Using Linked Data to Mine RDF from Wikipedia's Tables

Emir Muñoz
Fujitsu (Ireland) Limited,
Swords, Co. Dublin, Ireland
emir.munoz@ie.fujitsu.com

Aidan Hogan
Dept. of Computer Science,
Universidad de Chile
ahogan@dcc.uchile.cl

Alessandra Mileo
INSIGHT at NUI Galway,
Ireland
alessandra.mileo@deri.org

ABSTRACT

The tables embedded in Wikipedia articles contain rich, semi-structured encyclopaedic content. However, the cumulative content of these tables cannot be queried against. We thus propose methods to recover the semantics of Wikipedia tables and, in particular, to extract facts from them in the form of RDF triples. Our core method uses an existing Linked Data knowledge-base to find pre-existing relations between entities in Wikipedia tables, suggesting the same relations as holding for other entities in analogous columns on different rows. We find that such an approach extracts RDF triples from Wikipedia's tables at a raw precision of 40%. To improve the raw precision, we define a set of features for extracted triples that are tracked during the extraction phase. Using a manually labelled gold standard, we then test a variety of machine learning methods for classifying correct/incorrect triples. One such method extracts 7.9 million unique and novel RDF triples from over one million Wikipedia tables at an estimated precision of 81.5%.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.3.5 [**Information Storage & Retrieval**]: Online Information Services

Keywords

Linked Data; Web Tables; Wikipedia; Data Mining

1. INTRODUCTION

Wikipedia contains a wealth of encyclopaedic knowledge collaboratively curated by millions of users. Aside from unstructured textual content, Wikipedia articles contain rich factual data encoded in millions of tables. However, although these tables are presented in a semi-structured format, they are intended for human consumption and are not directly machine readable. The individual facts that each such table encodes are not readily recoverable by automatic methods and cannot be queried by users. Though the content of these tables could be imported into a relational representation, this still does not make the task of querying the tables much easier: a user would need to know, in advance, the specific structure of each table they wish to query against, which is impractical given the native heterogeneity of relational schemata for different tables. Furthermore, performing joins across such tables would require knowledge of co-reference (when tables refer to the same entities), since suitable primary/foreign keys are often not available.

Extracting the factual content of individual Wikipedia tables into a representation that is agnostic to their native structure requires methods to interpret the *semantics* of the table [21]. Such an interpretation aims to understand which entities referenced by the table are related to which other entities, and by what relationship. Once the semantics of the tables are (partially) recovered, their factual content can be presented as semantic triples [6]: binary relations of the form (s, p, o), where the subject entity s is related to the object entity o by the relationship p. The concept of semantic triples is isomorphic to that of the RDF data model [14], where URI identifiers can be used for entities and must be used for relationships. Irrespective of the structure of the originating tables, facts extracted as semantic triples could then be directly queried against using SPARQL [9]: the W3C recommended query language for the RDF data model.

Recovering the semantics of Wikipedia tables closely relates to two (mostly) orthogonal research topics.

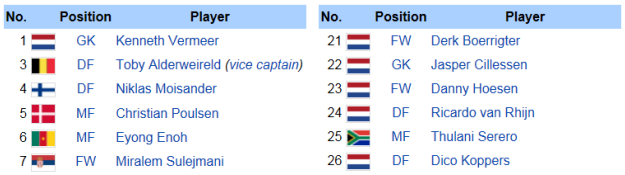
The first area relates to works on interpreting Web tables. Cafarella et al. [4] found 14.1 billion HTML tables in a Google crawl from 2008, estimating that 154 million contain high-quality relational data. These tables are extremely diverse in terms of representation, structure and vocabulary used; they often contain polysemous (or missing or otherwise vague) attribute labels, ambiguous free-text cell content and referents, cells spanning multiple rows and/or columns, split tables, obscured contextual validity, and so forth [11]. Recovering the semantics of such tables is thus exceptionally challenging, where existing methods typically aim to rather categorise or index the schemata of such tables [4, 6, 21].

The second area relates to works that export parts of the factual content of Wikipedia as RDF, creating knowledge-bases published as Linked Data. The most prominent works in this area are DBpedia [2] and YAGO2 [10]. Both works aim to extract factual knowledge from Wikipedia, representing the results as RDF, which can be queried by SPARQL.[1]

[1]For example, see `http://dbpedia.org/sparql`

WSDM'14, February 24–28, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2556195.2556266.



No.	Position	Player	No.	Position	Player
1	GK	Kenneth Vermeer	21	FW	Derk Boerrigter
3	DF	Toby Alderweireld *(vice captain)*	22	GK	Jasper Cillessen
4	DF	Niklas Moisander	23	FW	Danny Hoesen
5	MF	Christian Poulsen	24	DF	Ricardo van Rhijn
6	MF	Eyong Enoh	25	MF	Thulani Serero
7	FW	Miralem Sulejmani	26	DF	Dico Koppers

Figure 1: Split table listing all current AFC Ajax squad members (abridged from the Wikipedia article `http://en.wikipedia.org/wiki/AFC_Ajax`; CC–BY–SA)

However, both works rely primarily on mining facts from Wikipedia *info-boxes*: the attribute–value tables that appear on the top, right-hand side of many Wikipedia articles. Many such info-boxes follow standard templates that serve as a hook for writing bulk manual mappings. Even in cases where templates are not available, info-boxes are relatively straightforward to semi-automatically extract semantics triples from: s corresponds to the entity referred to by the article itself, p corresponds to the attribute and o corresponds to the value. Neither approach presents methods to extract the rich factual content present in the millions of relational tables that are embedded in the body of Wikipedia articles (which we henceforth refer to as *Wikitables*, as distinguished from info-boxes or tables-of-content).

Recently a number of works have looked to bridge these two areas—including works from Limaye et al. [12] and Mulwad et al. [17, 20, 16]—by using Linked Data knowledge-bases as references for interpreting the semantics of tables.

This paper follows on from these works with the aim of mining facts from all of Wikipedia's tables. Our methods are inspired by those from Web tables, but where we use the DBpedia Linked Data knowledge-base as a reference during the processing of tables. Though other knowledge-bases could also be used, we select DBpedia as a comprehensive dataset extracted from Wikipedia, with URIs for each subject of a Wikipedia article, and with a wide variety of relationships between those entities. We propose to first map entities in the tables to entities in the DBpedia knowledge-base. We can then query the knowledge-base for existing facts that involve those entities. By building a picture of which entities in the tables have which prior relationships, we can build an incomplete picture of the semantics of the table and attempt to extrapolate this to the rest of the table. We illustrate this idea with a motivating example.

