate the performance of link prediction [20].

Since we need to know the correct entity, we restricted this analysis to the redundant extractions (i.e., the ones already in the KG) and disregarded the novel ones. Tab. 1 reports the results both when we consider only the best result and the top three. We see that that our embedding-based ranking outperforms the index-based ranking in both cases, and predicts the correct entity at the top of the ranking in 61% of the time, compared to 37% for the Wikitable dataset. Moreover, the relatively low results obtained with the index-based ranking strategy indicate that labels are in general not reliable for disambiguating attributes.

6 Related Work

The first system for interpreting web tables was introduced by Limaye et al. [16]. The system uses a probabilistic graphical model that makes supervised predictions based on a large number of features. Subsequent work approached the problem with a task-specific knowledge graph [30,31] and sped up predictions by limiting the feature set [17] or using distributed processing [10]. Others used an entity label prior from hyperlinks on the web [2], and interpreted tables in limited domains [23].

A separate family of table interpretation systems limit themselves to attribute matching. The simplest approaches perform string similarities between the column headers and relation names or cell values and entity labels [8]. When no overlap between the table and KG can be assumed at all, the work at [22] uses supervised models based on features of the column header and cell values. Some approaches focus on matching tables to relation from Open Information Extraction [30,31] or exploit occurrences of cell value pairs in a corpus of text [28,5], and others perform supervised learning [22,9]. While several approaches have been proposed that are limited to entity linking [7,2], the focus of our work is to optimize table interpretation for novel fact extraction.

The systems evaluated in this paper are designed for open-domain table interpretation. In closed-domain settings, assumptions can reduce the redundancy of extractions. For example, the work of [23] models the incompleteness in the domain subset of the KG by estimating class probabilities based on relations between entities, which the limited domain makes tractable. The systems of [31] and [30] use a probabilistic KG created from a web corpus for supporting table search. This type of KG offers many strategies for improving the recall of new knowledge because it allows for an explicit model of low-confidence facts.

Several models use large web text corpora in addition to the information from the KG. The work by Bhagavatula et al. [2] uses the anchor text of hyperlinks on the web to create a prior for instance matching that takes the popularity of entities into account. Additionally, it exploits co-occurrences of anchor links to entity candidates on Wikipedia pages for predicting a coherent set of entities. The work of [28] creates a set of syntactic patterns from the ClueWeb09 text corpus featuring entities from relations in the KG. Given query pairs of entities from tables, the syntactic patterns from text featuring the query pair are matched to the patterns in the set. A probabilistic model then allows for the prediction of relations from the KG. A similar approach is taken by [5], who use a language model instead of extracted syntactic patterns. This approach queries a search engine with the entity pair, and classify the text that occurs between the entity mentions. A separate direction is the matching of numeric columns, either with metrics for numeric distribution similarity [19] or sophisticated ontologies of quantities and statistical models [12].

The survey at [13] discusses approaches and challenges to the slot filling task in the context of textual information extraction. Most systems use distant supervision for textual pattern learning, and some employ cross-slot reasoning to ensure the coherence of multiple extracted values. Recently, work on Universal Schemas by Riedel et al. [25] has allowed the joint factorisation of textual extractions and KB relations and this boosts slot-filling precision.

In the field of data fusion, systems explicitly aim for high recall and use a post-processing filter to improve precision. In [18], the extracted facts are filtered using machine learning models, and in [6] they are filtered using a sophisticated statistical model of the KG. In [27], the system of [26] is used to interpret a large collection of web tables, after which the extracted facts are filtered using several strategies. However, only 2.85% of web tables can be matched, which is attributed to a topical mismatch between the tables and the KG.

7 Conclusion

We investigate the problem of extending KGs using the data found in Web tables. Existing approaches have focused on overall precision and recall of facts extracted from web tables, but it is important for the purpose of KG completion that the extraction process returns as many (correct) *novel* facts as possible.

We developed and evaluated a new table interpretation method to counter this problem. Our method uses a flexible similarity criterion for the disambiguation of entity-row matches, and employs a PGM to compute new likelihood scores depending on how the various candidates are similar to each other to maximise the coherence of assignments. Because it combines the syntactic match between the tables and the KG with the coherence of the entity predictions, it can confidently predict more candidates for which the attributes in the table are not yet in the KG. Consequently, it extracts more novel facts for KG completion. For the task of slot-filling, we introduced a novel approach for attribute disambiguation based on KG embeddings, which outperforms a naive label-based approach.

We compared our method to two state-of-the art systems, and performed an extensive comparative evaluation on multiple knowledge bases. Our evaluation shows that our system achieves a higher recall during the interpretation process, which is necessary to extract novel information. Furthermore, it is able to extract more (correct) facts that are not yet in the KG.

