A Holistic Approach for Semantic Interpretation of Relational Web tables

Samia Knani, Nadia Yacoubi Ayadi

A Holistic Approach for Semantic Interpretation of Relational Web tables

Samia Knani and Nadia Yacoubi Ayadi
SMART Lab, Institut Supérieur De Gestion De Tunis, Université de Tunis
Tunis, Tunisia
samia.knani777@gmail.com
nadia.yacoubi.ayadi@gmail.com

ABSTRACT

The Web contains vast amounts of semi-structured data in the form of HTML tables found on Web pages which may serve for various applications. One prominent application, which is often referred to Semantic Table Interpretation, is to exploit the semantics of a widely recognized knowledge bases (KB) by matching tabular data, including column headers and cell contents, to semantically rich descriptions of classes, entities and properties in Web KBs. In this paper, we focus on relational tables which are valuable sources of facts about real-world entities (persons, locations, organizations, etc.) and we propose a robust and efficient approach for bridging the gap between millions of Web tables and large-scale Knowledge graphs such as DBpedia. Our approach is holistic and fully unsupervised for semantic interpretation of Web tables based on the DBpedia Knowledge graph. Our approach covers three phases that heavily rely on word and entity pre-trained embeddings to uncover semantics of Web tables. Our experimental evaluation is conducted using the T2D gold standard corpus. Our results are very promising compared to several existing approaches of annotation in web tables.

CCS CONCEPTS

• **Information systems → Data extraction and integration**; **Entity resolution**.

KEYWORDS

Semantic Table Interpretation, Entity Linking, Semantic embeddings

ACM Reference Format:
Samia Knani and Nadia Yacoubi Ayadi. 2020. A Holistic Approach for Semantic Interpretation of Relational Web tables. In *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics (WIMS'20). ACM, Biarritz, France.* . ACM, New York, NY, USA, 10 pages.

978-1-4503-7542-9/20/06...15.00
WIMS '20, June 30–July 03, 2020,

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
DOI: 10.1145/3405962.3405986

1 INTRODUCTION

Nowadays, vast amounts of information is available in structured forms like spreadsheets, database relations, and tables; all embedded in HTML Web documents. Among all of these semi-structured data sources, Web tables are considered as valuable sources of high-quality relational data. The Web contains approximately more than 154 million HTML tables of relational data [2], Wikipedia only containing 3.2 million high-quality tables. Recently, extracting the semantics of Web tables to produce machine understandable knowledge has become an active area of research [1, 6, 11, 27]. Among the different table types available on the Web, relational tables (also referred to as genuine tables) are of special interest. They describe a set of entities (such as persons, organizations, locations, etc.) along with their attributes. However, unlike tables in relational databases, these relationships are not made explicit in web tables; uncovering them is one of the main research challenges. The uncovered semantics can be leveraged in various applications, including table search[21], knowledge base construction, population[29] and completion, and table completion [28]. Fortunately, a large number of Knowledge Graphs have been developed rapidly during the last ten years as multi-relational structured sources of knowledge. Knowledge Graphs (KG) [22] have widespread applications in information retrieval, text mining, and natural language processing. Cross-domain knowledge bases such as DBpedia[14], YAGO [17], or the Google Knowledge Graph [25] are used as background knowledge in a growing range of applications including the Semantic Interpretation of Web tables [1, 6, 8, 27, 31]. However, in order to achieve this purpose, a number of tasks need to be implemented:

- The **Column Type Annotation** (CTA) task that identifies the most likely semantic type(s) in a given KG for the core column in a Web table.
- **Entity linking (EL)** is a key step towards uncovering semantics in Web tables by recognizing and disambiguating specific entities (such as persons, organizations, locations, etc.) and linking them to entities in Knowledge graphs.
- **Columns Predicate Annotation (CPA)** refers to the task of associating a pair of columns in a table with the relation that holds between them and/or extracting relationship information from tabular data and representing them in a structured format such as RDF triple (subject-predicate-object).

While large number of existing approaches focus on the task of EL in Web tables [1, 6, 16]. Few approaches focus on Column Type Annotation [11] and Relation extraction in Web tables [5, 8] even if they are crucial tasks for several applications such as Knowledge Base enrichment, completion and refinement. Although, semantic interpretation of Web tables is a crucial and complex process for

Rank	Title	Year	Director's
1	The godfather	1972	Francis Ford Coppola
2	Citizen Kane	1941	Orson Welles
3	Casablanca	1942	Michael Curtiz
4	Lawrence of Arabia	1962	David Lean
5	dr. Strangelove or: how I learned to stop worrying and love the bomb	1964	Stanley Kubrick
6	Apocalypse now	1979	Francis ford Coppola
7	The godfather: part ii	1974	Francis ford Coppola

Table 1: An example of a Web table describing movie titles

many reasons: (i) The types of the entities described in a table are not known in advance, and the entities described may correspond to more than one type in the target of KG; however, the poor context offered by Web tables in comparison with texts makes the entity disambiguation in Web tables crucial. (ii) The size of both KGs and Web tables corpora is extremely large; (iii) KGs are not complete enough to cover all the entities of Web tables and novel entity discovery in Web tables is challenging [29].