Motivating Example

Figure 1 presents a Wikitable abridged from the Wikipedia article for "AFC Ajax": a Dutch football club.[2] The table containing factual content about players: their shirt number, country and position. There are also relationships between players and the entity described by the article (their current club is AFC Ajax). However, relevant relationships do not hold across all pairs of columns: for example, there is no obvious factual relationship between positions and countries.

Aside from the **No.** columns, the cells of the table contain hyperlinks to other articles in Wikipedia, including countries, football positions, and individual players. For example, the flags link to articles for the country; MF links to the article for "Midfielder": a position in football. These links provide unambiguous referents to Wikipedia entities, which can in turn be mapped directly to DBpedia entities and descriptions. We currently focus on the extraction of relations between cells containing wiki-links and do not look at plain-string values. For example, we would not try to extract player numbers from the above table.

First we can determine the contextual entity (and in this case, the protagonist of the table [5]) by the article in which it appears. We can then query the DBpedia knowledge-base to find relations between the context entity and various entities mentioned in the table cells. For example, the following SPARQL query asks for all relationships in the DBpedia knowledge-base that hold between the entities identified by the CURIEs `dbr:Kenneth_Vermeer` and `dbr:AFC_Ajax` (this query is for the purposes of illustration only; as discussed later, for performance reasons, we do not use SPARQL):[3]

```
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?rel WHERE { dbr:Kenneth_Vermeer ?rel dbr:AFC_Ajax }
```

This query returns the relationship `dbp:currentclub`. We find DBpedia triples with predicate `dbp:currentclub` relating $\frac{9}{12}$ entities in the **Player** to the protagonist `dbr:AFC_Ajax`. For the remaining three players in those columns, we can thus suggest the following triples as extraction candidates:

```
dbr:Niklas_Moisander dbp:currentclub dbr:AFC_Ajax .
dbr:Christian_Poulsen dbp:currentclub dbr:AFC_Ajax .
dbr:Danny_Hoesen dbp:currentclub dbr:AFC_Ajax .
```

Next we can look at relationships within the table. Taking an example for **Player** and **Position** columns:

```
SELECT ?rel WHERE { dbr:Thulani_Serero ?rel dbr:Midfielder }
```

from the reference knowledge-base, `dbp:position` is present in $\frac{11}{12}$ of the rows for entities in the **Player** column to entities in the **Position** column. We can thus suggest the missing relation as a candidate for extraction:

```
dbr:Miralem_Sulejmani dbp:position dbr:Forward .
```

Similarly, in the reference knowledge-base, the relationship `dbo:birthPlace` holds for many entities between the (unlabelled) country and **Player** columns, suggesting the analogous missing relations as candidates for extraction.

Though this method is naturally imprecise, we could suppose, for example, that the more rows for which a given relationship with a given predicate holds in the reference knowledge-base across two given columns, the higher the likelihood that that relationship exists on all such rows. Other features, such as a match between the label of the candidate relation and a column header, can strengthen confidence in the match. Such features of the extracted triple can be used to train a classifier for correct/incorrect triples.

Though a relatively "clean" example, the table depicted in Figure 1 already exhibits two potential obstacles: the table is split in two, and one cell contains multiple links ("Toby Alderweireld *(vice captain)*"). In practice, Wikitables exhibit a high-degree of diversity, which poses major challenges for extracting triples, where we rely on machine learning techniques to increase accuracy.

[2]The choice of example does not necessarily reflect the sporting allegiances of the present authors.

[3]The CURIE prefixes used in this paper are `dbr:` for `http://dbpedia.org/resource/`, `dbp:` for `http://dbpedia.org/property/`, and `dbo:` for `http://dbpedia.org/ontology/`

Our methods can ultimately be viewed from three perspectives. From the Web tables perspective, we use an existing knowledge-base as a reference to interpret the semantics of tables. From the Linked Data perspective, we enrich the reference knowledge-base with new facts mined from tables. From the Wikipedia perspective, we make the cumulative knowledge of tables available for user queries.

Organisation

The rest of this paper is organised as follows:

§2. We provide an overview of related works in the areas of Web tables, mining RDF triples from Wikipedia and extracting RDF triples from tables.

§3. We survey the set of Wikitables mined from a recent Wikipedia dump, which serves as our experimental corpus.

§4. We propose our methods for extracting candidate RDF triples from Wikitables using DBpedia as a reference dataset (as sketched in the motivating example).

§5. We describe the features and machine learning methods that we use to classify (in)correct facts.

§6. We conclude with a summary of results and possible future directions.

This paper extends an earlier work [18] where we surveyed Wikipedia's tables and presented initial ideas for mining facts from them. In this paper, we evaluate novel features and machine learning methods to classify triples as correct or incorrect and provide extended discussion throughout.

2. RELATED WORK

We review recent works on three relevant topics: processing Web tables, extracting RDF from Wikipedia, and extracting RDF from Web tables.

Web tables:

Hurst [11] divides the problem of processing Web tables into four sub-tasks: (1) *table location*: identifying tables in source documents, such as HTML pages; (2) *table recognition*: parsing and normalising the table into its constituent cells for further analysis; (3) *functional & structural analysis*: determining the size and nature of the table; (4) *table interpretation*: recovering the semantics of the table to allow for deep processing of its content.

In the context of HTML, sub-tasks (1) and (2) are typically carried out by looking for table-related HTML tags in the Web page (e.g. `table`, `td` and `tr`). Sub-task (2) also refers to segmenting the table into a relative spatial description, which considers issues like internal cell structure, split cells, spanning errors, and other normalisation techniques. For sub-task (2), Pivk et al. [19] proposed the TARTAR model, which normalises a table into a logical matrix.

For sub-task (3), the most common classification of tables is simply *Relational* vs. *Non-relational* (aka. *Genuine* vs. *Non-genuine*); non-relational tables include those used for formatting and navigation [22, 4]. Going further, Crestan and Pantel [6] define a taxonomy with twelve structural classes of HTML tables and Yoshida et al. [24] propose nine groups for attribute–value tables, etc.

