TAIPAN: Automatic Property Mapping for Tabular Data

Ivan Ermilov and Axel-Cyrille Ngonga Ngomo

University of Leipzig, Institute of Computer Science, Leipzig, Germany
iermilov,ngonga@informatik.uni-leipzig.de
AKSW Research Group: http://aksw.org/

Abstract. The Web encompasses a significant amount of knowledge hidden in entity-attributes tables. Bridging the gap between these tables and the Web of Data thus has the potential to facilitate a large number of applications, including the augmentation of knowledge bases from tables, the search for related tables and the completion of tables using knowledge bases. Computing such bridges is impeded by the poor accuracy of automatic property mapping, the lack of approaches for the discovery of subject columns and the mere size of table corpora. We propose TAIPAN, a novel approach for recovering the semantics of tables. Our approach begins by identifying subject columns using a combination of structural and semantic features. It then maps binary relations inside a table to predicates from a given knowledge base. Therewith, our solution supports both the tasks of table expansion and knowledge base augmentation. We evaluate our approach on a table dataset generated from real RDF data and a manually curated version of the T2D gold standard. Our results suggest that we outperform the state of the art by up to 85% F-measure.

Keywords: Web tables, knowledge base augmentation, table expansion

1 Introduction

The Linked Data Web has developed from a mere idea to a set of more than 85 billion facts distributed across more than 10,000 knowledge bases[1] over less than 10 years. However, the Document Web is also growing exponentially, with a large proportion of the information contained therein not being available on the Data Web. Consequently, the gap between the Data Web and the Document Web keeps on growing with the addition of novel knowledge in either portion of the Web. Devising ways to bridge between the Document Web and the Linked Data Web has been the purpose of a number of works from different domains. The unstructured data on the Web is being transformed to RDF by means of a combination of named entity recognition (see, e.g., [5,14,19]), entity linking (see, e.g., [2,22]) and relation extraction (see, e.g., [15,6]) approaches. However, such approaches can only deal with well-formed sentences and do not address other structures that are commonly found on the Document Web, in particular, tables. While a few approaches for disambiguating entities in tables have been developed in the past [27,23,24,1,25], porting the content of tables to RDF has been the subject of a

[1] http://lodstats.aksw.org

limited number of approaches [11,13,16]. These approaches are however limited in the structure of the tables they can handle. For example, they partly rely on heuristics such as using the first non-numeric column of a table as subject for the triples that are to be extracted [10].

We present TAIPAN, a generic approach towards extracting RDF triples from tables. Given a table and a reference knowledge base, TAIPAN aims to (1) identify the column that contains the subject of the triples to extract, i.e., the *subject column*. To this end, our approach relies on maximizing the likelihood that the elements of a column (1) all belong to the same class and, (2) once disambiguated, will actually have property values that correspond to the properties found in the table; (2) detect properties that correspond to the columns of the tables. Here, TAIPAN maximizes the probability that the columns of the table will yield property values for the same property given the assumed assignment of the subject column; (3) facilitate the disambiguation and extraction of RDF from tables. Hence, the results of TAIPAN can be used to feed any entity disambiguation system for tables.

The rest of this paper is structured as follows: in section 2 we describe our conceptual framework. Then, we employ this framework to define the problem tackled by TAIPAN formally (see section 3). Thereafter, we use the same notation to explain our approach (see section 4). We clarify implementation details in section 5. In section 6, we evaluate our approach on a manually curated portion of the T2D benchmark (which we dub T2D*) against the approaches proposed in [24] and [16,17]. In particular, we measure the accuracy of our subject column identification approach as well as the F-measure achieved by our property mapping approach. Section 7 gives an overview of related work and section 8 concludes the paper.

2 Preliminary Definitions

In this section, we introduce the notation and definitions required to formalize the subject column identification and property mapping problems.

2.1 Tabular Data Model

For modeling tabular data we extend the canonical table model described in [4]. Essentially, the canonical table model distinguishes between the *header* of a table and the *data* of the same table (see Figure 1). A table is represented as a tuple, where the header is a vector and the data is a matrix.

Definition 1. *A* table $T = (H, D)$ *is a tuple consisting of a* header H *and* data D*, where:*

- *the header* $H = (h_1, h_2, \ldots, h_n)$ *is a vector of size* n *which contains* header elements h_i.
- *the data* $D = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix}$ *is a* (m, n)*-matrix consisting of* n *columns and* m *rows.*

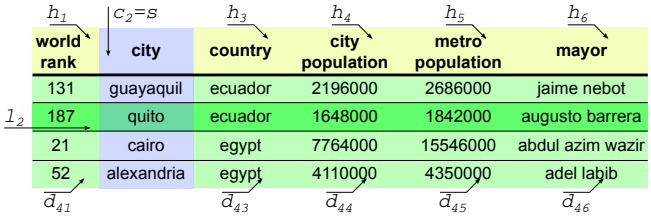

Fig. 1. An example of a table from T2D gold standard with semantics from our table model.