Interesting directions for future work include the development of extensions for tables where the entity is identified by multiple columns or where rows do not necessarily describe entities. In particular, the heuristics for determining the key column of the table (and whether such a column is present) would need to be replaced by a model that reliably detects the type of table. Moreover, the inclusion of external sources can be useful to extract more novel information from the table. Finally, despite the remarkable work by different research teams to produce good benchmark datasets, there is still the need for larger and more diverse benchmarks to further challenge the state-of-the-art.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Bhagavatula, C.S., Noraset, T., Downey, D.: TabEL: Entity Linking in Web Tables. In: Proceedings of ISWC. pp. 425–441 (2015)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. In: Proceedings of NIPS. pp. 2787–2795 (2013)
4. Cafarella, M., Halevy, A., Lee, H., Madhavan, J., Yu, C., Wang, D.Z., Wu, E.: Ten years of webtables. Proceedings of VLDB **11**(12), 2140–2149 (2018)
5. Cannaviccio, M., Barbosa, D., Merialdo, P.: Towards Annotating Relational Data on the Web with Language Models. In: Proceedings of WWW. pp. 1307–1316 (2018)
6. Dong, X.L., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge Vault: a Web-scale Approach to Probabilistic Knowledge Fusion. In: Proceedings of KDD. pp. 601–610 (2014)
7. Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., Christophides, V.: Matching Web Tables with Knowledge Base Entities: From Entity Lookups to Entity Embeddings. In: Proceedings of ISWC. pp. 260–277 (2017)
8. Efthymiou, V., Hassanzadeh, O., Sadoghi, M., Rodriguez-Muro, M.: Annotating Web Rables Through Ontology Matching. In: Proceedings of OM at ISWC. pp. 229–230 (2016)
9. Ermilov, I., Ngomo, A.C.N.: TAIPAN: Automatic Property Mapping for Tabular Data. In: Proceedings of EKAW. pp. 163–179 (2016)
10. Hassanzadeh, O., Ward, M.J., Rodriguez-Muro, M., Srinivas, K.: Understanding a Large Corpus of Web Tables Through Matching with Knowledge Bases: an Empirical Study. In: Proceedings of OM at ISWC. pp. 25–34 (2015)
11. Hayes, P.: RDF Semantics. W3C Recommendation. Available at http://www.w3.org/TR/rdf-mt/ (2004)
12. Ibrahim, Y., Riedewald, M., Weikum, G.: Making Sense of Entities and Quantities in Web Tables. Proceedings of CIKM pp. 1703–1712 (2016)
13. Ji, H., Grishman, R.: Knowledge base population: Successful approaches and challenges. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1. pp. 1148–1158. Association for Computational Linguistics (2011)

14. Kruit, B., Boncz, P., Urbani, J.: Extracting New Knowledge from Web Tables: Novelty or Confidence? In: Proceedings of KBCOM (2018)
15. Kruit, B., Boncz, P., Urbani, J.: Extracting Novel Facts from Tables for Knowledge Graph Completion (Extended version). arXiv e-prints arXiv:1907.00083 (2019)
16. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and Searching Web Tables Using Entities, Types and Relationships. PVLDB **3**(1-2), 1338–1347 (2010)
17. Mulwad, V., Finin, T., Joshi, A.: Semantic Message Passing for Generating Linked Data from Tables. In: Proceedings of ISWC. pp. 363–378 (2013)
18. Muñoz, E., Hogan, A., Mileo, A.: Using Linked Data to Mine RDF from Wikipedia's Tables. In: Proceedings of WSDM. pp. 533–542 (2014)
19. Neumaier, S., Umbrich, J., Parreira, J.X., Polleres, A.: Multi-level Semantic Labelling of Numerical Values. In: Proceedings of ISWC. pp. 428–445 (2016)
20. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2016)
21. Pearl, J.: Probabilistic reasoning in intelligent systems - networks of plausible inference. Morgan Kaufmann Publishers Inc. (1989)
22. Pham, M., Alse, S., Knoblock, C.A., Szekely, P.: Semantic Labeling : A Domain-independent Approach. Proceedings of ISWC pp. 446–462 (2016)
23. Ran, C., Shen, W., Wang, J., Zhu, X.: Domain-Specific Knowledge Base Enrichment Using Wikipedia Tables. In: Proceedings of ICDM. pp. 349–358 (2015)
24. Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: Advances in Neural Information Processing Systems. pp. 693–701 (2011)
25. Riedel, S., Yao, L., McCallum, A., Marlin, B.M.: Relation Extraction with Matrix Factorization and Universal Schemas. In: Proceedings of HLT-NAACL (2013)
26. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML tables to DBpedia. In: Proceedings of WIMS. p. 10 (2015)
27. Ritze, D., Lehmberg, O., Oulabi, Y., Bizer, C.: Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In: Proceedings of WWW. pp. 251–261 (2016)
28. Sekhavat, Y.A., Paolo, F.D., Barbosa, D., Merialdo, P.: Knowledge Base Augmentation using Tabular Data. In: Proceedings of LDOW at WWW (2014)
29. Sun, H., Ma, H., He, X., Yih, W.T., Su, Y., Yan, X.: Table Cell Search for Question Answering. In: Proceedings of WWW. pp. 771–782 (2016)
30. Venetis, P., Halevy, A., Madhavan, J., Paca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering Semantics of Tables on the Web. PVLDB **4**, 528–538 (2011)
31. Wang, J., Shao, B., Wang, H.: Understanding Tables on the Web. In: ER. vol. 1, pp. 141–155 (2010)
32. Yakout, M., Ganjam, K., Chakrabarti, K., Chaudhuri, S.: InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. In: Proceedings of SIGMOD. pp. 97–108 (2012)
33. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Semantic Web **8**(6), 921–957 (2017)