Sub-task (4) relates to interpreting tables and attempting to recover their semantics. Cafarella et al. [4] proposed the WebTables system, which adopts a distributional semantics approach, extracting co-occurrence distributions for attribute labels in tables for search and other applications. Zwicklbauer et al. [25] also use an existing knowledge-base to annotate the semantic types of individual table columns based on the most common type found in each column's cells. Venetis et al. [21] similarly look to annotate columns of tables, sourcing a set of target class and relationship labels from the Web. For interpreting attribute–value tables, Crestan et al. [6] introduce the *protagonist problem*: identifying the subject of relations for attribute–value tables in order to generate semantic triples. (We discuss works that can extract triples from tables later in this section.)

Novelty: We focus on the tables of Wikipedia. For sub-tasks (1–2), we re-use the TARTAR model [19] and parse Wikitables using HTML tags, such as `table`, `td`, etc. For sub-task (3), we simply look for tables with class `wikitable` to identify our target class of tables. Furthermore, we apply the notion of a protagonist [6] to generic relational tables (as per the AFC Ajax example). Our methods re-use and extend upon various ideas made in the Web-tables literature for extracting triples from Wikipedia tables.

Extracting RDF from Wikipedia

As aforementioned, the DBpedia knowledge-base [2] publishes RDF extracted from Wikipedia as Linked Data. The system extracts RDF from Wikipedia articles in various languages. DBpedia URIs are minted for the protagonist of each Wikipedia article. RDF triples are extracted from markup including titles, abstracts, categories, images and various forms of links. The richest source of information are the attribute–value "info-boxes" presented in the top right of Wikipedia articles (where available). DBpedia supports user-contributed mappings of RDF from tables in Wikipedia, but only isolated mappings have been provided.[4]

YAGO2 [10] also extracts RDF from Wikipedia, relying on hard-coded declarative rules that specify extraction, normalisation and implication logic for mapping (primarily) infobox attributes to known terms in the ontology. Facts are enhanced and annotated through use of external knowledge-bases, such as WordNet synsets to enrich wiki-categories, and GeoNames to enrich geographical information. Estimates for the precision of YAGO2 lie at around 95% [10].

Novelty: Although both YAGO2 and DBpedia support manual mappings from generic tables in Wikipedia, the coverage of these mappings appears to be very limited: both works rather focus on extraction from info-boxes. To assure novelty, we thus deliberately ignore info-boxes and instead target only Wikitables embedded in the body of the article. We currently use DBpedia as a reference knowledge-base, but in future could also use other reference knowledge-bases that are reconcilable with Wikipedia entities, such as YAGO2 or Freebase, or perhaps even Wikidata.

Extracting RDF from Tables

Various languages, such as R2RML [7], can express custom mappings from relational database tables to RDF. In terms of representing the *structure* of tables in RDF, Ding et al. [8] propose extracting each row of the table as a subject, each

[4] `http://mappings.dbpedia.org/index.php?title=Special%3APrefixIndex&prefix=Table&namespace=204`

column as a predicate and each cell as an object. Similar methods are used for the Direct Mapping standard [1].

More closely related to our own work are the methods of Limaye et al. [12] and Mulwad et al. [17, 20, 16], who take a reference knowledge-base and use it to annotate entities and columns in tables with types and relationships. As such, these approaches can be used to extract triples from tables.

Novelty: Our approach is automatic and does not require an input mapping (per R2RML). We export content that can be queried independently of the structure of the originating table (vs. Ding et al. [8] and the Direct Mapping).

Although Limaye et al. [12] evaluate average timings of their extraction for 250,000 Web tables, their work and the works by Mulwad et al. [17, 20, 16] evaluate the quality of relation extraction based on 25–36 hand-picked Wikitables and 150–371 hand-picked Web-tables, whereas we evaluate the quality of our extraction for over one million Wikipedia tables (all Wikitables meeting certain minimal criteria, laid out in §3.2) based on random sampling. In general, we focus specifically on the problem of relation extraction in the context of Wikipedia tables, running the extraction process over the entire corpus (post-filtering), presenting a detailed set of "row-centric" features for machine learning methods to perform classification, and evaluation based on sampling from all tables. In doing so, we exploit features specific to Wikipedia to improve results: for example, we use Wikilinks to detect entities and we extract additional relations from table cells to the protagonist of the article, etc.

3. SURVEY OF WIKITABLES

We begin our core contribution by formalising our notion of a table, defining the tables in Wikipedia that we consider in-scope for extraction (Wikitables), and providing some statistics on the set of Wikitables extracted from the February 13th, 2013 English Wikipedia dump.

3.1 Classifying Wikipedia's Tables

Wikipedia editors can choose from three classes of tables; each has a different value for the HTML attribute class in the associated table tag in the Wikipedia page, as follows:

- toc: table of contents summarising the layout of the article, typically found after its abstract;
- info-box: attribute–value tables embedded in the top-right of the article providing core information about the protagonist, typically following a standard template based on the type of protagonist (e.g., city, person, etc.);
- wikitable: tables embedded in the article's body, typically containing relational data about the protagonist.

We focus on the wikitable class since toc tables refer to article layout, not factual content, and the extraction of facts from info-box tables has already been near-exhaustively covered by works such as DBpedia and YAGO2. To the best of our knowledge, we are the first work to tackle a full-scale extraction of facts from the wikitable class of tables.

3.2 Normalising Wikipedia's Tables

The structure and content of Wikitables can be quite complex. Since these tables are intended for human consumption, their layout is guided by visual aesthetics. Tables often contain *spans* to avoid repetition: these spans cover multiple cells with one entry, and include column-spans (colspans), row-spans (rowspans) or a combination of both. To avoid tables wider than the standard display width or to avoid ugly long and narrow tables, editors will often employ split tables (as per Figure 1). Cells may also contain (i) multiple values, (ii) prose-text alongside the primary value giving justification or context for that value, (iii) superscript references to sources, (iv) images or other embedded content, (v) empty cells in "optional" columns, and so forth.

To overcome this heterogeneity, we normalise Wikitables by "squaring" them and filtering text, such that they can be represented as a logical matrix of cells prior to extraction.

Table 1: Overview of tables in the dump

• Total articles:	10,854,204	100.00%
▸ Articles w/tables:	1,875,415	17.28%
• All tables:	3,923,427	100.00%
▸ Ill-formed:	163,105	4.16%
▸ Well-formed:	3,760,322	95.84%
◦ toc:	1,648,786	43.85%
◦ info-box:	815,350	21.68%
◦ wikitable:	1,296,186	34.47%
▹ $< 2 \times 2$:	158,940	12.26%
▹ $\geq 2 \times 2$: (*used*)	1,137,246	87.74%

Table Model

We consider as input a source of tables $\mathcal{T}$ (a set of tables). We model an individual table $T \in \mathcal{T}$ as an $m \times n$ matrix M_T. Table headers are very common in tables, and can be simple (one row) or hierarchical (multiple rows). We let h denote the index of the bottom header row, such that $M_T(i, .)$, $i \leq h$ denotes the header rows of M_T.