Consequently, we introduce the concept of table projections, where the data of a table is represented as a one-dimensional vector of value vectors.

Definition 2. *The* column projection *of a* table $T = (H, D)$ *is a table* $col(T) = (H, col(D))$ *, where* $col(D) = (c_1, c_2, \ldots, c_n)$*, with* $c_n = (d_{1,n}, d_{2,n}, \ldots, d_{m,n})$*. Similarly, the row projection of a table* $row(T) = (H, row(D))$ *where* $row(D) = (l_1, l_2, \ldots, l_m)$*, with* $l_m = (d_{m,1}, d_{m,2}, \ldots, d_{m,n})$

Hereafter, we will commonly work with the row projections of tables.

Informally, the *subject column* of a table T is a column that contains labels of resources that instantiate the main subject of a table. For instance, in a table taken from the T2D reference dataset [16] with the header H = `(world rank, city, country, city population, metro population, mayor)` (see Figure 1), the main subject is `city`. Consequently, the second column is the subject column. In general, we assume that the subject column is to be connected to every other column in the reference table via a binary relation. Hence, we adopt the following functional definition:

Definition 3. *The subject column* s *is a column which divides table* T *into* $(n-1)$ *two-column tables (which we dub* ***atomic tables****), where the binary relation* ρ_i *between* s *and each of the other columns* c_i *in* T *corresponds to a property in a reference knowledge base* K *(e.g., see Figure 2).*

Following the definition 3, we define an atomic table as follows:

Definition 4. *An atomic table is a table* $T'_i = (H'_i, D'_i)$ *such as* $H'_n = (h_s, h_i)$ *and* $col(D'_i) = (s, c_i)$*, where* h_s *is a header item of the subject column and* s *is a subject column.*

For example, in Figure 2, for the left-most atomic table $T'_1 = (H'_1, D'_1)$, the header is $H'_1 = (city,\ world\ rank)$. The column projection consists of subject column and the first column of the source table: $col(D'_1) = (s, c_1)$, where s = `(guayaquil, quito, cairo, alexandria)` and $c_1 = (131,\ 187,\ 21,\ 51)$.

2.2 Knowledge Base Model

We now introduce the knowledge base model (derived from [4]) underlying our work. Let $\mathcal{U}$ be the set of all URIs, $\mathcal{B}$ be the set of all blank nodes, $\mathcal{L}$ be the set of all literals and Γ be the set of all *RDF terms* with $\Gamma = \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$. Furthermore, we make use of the following notions:

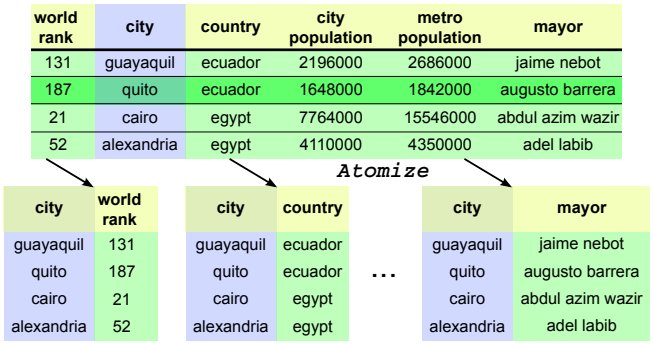

Fig. 2. Example of a table atomization.

- $\mathcal{S}$ is the set of RDF subjects with $\mathcal{S} \subseteq \mathcal{U} \cup \mathcal{B}$,
- $\mathcal{R}$ is the set of RDF properties (relations) with $\mathcal{R} \subseteq \mathcal{U}$,
- $\mathcal{O}$ is the set of RDF objects, with $\mathcal{O} \subseteq \mathit{\Gamma}$,
- $\mathit{\Pi}$ is the set of all *triples*, defined as $\mathit{\Pi} \subseteq \mathcal{S} \times \mathcal{R} \times \mathcal{O}$,
- $\mathbb{E}$ is the set of all *entities*, and
- $\mathcal{C}$ is the set of all classes, that is the subset of $\mathcal{U}$, which describes the classes of the entities $\mathbb{E}$ in $\mathit{\Pi}$.

Our basic assumption is that a binary relation between columns of a table can correspond to a property inside a knowledge base.

3 Problem Statement

TAIPAN aims to expose the semantics of tabular data. To this end, we address the following two subproblems.

3.1 Problem 1: Subject Column Identification

The problem of subject column identification can be formalized using previously introduced concepts as follows.