Extraction and Normalisation

Our initial goal is to convert a Wikipedia article in HTML to a source of tables $\mathcal{T}$. Mapping HTML tables to square matrices is often challenging due to syntax-level issues (e.g., non-closed HTML tags, fewer or more columns in some rows creating jagged tables, spans, nesting, etc.).

Though we cannot *fully* normalise the tables (e.g., we do not rejoin split tables), we use TARTAR [19] to apply some standard normalisation techniques, as follows. Tables often contain their caption embedded in the first row: we discard that row when generating the matrix. Header rows are identified by the (<TH>) tag. For each table, we must also deal with colspans and rowspans: we divide such cells and copy the original content into each division. Where we find "jagged rows", we consider missing cells to contain blank strings. If the table can be parsed into a matrix M_T (i.e., is syntactically valid and dense), we call the table *well-formed*. Otherwise we call it *ill-formed*. Finally, we only consider well-formed tables where the size of M_T is at least 2×2; i.e., it contains at least two rows *and* at least two columns.

3.3 Corpus Composition

We extract our corpus of Wikitables from the February 13th, 2013 English Wikipedia dump. We first apply the Bliki engine parser[5] to convert wiki-markup to HTML articles.

[5] http://code.google.com/p/gwtwiki/

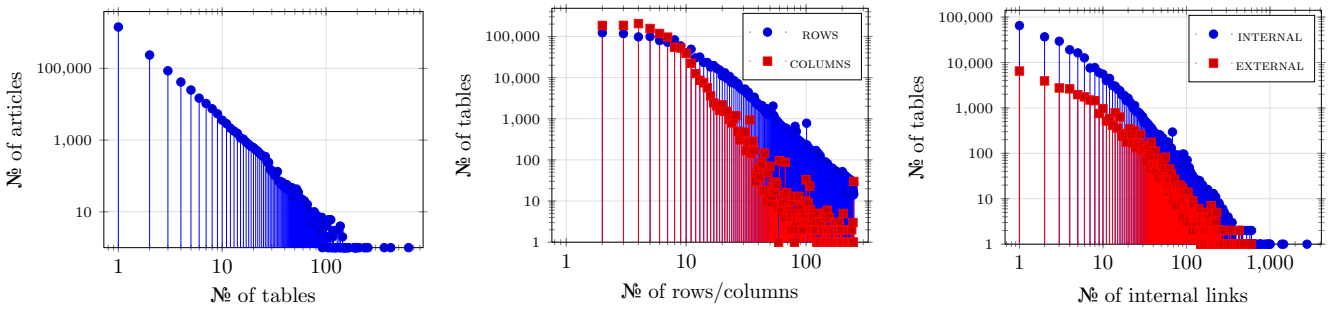


Figure 2: Various distributions for the Wikitables considered

Each HTML page is then cleaned and canonicalised (fixing syntax mistakes) using CyberNeko[6] before all HTML tables (including nested tables) are extracted.

Table 1 provides a breakdown of the high level details of the corpus. We found 10.85 million articles (more accurately "pages"), containing a total of 3.92 million HTML tables, where 17.3% of the articles contained at least one table.

In Figure 2, we plot the distributions of (a), tables per article, (b) number of tables for increasing numbers of rows/-columns, and (c) number of internal/external links per table. All distributions are plotted in log/log scale, where we see long-tailed distributions for all such dimensions. Furthermore, we make the following observations:

- The article[7] with the largest number of well-formed tables contains 623 (most of them nested inner-tables).
- Considering articles with at least one table, there are 1.66 tables per article.
- The maximum number of detected rows in a table is 250, commonly found in List_of_* articles (likely due to a Wikipedia formatting constraint). The average number of rows per table is 12.44.
- The maximum number of columns in a table[8] is 250 (due to an erroneous colspan for a caption in a table with 3 columns). The average number of columns per table is 5.55.
- The highest number of internal links in a single table is 2,774, and 594 for external links. The averages are 1.93 and 0.35 for internal and external links, respectively. Internal links are important for us since they ensure that we can map cell entries to Wikipedia articles (and thus to DBpedia entities).
- 19.4% of tables do not contain column headers. 79.9% contain headers only in the first row, 7.4% contain headers in the second row, and the remaining tables contain captions in further rows. 5.5% of tables contain non-empty embedded captions.

We thus view this corpus of tables as a rich source of structured data that can be exploited for information extraction and ultimately for triplification, with many tables containing a high number of internal wiki-links (as per the bottom plot in Figure 2), which can be used for entity disambiguation. After filtering toc, infobox, small and ill-formed tables, we end up considering 29.0% of all Wikipedia tables for our extraction: 1.14 million tables in total.

[6] http://nekohtml.sourceforge.net/

[7] http://en.wikipedia.org/wiki/Winners_and_runners-up_in_the_legislative_ elections_of_Nepal_1994_and_1999

[8] http://en.wikipedia.org/wiki/2007-08_QMJHL_season

4. MINING RDF FROM WIKITABLES

To extract RDF from these Wikitables, we rely on a reference knowledge-base, where we use DBpedia for the purposes of this paper. We extract internal links present in cells and map them directly to DBpedia entity URIs. We then perform lookups on DBpedia to find relationships that exist between entities on the same row of the table. These relationships are then proposed as candidates between entities in cells of analogous columns on other rows.

4.1 Reference Knowledge-base

We use English-language data from DBpedia v3.8, describing 9.4 million entities. The overall corpus consists of 465 million unique RDF triples, and contained 57,985 unique RDF relations (i.e., RDF predicates), 48,293 of which were in the DBpedia property namespace, 1,397 of which were in the curated DBpedia ontology namespace, and 28 of which were from external vocabularies.[9]

4.2 Entity/Resource Extraction

In Section 3, we described how we extract a set of tables from Wikipedia, where each table is represented as an $m \times n$ matrix M_T and where each cell of the matrix $M_T(i, j)$ (for $i \leq m$, $j \leq n$) can contain a variety of content. However, we are primarily interested in internal wiki-links present in the cells since they can be directly mapped to DBpedia entities. We represent the set of wiki-link URLs in a cell as $W_T(i, j)$, which excludes superscript reference links, links with prefix Image: or File:, and external links. We also strip fragment identifiers from URLs (i.e., strings following a hash).[10]

For each $w \in W_T(i, j)$, we map the corresponding Wikipedia URL to a DBpedia entity URI by following redirects and replacing the namespace http://en.wikipedia.org/wiki/ with http://dbpedia.org/resource/; for optimisation reasons, we cache redirects. We additionally filter entities corresponding to DBpedia categories and list pages. We denote

[9] We also found noise where 8,267 DBpedia entity URIs referring to Wikipedia templates appeared as predicates.