Problem 1. Given a table $col(T) = (H, col(D))$, where $col(D) = (c_1, c_2, \ldots, c_n)$, find a column c_i such that c_i satisfies definition 3, i.e., such that $col(T)$ can be split into atomic tables which express the extension of a property $r \in \mathcal{R}$ or the inverse r^{-1} of such a property.

The subject column identification is an important preprocessing step, which has to be performed with the highest precision possible. Failing to identify subject column will lead to erroneous atomic tables and thus to less information being ported from T to the reference knowledge K. For example, for a correctly identified subject column $c_i = s$ dubbed `city` (see Figure 1), the binary relation ρ_i between "cairo" and "abdul azim wazir" (i.e. $\rho_i(\text{"}cairo\text{"}, \text{"}abdul\ azim\ wazir\text{"})$) can be mapped to a knowledge base such as DBpedia, where ρ_i corresponds to `dbo:mayor` property. Another important consequence of subject column identification is the possibility to decompose table into atomic tables.

3.2 Problem 2: Property Mapping

The property mapping of a table can be defined as a function λ, such as for each binary relation $\rho_i : s \rightarrow c_i$ between the subject column s and every other column of a table, it assigns a property inside a knowledge base. Therefore, for each ρ_i we have to find a mapping to a particular $r \in \mathcal{R}$. We denoted this mapping by λ and write $\lambda(\rho_i) = r$.

As table semantics are ambiguous, we cannot determine the definite correspondence between a binary relation in a table and a property inside a knowledge base. Moreover, a single binary relation can be mapped to several properties. However, relational tables are likely to have functional binary dependencies, which are mapped to particular functional properties inside a knowledge base. Therefore, given a single binary relation between columns and for each property $r \in \mathcal{R}$, we can define the probability of r being the correct binary relation ρ_i. We denote this probability $P(\lambda(\rho_i) = r)$. The problem at hand can now be reduce to finding the best mapping function λ, i.e., the λ that maximizes $P(\lambda(\rho_i) = r)$ for all ρ_i.

Problem 2. Given a table $col(T) = (H, col(D))$, where $col(D) = (c_1, c_2, \ldots, c_n)$ and $c_k = s$, find a mapping function λ, which maximizes the probability of having mapped each $\rho_i : s \rightarrow c_i$ to the correct $r_j \in \mathcal{R}$.

Note that by these means, we reduce the two tasks to the same core problem formulation. In the following, we will use this formulation to derive approaches for addressing the two problems at hand.

4 Approach

In this section we describe our solutions to the subject column identification and property mapping problems.

4.1 Subject Column Identification

To support column identification we extend an idea from distant supervision learning [18,12]. Essentially, we boil down the column identification to finding the column c_i in a table that has the most relations to other columns inside the same table. To find such a column, we begin by selecting m' rows of the given table T. Then, for each row, we disambiguate cell values against entities from a given reference knowledge base. Finally, we apply four triple patterns to find potential relations between each combination of columns. The approach derives two important features for each column: *support* and *connectivity*.

Definition 5. *The* support St_i *of a column* c_i *in a table* T *is the ratio between cells with disambiguated entities inside and total number of cells for a column.* $St_i = \frac{\sum_{j=1}^{|row(D)|} e_j}{|row(D)|}$, *where*

$$e_j = \begin{cases} 1, & \textit{if } d_{ij} \textit{ could be disambiguated to some } e \in \mathbb{E} \\ 0, & \textit{otherwise} \end{cases} \quad (1)$$

Definition 6. *The* connectivity C_i *of a column* c_i *in a table* T *is the ratio between number of connections (i.e., properties) of the column to other columns inside the same table to the total number of columns.*

In our implementation, we evaluated the *support* of a particular column by using *AGDISTIS* [21] to disambiguate the entries d_{ij} (disambiguateEntities on line 4 in Algorithm 1) and used DBpedia as reference knowledge base. For example, given the table in Figure 1, the entry d_{22} = `quito` was disambiguated as `http://dbpedia.org/resource/Quito`. All entities in the columns c_2, c_3 and c_6 of the example table could be disambiguated. Hence, their support is $\frac{4}{4} = 1$. In contrast, all numerical columns have support of 0. Our approach towards computing the support of all columns in a table is shown in Algorithm 1.

Algorithm 1: TAIPAN Column Support Evaluation. Runs in $\mathcal{O}(m'n)$ time.