[10] Typically these are references to sections or to footnotes, but occasionally they may refer to a sub-topic in the article. However, DBpedia does not have corresponding URIs for the entities referred to by these sub-topics.

by $E_T(i,j)$ the resulting set of DBpedia entity URIs extracted for wiki-links $W_T(i,j)$ for table cell $M_T(i,j)$.

4.3 Discovering Relations

We aim to discover two types of relations: between entities on the same row of a table and between entities in the table and the protagonist of the article.

To discover relations within tables, we query the DBpedia knowledge base for existing relationships between entities in different cells of the same row. We look for relations between all pairs on the same row i but in different cells: $\{(e_{i,j}, e_{i,k}) \mid e_{i,j} \in E_T(i,j), e_{i,k} \in E_T(i,k), 1 \leq i \leq m, 1 \leq j < k \leq n\}$. Any relations found for $(e_{i,j}, e_{i,k})$ are suggested as candidates for relating $(e_{h,j}, e_{h,k})$ for $1 \leq h \leq m$, $h \neq i$.

To discover relations from entities in the table and p the protagonist entity of the article, we query the knowledge-base for pairs $\{(e_{i,j}, p) \mid e_{i,j} \in E_T(i,j), 1 \leq i \leq m, 1 \leq j \leq n\}$. Any relations found for $(e_{i,j}, p)$ are suggested as candidates for relating $(e_{h,j}, p)$ for $1 \leq h \leq m$, $h \neq i$.

We query the DBpedia knowledge-base for existing relationships that hold in either direction for each such pair. We run these queries over local indexes of DBpedia data with in-memory caching to efficiently support repeated queries caused by colspans. Though we avoid using a SPARQL engine for performance reasons, the lookups for a pair $(e1, e2)$ would be equivalent to the following query:

```
SELECT DISTINCT ?p1 ?p2
WHERE { { <e1> ?p1 <e2> } UNION { <e2> ?p2 <e1> } }
```

The `UNION` clause in SPARQL is used to denote a disjunctive query. The terms `<e1>` and `<e2>` are instantiated with the DBpedia URIs of the entities in question.

Example 4.1. With respect to relations within tables, if we consider some example cells from row 3 of table T in Figure 1, we have:

$$E_T(3,1) = \emptyset,$$
$$E_T(3,2) = \{\texttt{dbr:Belgium}\},$$
$$E_T(3,3) = \{\texttt{dbr:Defender}\},$$
$$E_T(3,4) = \{\texttt{dbr:Toby\_Alderweireld}, \texttt{dbr:Vice\_Captain}\}.$$

If we query DBpedia indexes looking for relationships in this row between, for example, columns 2 and 4, we could find `dbo:birthPlace` from `dbr:Toby_Alderweireld` in $E_T(3,4)$ to `dbr:Belgium` in $E_T(3,2)$. We can suggest that this relationship may hold, e.g., from entities in $E_T(2,4)$ to $E_T(2,2)$, and so forth for all other non-header rows in the table.

If we query DBpedia for existing relations between the entity `dbr:Toby_Alderweireld` in $E_T(3,4)$ to the article entity `dbr:AFC_Ajax`, we will find `dbp:clubs` and `dbo:team`. We consider these as candidate relations potentially holding between `dbr:AFC_Ajax` and other entities in column 4.

Of course, this process can be imprecise. We will discuss this further in Section 5. □

4.4 Triple Extraction

We extract all candidate relationships found for all Wikitables as RDF triples:

Example 4.2. Consider applying the `dbo:birthPlace` candidate relation found in Example 4.1 to row 2 (from $E_T(2,4)$ to $E_T(2,2)$). The resulting RDF triple is then:

```
dbr:Ricardo_van_Rhijn dbo:birthPlace dbr:Netherlands .
```

representing the fact that Ricardo van Rhijn was born in the Netherlands. □

Based on preliminary evaluation of the extraction process, we chose to ignore reflexive triples and triples with the predicate `dbo:wikiPageDisambiguates`, which very often pertained to incorrect facts.

Though we do not wish to focus on computational performance in this paper, we can make some high-level remarks about the general scalability of our approach. As stated previously, we use local indexes of the DBpedia knowledge-base for answering queries, and for each pair, we perform two atomic on-disk lookups for relations in either direction. However, given that tables can potentially contain hundreds of entities, and that the number of entity-pairs to consider is quadratic for a given row, each table may require a large amount of lookups for relations.

However, with respect to scalability of the global processing, our problem can be effectively partitioned and parallelised on a per-table basis. For example, given a cluster of shared-nothing commodity servers, if we make the full index for the reference knowledge-base available to each machine (in our scenario, a 7.7GB file), individual tables can then be assigned arbitrarily to each machine, and the extraction of triples run in an embarassingly parallel manner. We adopt this approach and using eight machines (ca. 2005) with 4GB of RAM, 160GB SATA hard-drives, 2.2GHz single-core processors, assigning an even work-load of input Wikipedia articles to each, the full process of extracting and normalising Wikitables from the articles and computing the full set of candidate triples took approximately 12 days.[11]

We extracted a total of 53.2 million candidate RDF triples, of which 37.3 million were unique, of which 34.9 million in turn were not already in DBpedia. These candidate triples are assigned features during extraction that serve as input to the classification process described in the next section.

5. LEARNING RDF TRIPLES

As alluded to before, many of the triples we extract in the previous phase are likely to be incorrect for a variety of reasons. We illustrate this with two further brief examples.

Example 5.1. In the table of Figure 1, we do not distinguish the entities Toby Alderweireld and *vice captain*, which appear in the same cell. Hence, when multiple entities appear in one cell, we may extract incorrect triples, such as:

```
dbr:Vice_Captain dbo:team dbr:AFC_Ajax .
```

To illustrate another common case we encountered: in our reference knowledge-base, the entity Derk Boerrigter is related to the protagonist AFC Ajax through the relation `dbp:youthclubs`. However, this relation would not hold between other players in the same column and AFC Ajax even though said candidate triples will have been produced in the previous phase: just because a relation holds for (at least) one row does not mean it *must* hold for others. □

[11] An alternative route for optimisation could be to build a compressed in-memory index over the reference knowledge-base or to use a solid-state disk, but this was not feasible for our hardware.