Data: Table T of size (m, n), m'
Result: St - support vector for the table columns, Et - entity matrix

```
1  Instantiate St, Et;
2  for row = 1 to m' do
3      for col = 1 to n do
4          Et[row][col] ⟵ disambiguateEntities(T[row][col]);
5          if |Et[row][col]| > 0 then
6              St[col] ← St[col] + 1
7          end
8      end
9  end
10 for col = 1 to n do
11     St[col] = St[col]/m' · 100%
12 end
13 return St, Et
```

After the disambiguation, we now employ a set of triple patterns to find potential properties in a knowledge base as follows.

```
<%value> ?property <%value>
```

Listing 1.1. Entity-Entity Triple Pattern (1)

```
<%value> ?property "%value"@en
```

Listing 1.2. Entity-Literal Triple Pattern (2a)

```
<%value> ?property ?o .
FILTER regex(?o, ".*%value.*", "i")
```

Listing 1.3. Regex Entity-Literal Triple Pattern (2b)

These patterns are a heuristic mean to determine the set of potential properties between pairs of columns. To this end, we combine the results of the disambiguation step

with the original cell values (for entries that could not be disambiguated). Correspondingly, `%value` is instantiated by using either the disambiguated entity from a column value (patterns 1 and 2a-b) or a column value itself (patterns 2a-b). For instance, to find relations between *city* and *city population* in our example, given that $quito$ was disambiguated and 1648000 not, the triple patterns (2a-b) are used. In this case triple pattern (2b) will be instantianed as follows.

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
dbpedia:Quito ?property ?o .
FILTER regex(?o, ".*1648000.*", "i")
```

Listing 1.4. Example of TP (2b) with instantiated variables

The retrieved properties from triple patterns are stored in a *connectivity tensor* of order 3 and of dimensions $m' \times n \times n$ (m' is the sample size for rows and stands for the number of rows used in the Algorithm 1 as disambiguated entities are used in the triple patterns). Each entry Cn_{ijk} contains the set of properties that were detected by the approach above for the pair of column entries d_{ij} and d_{ik}. The connectivity C_j of a column c_j can be inferred from Cn as follows:

$$C_j = \frac{\sum_{i=1}^{|row(D)|} \frac{\sum_{k=1}^{|col(D)|} |Cn_{ijk}|}{|col(D)|}}{|row(D)|}. \tag{2}$$

The evaluation of *connectivity tensor* is shown in Algorithm 2.

For example, the connectivity of column `country` of our running example (see Figure 1) can be evaluated as: $C_3 = \frac{\sum_{i=1}^{4} \frac{\sum_{k=1}^{6} |Cn_{i3k}|}{6}}{4}$.

$$Cn_{i3k} = \begin{pmatrix} \emptyset & country & \emptyset & populationTotal & \emptyset & citizen, official \\ \emptyset & country & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & populationTotal & citizen, official \\ \emptyset & country & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix} \tag{3}$$

Given Cn_{i3k} as in Equation 3, the connectivity evaluates to $C_3 = 0.375$.

After characterizing columns by means of their support and connectivity scores, we can use binary classifiers to classify columns of a table as being either subject columns or not. Binary classifiers used in the experiments as well as discussion on their performance are described in section 6.2.

4.2 Property Mapping

In this section we describe our approach to find an adequate mapping function λ. Our approach assumes that a subject column has already been identified. As a first step, we take the header $H = (h_1, h_2, \ldots, h_n)$ of the input table T and for each element h_i retrieve seed properties from a reference set of potential properties. Then, the set of seed properties is ranked according to the property frequency inside the reference knowledge base K.

Algorithm 2: TAIPAN Column Connectivity Tensor Evaluation. Runs in $\mathcal{O}(mn^2)$ time.

Data: Table T of size (m, n), entity matrix Et, m'
Result: Cn, connectivity matrix for table T