This section thus evaluates a variety of machine-learning methods with respect to identifying correct triples from our set of raw extracted triples. For this phase, a set of features can be associated with extracted triples to help determine if the triple is likely to be (in)correct. For example, we can consider the number of rows for which a relation holds versus the total number of rows in a table (e.g., $\frac{1}{6}$ in the case of the `dbp:youthclubs` relation mentioned in the previous example), or the presence of multiple entities in a cell (cf. Example 5.1), or string matching between a column header and a predicate label (e.g., `dbp:position` for the **Position** header in Figure 1). Along these lines, we now discuss a set of features we associate with extracted triples, which can be used by machine learning methods (described later in the section) for classification as correct/incorrect.

5.1 Features

In previous work, the classification problem was tackled primarily by ranking each relation according to the number of times it held in the knowledge-base between entities in those two columns. However, we feel that there are many factors—such as the structure of the table, the nature of the cell content, textual overlap with one of the column headers, etc.—that provide important signals for determining the accuracy of extracted triples. Furthermore, like Limaye et al. [12], we believe that statistics about the predicate in the reference knowledge-base serve as useful input; e.g., if a predicate is generally found to follow a one-to-one relation (e.g., `dbp:captain`), then it is unlikely to be a good relation for linking an article (e.g., `dbr:AFC_Ajax`) to multiple entities in a table column (e.g., **Players**).

The full set of features we generate is listed in Table 2, broken down by article features, table features, column features (i.e., features specific to the subject/object column from which the triple was extracted), predicate features (i.e., features specific to the predicate of the extracted triple), cell features (i.e., features specific to the subject and object cells from which the triple was extracted), predicate/column features (features that are a function of the predicate and subject/object columns) and triple features. Feature 38 tracks if the triple exists in DBpedia for reference only and is never used for learning. We include a wide range of features, with the goal of applying feature selection at a later stage.

Given that most of the features are self-explanatory, we give a brief explanation for non-trivial entries:

15 The number of pairs of the form $(e_{i,s}, e_{i,o})$ for s the subject column, o the object column and $h < i \leq m$

16 As per feature 15, but only unique pairs (for the given table) are counted

17–20 All predicate features are normalised as a real value in the range $(0, 1]$ (counts are divided by the maximum such value for all predicates)

24–25 The amount of prose text in the cells from which the subject and object of a triple is extracted, where long text is indicative of possible noise or hidden context

26–27 A boolean value indicating the presence of, e.g., newlines, bullets or other HTML formatting tags in the source cells

28–29 The string similarity, measured as the closer value given by the Jaro–Winkler distance and the Sørensen–Dice coefficient, between the predicate label and the header label for the subject/object column

31 For how many rows there exists a relation in the KB with the given predicate from the subject column to the object column

33 For how many pairs in feature 15 there exists a relation in the KB with the given predicate

35 For how many unique pairs in feature 16 there exists a relation in the KB with the given predicate

With respect to features 31, 33 and 35; which are similar with respect to measuring ratios of rows/relations for which a predicate exists in the KB; we include all three versions since we found that certain tables in Wikipedia can exhibit

Table 2: Full list of features

Article Features
(1) № of tables
Table Features
(2) table id in article
(3) № of rows
(4) № of columns
(5) ratio: (3)/(4)
(6) total relations extracted
Column Features
(7) subject column id
(8) object column id
(9&10) № of entities in s&o cols.
(11) ratio: (9)/(10)
(12&13) № of unique entities in s&o cols.
(14) ratio: (12)/(13)
(15) potential relations
(16) unique potential relations
Predicate Features
(17) normalised triple count
(18) normalised unique subject count
(19) normalised unique object count
(20) ratio (18)/(19)
Cell Features
(21&22) number of entities in s&o cells
(23) ratio (21)/(22)
(24&25) string length in s&o cells
(26&27) formatting present in s&o cells
Predicate/Column Features
(28&29) string sim. for pred and s&o header
(30) maximum between (28) and (29)
(31) № of rows where relation holds
(32) ratio: (31)/(3)
(33) № of potential relations held
(34) ratio: (33)/(15)
(35) № of unique potential relations held
(36) ratio: (35)/(16)
Triple Features
(37) from article or body relation
(38) already exists in DBpedia

very different values for these features due to duplicate entities existing in individual cells (often due to spans).[12]

5.2 Machine Learning Models

We use machine learning models to classify (in)correct triples according to these features. For evaluation, we select five classifiers listed by Witten et al. [23] capturing a variety of different high-level approaches. We select one probabilistic classifier: Naïve Bayes; two based on trees: Bagging Decision Trees and Random Forest; and two based on functions: Support Vector Machines (SVM) and Simple-Logistic. We use all such methods to learn a classification model using 10-fold cross-validation over a gold standard of manually annotated triples (described in the next sub-section). The resulting models can then be used for the binary classification of triples (correct/incorrect) according to its supporting features. Experiments are run using the WEKA machine learning framework.[13] Full details of these machine learning techniques are out-of-scope (see, e.g., [13, 23]), here we provide a brief overview:

Naïve Bayes is a simple probabilistic classifier. The classification is assigned based on the probabilistic distribution of the training set. We highlight that this method (naïvely) assumes that attributes are independent, which is certainly not true in our case, but we can still apply and test the model.

Bagging Decision Trees consists of an ensemble of decision trees (trees of nodes that branch depending on the value of a certain feature) for each class, where the final classification decision for each instance is made based on a form of voting of the committee of classifiers.

Random Forest is another ensemble classifier that consists of multiple decision trees. The class is assigned according to the mode of the classes output by individual decision trees.

Support Vector Machine represents each feature tuple from the training data as a point in Euclidean space and searches for a good hyperplane division between the classes, which is then used for binary classification. We use SVM with a polynomial kernel.

Simple-Logistic uses logistic regression models, and is appropriate for classification in domains with (continuous) numeric attributes.