```
Instantiate Cn;
for row = 1 to m' do
    for col = 1 to n do
        for otherCol = col + 1 to n do
            Cn[row][col][otherCol], Cn[row][otherCol][col] ⟵
              findRelation(T[row][col], T[row][otherCol], Et)
        end
    end
end
return Cn
```

Given an identified subject column, a table of size (m, n) is atomized into $(n-1)$ two-column tables $T'_i = (H'_i, D'_i)$. Each atomic table represents exactly one binary relation ρ_i, which should have a correspondence to a property $r_j \in \mathcal{R}$ inside a knowledge base. For example, table shown in Figure 1 is decomposed as shown in Figure 2.

While connectivity performs well to identify subject column of a table, the connectivity tensor (i.e. properties found by triple patterns) does not contain the target properties from a knowledge base. Therefore, for each element h_i we retrieve seed properties in addition to properties extracted via triple patterns. To retrieve seed properties from a knowledge base, we perform a look up on an index created from the values of `rdfs:label` and `rdfs:comment`. This index is queried with the values of the table header such as $h_3 = country$.

To rank the properties, we employ a probabilistic model. The probability of a relation ρ_i for an atomic table $T'_i = (H'_i, D'_i)$ to map to a property r_j is defined as follows:

Definition 7. *A probability of relation ρ_i to correspond to property r_j equals to a number of pairs (s_m, d_{mi}) corresponding to property r_j divided by size of a table:* $P(\lambda(\rho_i) = r_j) = \frac{\sum_{m=1}^{|row(D)|} |(s_m, d_{mi}) \in r_j|}{|row(D)|}$

For example, for the atomic table shown in Figure 2 we would retrieve two properties from DBpedia knowledge base such as: *dbo:country* and *dbo:largestCity*. Let us assume the following knowledge base for the sake of simplicity:

City	**dbo:country**	**dbo:largestCity**
guayaquil	equador	equador
london	UK	UK
cairo	egypt	egypt
alexandria	egypt	N/A

We can calculate probabilities for the properties as: $P(h_3 = dbo : country) = \frac{3}{4}$, $P(h_3 = dbo : largestCity) = \frac{2}{4}$.

The property with the highest probability as defined in definition 7 would be selected, i.e. `dbo:country`.

5 Implementation Details

In the implementation, we use DBpedia as a reference knowledge base. The properties are retrieved from DBpedia with triple patterns as well as from LOV.[2] LOV maintains a reverse index of classes and properties from different ontologies based on `rdfs:label` and `rdfs:comment`. The property ranking is performed as described in section 4.2. For the property lookups LOV returns a score which quantify the relevance of each result. The score is based on TF/IDF and field norms.[3] To improve the precision of TAIPAN, we introduce a score threshold (i.e., we only accept properties which have a score higher than the specified threshold as candidates). As we can see in Figure 3, the best performance is achieved when the threshold is set to 0.8, which the value we use throughout our experiments.

6 Experiments and Results

The goal of our experiments was to measure how well our column identification and our property mapping approaches perform. Hence, we compared the recall and precision achieved by our approach with that of the approaches presented in [24] (subject column identification) and [16] (property mapping). To the best of our knowledge, these are the best performing approaches on these tasks at the moment. The data used in our experiments and the source code of TAIPAN and the annotation interfaces used to curate T2D are available on Github.[4]

6.1 Experimental Setup

Hardware The T2K algorithm [16] requires at least 100 GB RAM to run. Therefore, the experiments for T2K algorithm were performed on a virtual machine running Ubuntu 14.04 with 128 GB RAM and 4 CPU cores. All experiments with TAIPAN were evaluated on an Ubuntu 14.04 machine with 4 cores i7-2720QM CPU and 16 GB RAM.

Gold Standard We aimed to use T2D entity-level Gold Standard (T2D), a reference dataset which consists of 1 748 tables and reflects the actual distribution of the data in the Common Crawl,[5] to evaluate our algorithms. However, the analysis of T2D showed a substantial amount of annotation mistakes such as[6]:

- Tables containing data about `dbo:Plant`, `dbo:Hospital` instances are annotated with the class `owl:Thing`.
- `rdfs:label` is used in an inflationary manner. For example, both first and last name of persons are marked as `rdfs:label`.
- Columns with country names is annotated with `dbo:collectionSize`.

[2] `http://lov.okfn.org/`

[3] `https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html`

[4] `https://github.com/aksw/taipan`

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

[6] For a complete analysis, see `https://github.com/AKSW/TAIPAN-Datasets/tree/master/T2D`

– Columns with active drug ingredients is annotated with `dbo:commonName`.

It is noticeable, that T2D contains 978 tables annotated with owl:Thing class. An analysis of a random sample (50) of the tables from these 978 showed that all of them contain annotation mistakes.

To address T2D annotation problems, we asked expert users to annotate both subject columns and DBpedia properties. For the subject column identification annotation task, we had 15 expert users annotate 322 randomly picked tables from T2D with 2 annotators per table. We discarded the tables where the experts did not agree. As a result, the 116 tables that (1) had no subject column at all (4 tables) and (2) which possessed a subject column upon which the experts agreed (112 tables) were included into our manually curated dataset, which we dub T2D*. To assess the quality of T2D*, we calculated the F-measure achieved by each annotator as proposed in [7]: $F = \frac{2 \cdot 116}{2 \cdot 116 + (322 - 116)} = 0.53$. According to [9], the interval (0.41, 0.60) represents moderate agreement strength. This hints at how difficult the problem at hand really is.

For the property annotation, we involved 12 Semantic Web experts. All experts were experienced DBpedia users or contributors. Each user annotated 20 tables (2 annotators per table). However, to reduce the time per annotation, we also displayed property suggestions from the LOV search engine. On average, each user spent approximately 30 minutes to complete the task. Out of 116 annotated tables, 90 (77.5%) tables had properties upon which the experts agreed. Moreover, the experts agreed on 236 (53.5%) properties for the 441 columns we considered in T2D* (subject columns excluded). Out of 236 annotated properties, the experts identified 104 (44%) properties from DBpedia. The F-measure for the property annotation task is defined as $F = \frac{2 \cdot 236}{2 \cdot 236 + (441 - 236)} = 0.70$. According to [9] (0.61, 0.80) interval represents substantial agreement strength. Note that we shuffled the positions of the columns in the T2D* dataset randomly as in real-life scenarios the subject column can be at any position in a table (in contrast to most tables in T2D). The same holds for the subsequent dataset.

DBpedia Table Dataset (DBD) We also evaluated TAIPAN using a dataset generated directly from DBpedia concise bounded descriptions[7] (CBDs) dubbed **DBD**. We selected 200 random classes with at least 100 CBDs in each class. For each class, we generated 5 tables with 20 rows each (i.e. using 20 CBDs). Inside a table, each row corresponds to a CBD. The subject column was assigned the header `label` and contained the `rdfs:label` of the resource whose CBD was described by the row at hand. The headers of all other columns were values of `rdfs:label` of corresponding properties. The values of the columns are the values of corresponding properties. We selected only direct property/value pairs for CBDs, ignoring blank nodes. The resulting dataset contains 1 000 tables. The implementation of the data generator[8] as well as the DBD[9] are available on Github.

Training and Testing Given that one usually only has a small number of annotated tables to train an extraction approach, we opted to use an inverse 10-fold cross-validation

[7] `https://www.w3.org/Submission/CBD/`

[8] `https://github.com/aksw/TAIPAN-DBD-Datagen`

[9] `https://github.com/AKSW/TAIPAN-Synth-Datagen/tree/master/DBpediaTableDataset/tables`

	Rule-based	Support	Connectivity	Support-Connectivity
T2D*	51.72%	54.31%	36.00%	56.89%
DBD	52.20%	90.80%	80.00%	84.40%

Table 1. Accuracy for subject column identification. Evaluation of support and connectivity features.

to evaluate TAIPAN. This means that each dataset was subdivided into 10 folds of the same size. 10 experiments were then ran, within which one fold was used for training and the 9 other folds for testing.

6.2 Subject Column Identification

According to [24], a simple rule-based approach (pick the left-most column which is not a number or date) for subject column identification achieves 83% accuracy[10], while an SVM with an RBF kernel with the following 5 features increases accuracy up to 94%: (1) fraction of cells with unique content, (2) fraction of cells with numeric content, (3) variance in the number of date tokens in each cell, (4) average number of words in each cell, and (5) column index from the left.

We recreated the experiment on T2D* and DBD. Our experiments (see Table 1) show that for T2D*, the rule-based approach (the baseline) achieves only 51.72% accuracy, while the SVM proposed in [24] achieves 49.52% accuracy in an inverse ten-fold cross-validation. Note that this performance is different from stipulated by the authors on their corpus.[11] On the other hand, selecting the column that achieves the highest support (see Table 1) already performs by 5.17% better than the rule-based baseline. While selecting a column based on connectivity alone performs much worse than baseline, a linear combination of the support and connectivity features $\alpha \cdot St_i + (1 - \alpha) \cdot C_i$ with $\alpha = 0.3$ achieves further gain over the baseline (6.04%).

In an effort to check whether more complex models would lead to even better results, we evaluated TAIPAN feature set with 7 different classifiers (see Table 2).[12] TAIPAN feature set includes all the features proposed by [24] with addition of connectivity and support. For T2D*, the best performing method for TAIPAN was based on SVM. This method achieves 80.74% accuracy in an inverse tenfold cross validation and thus achieves 29.02% gain over the baseline. The further experiments for DBD dataset showed that decision tree classifier performs the best on average for both T2D* and DBD. As a result, we selected decision tree classifier to be default setting for TAIPAN.