5.3 Gold Standard

We require a gold standard to train and evaluate these models. For this, we randomly selected 750 unique triples from the total set of 37.3 million triples extracted in the previous phase. Three judges (authors on this paper) then labelled the 750 triples as being *correct*, *incorrect* or *unknown*. The information available for each triple were the subject, predicate and object URIs, as well as a link to the original Wikipedia article containing the source table and a number identifying which table in the article was the source. Triples were generally validated by visiting the Wikipedia article in question to see the original table, as well as Web searches for the entities in question. We did not pre-agree on any judging strategies, where interpretation of the validity of triples was left to the discretion of individual judges.

Table 3: Inter-rater consensus for manually labelled triples

All *correct*	234	31.2%
Some *correct*, Rest *unknown*	27	3.6%
Positive consensus	**261**	**34.8%**
All *incorrect*	195	26.0%
Some *incorrect*, Rest *unknown*	48	6.4%
Negative consensus	**243**	**32.4%**
Some *correct* and some *incorrect*	**246**	**32.8%**

In terms of inter-rater agreement for the three judges, we computed a Fleiss' κ coefficient of 0.45 (0 indicates no agreement, 1 indicates perfect agreement), which by convention is considered as "moderate agreement". Looking through examples of disagreement, some common themes emerged. In particular, the relations of DBpedia are often not well defined and their meaning often has a subjective dimension (for example, `dbr:Pulp_Fiction dbo:starring dbr:Quentin_Tarantino` when he arguably only had a cameo appearance). Another recurring issue was that of temporality: for example, would stating that `dbr:Bill_Clinton` has the relation `dbp:president` to `dbr:United_States` be correct or incorrect? These questions are subjective and depend on the precise semantics for the predicate and for which DBpedia provides no definitive intensional answer. Table 3 summarises the consensus between raters. For the gold standard, we filter 246 examples where there were conflicting labels from different judges, leaving 504 triples.

From these results, we can also get some direct insights into the accuracy of the raw candidate triples. Based on the consensus labels, 51.8% of triples are considered correct and 48.2% considered incorrect without any further classification. However, 112 of the 504 triples appear in DBpedia, where the analogous results filtering these triples would be 39.6% and 60.4% respectively.[14] Next we use machine learning methods to further boost accuracy by classifying these raw extracted triples as correct/incorrect.

5.4 Classification Evaluation

To train and evaluate the five classifiers, we apply 10-fold cross-validation over the 504 triples with non-conflicting labels: we randomly generate 10 non-overlapping test sets with ∼54 examples each, while the remaining ∼450 examples are used for training purposes in each run. As a baseline, we also compare three simple rule-based models relying on features 32, 34 and 36 respectively (i.e., ratios of rows, relations and unique relations in a table for which a given predicate is found to hold), where extracted triples with a score above a threshold for the given feature are classified as positive and triples below the threshold as negative.

Given that we use many features, and that some may not be deemed relevant to the machine learning process, we

[12] See `http://en.wikipedia.org/wiki/KJEE` for such an example. The table contains 11 rows. For columns **Venue** and **City**, there exists 18 potential relations and 4 unique potential relations.

[13] `http://www.cs.waikato.ac.nz/ml/weka/`

[14] These triples are left in to provide more positive examples. Interestingly, 7 out of the 112 DBpedia triples were marked as incorrect (6.3%).

Table 4: Classification performance for each approach with best figures for each metric underlined

Schema	Acc.	Pre.	Rec.	F_1	Output
Feature 32	65.41	50.77	74.16	60.27	15,415,810
Feature 34	70.18	58.08	78.65	66.81	3,864,653
Feature 36	67.59	64.23	70.46	67.20	4,263,470
Naïve Bayes	75.75	88.85	71.30	79.11	3,902,861
B.D. Trees	78.13	81.54	77.37	79.40	7,871,344
Rand. Forest	76.34	76.15	77.65	76.89	8,979,392
SVM	72.56	72.43	75.77	74.06	15,320,590
SimpleLogistic	78.53	79.62	79.01	79.31	6,376,427

investigated using standard "feature selection" techniques, using a greedy best-first algorithm to search for a set of features that optimises accuracy (i.e., ratio of correct true positives and true negatives) for each model. However, we found that the feature selection process had little effect on accuracy; hence we opted not to use feature selection.

Next, for the model selection, we applied a cost–benefit analysis to determine the optimal decision threshold for each classifier (aka. calibration), i.e., we maximise accuracy by varying the thresholds of each classifier below which an example is predicted for the -1 (negative) class. Considering these optimised thresholds, Table 4 compares the accuracy, recall, precision and F_1-measure of the eight classifiers, where the latter three metrics are given as weighted averages over the positive and negative classes. We also ran the trained models over the raw extracted triples and count the number of unique novel triples output with a positive classification (from a possible 34.9 million unique triples not appearing in DBpedia, of which we would expect approx. 13.8 million (39.6%) to be correct).

We first see that the single-feature baseline models are outperformed by the multi-feature models in terms of accuracy, precision and F_1 measures. Although Feature 32 classifies the most triples as correct, it does so at the cost of much lower precision. Also we see that the number of output triples can be significantly lower than would be expected based on the corresponding recall figures. We believe that this disparity is due to 40.3% of the positive examples used in the gold standard training data coming from DBpedia; such positive examples are removed from the 34.9 million triples classified in the output. The selected thresholds for the training data may thus be too selective for the real-world data without these examples.

Amongst the other multi-feature models, we see that there is a clear trade-off in terms of precision versus recall. Thus it is difficult to identify a "clear" winner. For example, the highest precision (88.85%) is achieved by Naïve Bayes, but this comes at the cost of recall and output triples captured. In terms of combined measures such as accuracy and F_1 measure, Bagging Decision Trees is close to the metrics for SimpleLogistic but outputs significantly more positive triples. On the other hand, SVM outputs nearly double the number of output triples that Bagging Decision Trees does, but at a lower precision. Hence it would seem that Bagging Decision Trees classifying 7.9 million unique novel output triples with an estimated 81.5% precision is amongst the most favourable results.

6. CONCLUSION

Wikipedia tables contain rich encyclopaedic knowledge but their content cannot be directly queried against. Herein we proposed methods to extract factual content from over one million Wikipedia tables. Following existing works, we use an existing Linked Data knowledge-base as a reference to guide the extraction process and to help automatically piece together the semantics of the tables.

More precisely, we look for existing relationships in the knowledge-base between entities on the same row of a table and extrapolate them as candidate relationships between entities in analogous columns on different rows of that table. We also extract relations from entities in the same column of a table to the protagonist of the article in which the table is found. Using DBpedia as a reference knowledge-base, we successfully mined 34.9 million unique and novel triples from 1.1 million Wikipedia tables with a precision of 40% without any further classification.