6.3 Property Mapping

We evaluated TAIPAN using our T2D* and DBD by comparing it with the state-of-the-art solution for table to knowledge base mapping T2K described in [16,17]. T2K is

[10] Accuracy is defined as a ratio of correctly guessed subject columns to a number of overall guessed subject columns.

[11] We contacted the authors to obtain their corpus but were not provided access to it. Still, we followed the specification of the SVM in their paper exactly.

[12] We used the classifier implementations from scikit-learn python library at `http://scikit-learn.org/`. For more information on the implementation, please refer to the TAIPAN Github repository at `https://github.com/AKSW/TAIPAN`.

	T2D*	DBD
SVM	$(80.74 \pm 9.17)\%$	$(69.64 \pm 19.91)\%$
KNeighbors	$(36.94 \pm 15.17)\%$	$(87.36 \pm 3.37)\%$
SGD	$(34.29 \pm 30.69)\%$	$(39.69 \pm 22.46)\%$
Decision Tree	$\mathbf{(72.59 \pm 15.04)\%}$	$\mathbf{(79.50 \pm 5.76)\%}$
Gradient Boosting	$(75.77 \pm 11.93)\%$	$(67.35 \pm 2.29)\%$
Nearest Centroid	$(51.11 \pm 9.84)\%$	$(59.19 \pm 4.09)\%$
SGD (perceptron loss function)	$(37.25 \pm 27.84)\%$	$(29.63 \pm 19.88)\%$

Table 2. Accuracy for subject column identification. TAIPAN.

	T2D*			DBD		
	Recall	**Precision**	**F-measure**	**Recall**	**Precision**	**F-measure**
TAIPAN	72.12%	39.27%	50.85%	84.31%	86.01%	85.15%
T2K	36.54%	48.72%	41.76%	0.002%	0.002%	0.002%

Table 3. Recall, precision and F-measure of TAIPAN and T2K algorithm.

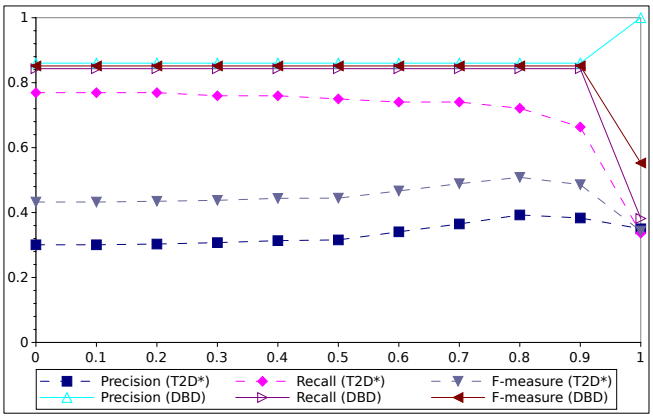

Fig. 3. Recall, precision and F-measure of TAIPAN as a function of a score threshold.

open-source and available online.[13] We do not compare T2K to TAIPAN on T2D due to substantial amount of annotation mistakes in T2D (see section 6.1).

We calculated the recall achieved by the approaches as the number of correctly mapped properties divided by the number of properties in a gold standard. The precision was computed as the number of correctly mapped properties divided by total number of mapped properties.

The results achieved by both approaches are shown in Table 3. For T2D*, T2K has a 9.5% better precision than TAIPAN. However, TAIPAN achieves a 36% better recall, hence outperforming T2K by 9% F-measure. An error analysis of TAIPAN suggests that the 39% precision it achieves can be improved significantly by enhancing the ranking of

[13] http://dws.informatik.uni-mannheim.de/en/research/T2K

properties with heuristics from the whole table corpus and not only using the information available in a single table. For example, given the frequency of the header `Anglican Church` inside the data corpus `Frequency("Anglican Church") = 1`, it is possible that this property is not available in the reference knowledge base.

For DBD, T2K could only match 6 columns correctly, resulting in under 1% F-measure. TAIPAN achieved 85.15% F-measure, significantly outperforming T2K. TAIPAN does not achieve a perfect property mapping because the DBD dataset contains columns homonymous columns from two different namespace, i.e., the ontology and the property namespace (for example, `http://dbpedia.org/property/birthDate` and `http://dbpedia.org/ontology/birthDate`). Overall, our results suggest that TAIPAN outperforms the state of the art significantly in both subject column identification and property mapping.

7 Related Work

In this paper, we focus on the problem of automatic mapping of web tables to ontologies. Semi-automatic and manual approaches, which rely on user input (e.g. [8], [3]) as well as ontology alignment (e.g. [20]) are out of scope of this paper. Research on the topic of web tables is mostly carried out by two communities: Researchers from major search engines and researchers involved in open projects such as Common Crawl[14] and Web Data Commons[15]. A significant portion of the related work on web tables is enlisted on the Web Data Commons web site.[16] In general, WDC identified four different applications in the field of web tables: (1) data search, (2) table extension, (3) knowledge base construction, and (4) table matching. Approaches supporting data search are represented, for instance, by [23,24,1]. The authors describe creation of a `isA` database from webpages via Herst patterns and using it to identify column classes and relations between columns. In a table extension application, a local table is extended with additional columns based on the corpus of tables that are published on the Web.

In the table matching applications [11,13,16,17], most approaches perform three basic steps: (1) column class identification, (2) entity disambiguation and (3) relation extraction. Only recent work by Ritze et. al [16,17] made the T2D gold standard available.

Subject column identification is addressed to a larger extent by [24,26]. Wang et. al [26] propose a naive approach, where the subject column is simply the the first column from the left that satisfies a fixed set of rules. Venetis et. al [24] identify subject column using a SVM with an RBF kernel. However, they do not open-source their code or their data. To the best of our knowledge, we outperform both state of the art approaches w.r.t. the F-measure that we achieve.

8 Conclusions and Future Work

In this paper, we described novel approach for subject column identification and property mapping for web tables. We improved the T2D gold standard by curating it manually

[14] `https://commoncrawl.org/`
[15] `http://webdatacommons.org/`
[16] `http://webdatacommons.org/webtables/`

with the help of 20 Semantic Web experts and used this T2D* to evaluate our approach against the state-of-the-art. While we were able to achieve a recall and an F-measure that were considerably higher than the state-of-the-art, our evaluation also revealed that the precision of TAIPAN can still be improved. The improvements can be achieved by supplementing our property ranking with additional heuristics over the whole table corpus. Moreover, we noticed that a large portion of the columns (56%) in our benchmark contained meaningful information that can be potentially mapped to other knowledge bases. We will thus extend our extraction approach to cover such cases in future work.

9 Acknowledgments

This work has been supported by Eurostars projects DIESEL (project no. 01QE1512C), the BMWI Project GEISER (project no. 01MD16014E) as well as the European Union's H2020 research and innovation action HOBBIT (GA no. 688227).

References

1. S. Balakrishnan, A. Halevy, B. Harb, H. Lee, J. Madhavan, A. Rostamizadeh, W. Shen, K. Wilder, F. Wu, and C. Yu. Applying webtables in practice.
2. D. Carmel, M.-W. Chang, E. Gabrilovich, B.-J. P. Hsu, and K. Wang. Erd'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum*, volume 48, pages 63–77. ACM, 2014.
3. I. Ermilov, S. Auer, and C. Stadler. Csv2rdf: User-driven csv to rdf mass conversion framework. In *Proceedings of the ISEM '13, September 04 - 06 2013, Graz, Austria*, 2013.
4. I. Ermilov, S. Auer, and C. Stadler. User-driven semantic mapping of tabular data. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, pages 105–112, New York, NY, USA, 2013. ACM.
5. O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.
6. D. Gerber and A.-C. N. Ngomo. Extracting multilingual natural-language patterns for rdf predicates. In *Knowledge Engineering and Knowledge Management*, pages 87–96. Springer, 2012.
7. G. Hripcsak and A. S. Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, 2005.
8. C. A. Knoblock, P. Szekely, J. L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan, and P. Mallick. Semi-automatically mapping structured sources into the semantic web. In *Extended Semantic Web Conference*, pages 375–390. Springer, 2012.
9. J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.
10. O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer. The Mannheim Search Join Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2015.
11. G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3(1-2):1338–1347, 2010.

12. M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.
13. V. Mulwad, T. Finin, Z. Syed, and A. Joshi. Using linked data to interpret tables. *COLD*, 665, 2010.
14. D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.
15. N. Nakashole, G. Weikum, and F. Suchanek. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics, 2012.
16. D. Ritze, O. Lehmberg, and C. Bizer. Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, page 10. ACM, 2015.
17. D. Ritze, O. Lehmberg, Y. Oulabi, and C. Bizer. Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *Proceedings of the 25th International Conference on World Wide Web*, pages 251–261. International World Wide Web Conferences Steering Committee, 2016.
18. R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*, 2004.
19. R. Speck and A.-C. N. Ngomo. Ensemble learning for named entity recognition. In *The Semantic Web–ISWC 2014*, pages 519–534. Springer, 2014.
20. F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *Proceedings of the VLDB Endowment*, 5(3):157–168, 2011.
21. R. Usbeck, A.-C. Ngonga Ngomo, M. Röder, D. Gerber, S. Coelho, S. Auer, and A. Both. AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, editors, *The Semantic Web – ISWC 2014*, volume 8796 of *Lecture Notes in Computer Science*, pages 457–471. Springer International Publishing, 2014.
22. R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, et al. Gerbil: General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1133–1143. International World Wide Web Conferences Steering Committee, 2015.
23. P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. Table search using recovered semantics, 2010.
24. P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 4(9):528–538, 2011.
25. C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein. Concept expansion using web tables. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1198–1208. International World Wide Web Conferences Steering Committee, 2015.
26. J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding tables on the web. In *Conceptual Modeling*, pages 141–155. Springer, 2012.
27. Z. Zhang. Towards efficient and effective semantic table interpretation. In *The Semantic Web–ISWC 2014*, pages 487–502. Springer, 2014.