We associate each extracted triple with a rich set of features and investigate a variety of machine-learning methods to identify triples as correct or incorrect. Using a manually-labelled random-sample of extracted triples as a gold standard, we found that different classifiers produced different trade-offs with respect to precision/recall. Arguably the best result was given by the Bagging Decision Trees ensemble classifier, with which we mined 7.9 million unique novel triples at an estimated 78.1% accuracy, 81.5% precision, 77.4% recall and 79.4% F_1 measure.

Other works have also tackled relation extraction from tables using Linked Data knowledge-bases. Limaye et al. [12] report accuracy figures of 52–69% for this task over a selection of manually labelled hand-picked tables. In very recent work, Mulwad et al. [16] report F_1-scores of between 89–97% for this task over a similar evaluation dataset. However, the results from these papers are not directly comparable with ours since, for example, we apply extraction over all well-formed Wikitables and randomly sample our test-set (vs. using around 200 manually selected tables), we extracted relations on a per-row basis (vs. selecting the top-k relations for each pair of columns), we use wiki-links to map to entities in our knowledge-base (vs. using string matching and search methods), and so forth.

To the best of our knowledge, our work is the first to look at extracting the factual content of Wikitables at full-scale and is also the first to present end-to-end results for extracting semantic triples from a large corpus of Web tables (albeit only for the easier case of Wikitables).

Future work

We identify three major directions in which our work can be extended, generalised, and/or improved:

1. With respect to improving our extraction methods, we do not yet exploit the semantics of the reference knowledge-base, such as the types of entities, the domain or range of properties, the hierarchies of classes, etc. Considering such semantics could help to improve our precision and recall measures further, by, for example, splitting columns that contain multiple entities in each cell according to the type of those entities (e.g., player/position), or by rejecting relationships where the domain/range definitions do not match.

2. Our approach could be generalised to use other Linked Data knowledge-bases of encyclopaedic content—such as YAGO2 [10] or Freebase [3]—as an alternative to DBpedia. Such knowledge-bases have already been investigated by Limaye et al. [12] and Mulwad et al. [16]. Such a generalisation would require a method for mapping entities referenced in Wikipedia tables to native identifiers in the reference data-set: DBpedia offers links to both YAGO2 and Freebase that could be used for this purpose. Our approach could also take advantage of multiple knowledge-bases at once by taking the union of candidate triples produced for each such reference collection.

3. Most ambitiously, our method could potentially be generalised to Web tables in the broader sense [12, 16]. Again, a method for matching entities in the tables with the reference knowledge-base would be required, where some standard NER technique(s) could be used. For example, the DBpedia Spotlight tool [15] already offers entity recognition for DBpedia entities.

Source code, data, models and a demonstrator are available from `http://emunoz.org/wikitables/`.

Acknowledgements: *This work was supported in part by Fujitsu (Ireland) Ltd., by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004, and by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.*

7. REFERENCES

[1] M. Arenas, A. Bertails, E. Prud'hommeaux, and J. Sequeda. A Direct Mapping of Relational Data to RDF. W3C Recommendation, September 2012.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia – a crystallization point for the Web of Data. *J. Web Sem.*, 7(3):154–165, 2009.

[3] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In J. T.-L. Wang, editor, *SIGMOD Conference*, pages 1247–1250. ACM, 2008.

[4] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the Web. *PVLDB*, 1(1):538–549, 2008.

[5] E. Crestan and P. Pantel. A fine-grained taxonomy of tables on the Web. In J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, and A. An, editors, *CIKM*, pages 1405–1408. ACM, 2010.

[6] E. Crestan and P. Pantel. Web-scale table census and classification. In I. King, W. Nejdl, and H. Li, editors, *WSDM*, pages 545–554. ACM, 2011.

[7] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language. W3C Recommendation, September 2012.

[8] L. Ding, D. DiFranzo, A. Graves, J. Michaelis, X. Li, D. L. McGuinness, and J. Hendler. Data-gov Wiki: Towards Linking Government Data. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*. AAAI, 2010.

[9] S. Harris and A. Seaborne. SPARQL 1.1 Query Language. W3C Recommendation, March 2013.

[10] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artif. Intell.*, 194:28–61, 2013.

[11] M. Hurst. Layout and language: Challenges for table understanding on the Web. In *Workshop on Web Document Analysis (WDA)*, pages 27–30, 2001.

[12] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching Web tables using entities, types and relationships. In *PVLDB*, 2010.

[13] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications. Springer, 2011.

[14] F. Manola and E. Miller. RDF Primer. W3C Recommendation, February 2004.

[15] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia spotlight: shedding light on the web of documents. In *I-SEMANTICS*, pages 1–8, 2011.

[16] V. Mulwad, T. Finin, and A. Joshi. Semantic message passing for generating Linked Data from tables. In *ISWC*, pages 363–378, 2013.

[17] V. Mulwad, T. Finin, Z. Syed, and A. Joshi. T2LD: Interpreting and Representing Tables as Linked Data. In *ISWC Posters&Demos*, 2010.

[18] E. Muñoz, A. Hogan, and A. Mileo. Triplifying Wikipedia's Tables. In *Linked Data for Information Extraction (LD4IE) Workshop, ISWC*. CEUR, 2013.

[19] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovic, and R. Studer. Transforming arbitrary tables into logical form with TARTAR. *Data Knowl. Eng.*, 60(3):567–595, 2007.

[20] Z. Syed, T. Finin, V. Mulwad, and A. Joshi. Exploiting a Web of Semantic Data for Interpreting Tables. In *WebSci10*, Raleigh NC, USA, April 26–27th 2010.

[21] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the Web. *PVLDB*, 4(9):528–538, June 2011.

[22] Y. Wang and J. Hu. A machine learning based approach for table detection on the web. In *WWW*, pages 242–250, New York, NY, USA, 2002. ACM.

[23] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*. Data Mining, the Morgan Kaufmann Ser. in Data Management Systems Series. Elsevier Science, 2011.

[24] M. Yoshida, K. Torisawa, and J. Tsujii. A method to integrate tables of the World Wide Web. In *Workshop on Web Document Analysis (WDA)*, pages 31–34, 2001.

[25] S. Zwicklbauer, C. Einsiedler, M. Granitzer, and C. Seifert. Towards disambiguating Web tables. In *ISWC (P&D)*, pages 205–208, 2013.