In this paper, we propose a holistic approach for Web tables interpretation (annotation) by uncovering their semantics based on to the DBpedia knowledge graph as background knowledge. Thus, we propose a robust and efficient approach for bridging the gap between millions of Web tables and large-scale Knowledge graphs such as DBpedia. The main contributions of our approach are:

- An unsupervised and holistic approach for semantic table interpretation that covers all key tasks of Column Type Annotation, Entity Linking and Columns Predicate Annotation.
- We rely on Dbpedia knowledge graph as background knowledge for semantic interpretation of Web tables, but our approach can be generalised to any cross-domain knowledge graphs.

2 MOTIVATING EXAMPLE

In this section, we provide an example to explain the problem of table-to-KB matching. Table 1 serves as an example of a movies web table that contains a list of films with their titles, release years and director's, as well as a ranking attribute. The header row (in gray color), gives the attributes labels of the described entities. Each row in the table describes a real-world entity (e.g., the second row describes The *godfather* film), and each column contains the value of the corresponding property, e.g., ("Year",1972), ("Director", *Francis Ford Coppola*).

For humans, understanding the web table content is often an easy task, whereas for machines, web tables present a two-dimensional tabular structure including numeric and string values without any indication about their implicit relationships. For example in Table 1, the cell mention "Casablanca" can refer to different real-world entities such as a city in Morocco, an American film, or an organisation. Same case for "The godfather" mention which refer to either a Book or a Film. The entities mentioned in a web table often differ in their descriptions from the entities in the knowledge graph. Therefore, it would be better to use semantic similarity instead of string similarity to correctly annotate the web table. For example, the mention in row 6 in Table 1 has the name "dr. strangelove or: how i learned to stop worrying and love the bomb" which is different from the entity surface form as described by dbpedia ("Dr._Strangelove").

Moreover, detecting the column type is an important task for the semantic interpretation of web tables and especially crucial for the task of detecting relationships between columns (third task). However, the header name used as the label column may differ from table to another as well as from the semantic class name in the KB. Another crucial task is to detect the subject column of a Web table which refers to the entity described by the Web table. In our example, the subject column is the second column, covering the titles of films. We describe in the next section the different steps of our approach that covers all of these key tasks.

3 OUR APPROACH

In this Section, we describe our approach for semantic annotation in web tables using the DBpedia Knowledge Graph. It performs all the three matching tasks continuously: Column Type Annotation (CTA), Entity Linking in Web tables (EL) and Columns Predicate Column (CPA) tasks. Our approach exploits mainly vector representations of words and entities on all three matching tasks and based on collective strategy which combines local and global features to infer entity linking decisions.

3.1 Column Type Annotation

Column Type Annotation (CTA) aims to associate to each column table a semantic type of the KB. In our approach, it is the task of matching the dominant type of cells in a target column to a recognized knowledge graph concept. For example, the column 2 of Table 1 is annotated with $dbo:Film$ and $dbo:work$, two classes of Dbpedia Ontology. The CTA tasks is crucial forthe remaining two table annotation tasks, namely entity linking (section 3.2) and Columns Predicate Annotation (3.3).

In our work, we perform the CTA task by combining knowledge base lookup and word embeddings representation. Hence, our CTA task combines several features extracted from the context of the Web. CTA Features cover table metadata such as table names, header of columns and content cells (the set of mentions in a given column of a Web table). First, we rely on a knowledge graph lookup method to associate to each mention of a given column a set of candidate classes collected from DBpedia ontology in our case. After That, candidates classes are ranked according to a majority vote.

For each candidate class, we calculate a confidence score depending on three features: a column context that consists of all mentions in the target column, the value of the column header name, and the majority vote score estimated based on the frequency of occurrences of each candidate class. We combine frequency, lexical and semantic features in the following simple formula:

$$Conf\_Score_c = \alpha * \left(\frac{freq(c)}{nb_C}\right) + \beta * CC_{score} + \gamma * lev(hn, clabel) \quad (1)$$

Where $\frac{freq(c)}{nb_C}$ is the majority vote of each candidate class determined by the frequency of this class divided by the total number of candidate classes, lev is the $levenshstein$ similarity that represent

the syntactic similarity between both column header name and DBpedia ontology class label and CC_{score} is the semantic similarity between the column header name (hn) (if it exists) and the candidate class label ($clabel$).

$$\mathrm{CC}_{score} = \begin{cases} Cos\_Sim(hn, clabel) & \text{if header is available} \\ Cos\_Sim(ml, clabel) & \text{else.} \end{cases} \quad (2)$$

Voting with majority can contribute to a highly confident prediction of CTA. Referring to the example of the table 1, the KB lookup method for the mention "Casablanca" should return $dbo : city$ DBpedia ontology class as semantic type instead or in addition to $dbo : movie$ class. Nonetheless, the $dbo : city$ concept will be eliminated by majority vote score according to our confidence score. Note that the semantic similarity determined by the cosine similarity between both vector representations of the column header cell and the candidate class. However, only relying on the header cell results is insufficient because the header is not always available and arbitrary terms can be used. Therefore, if there are no headers, CC_{score} calculated using the cosine similarity between the mention label (ml) and the class surface name($clabel$), after representing each cell value in the target column with a vector using word embeddings. Finally, the candidate class with the highest score will be selected as the best column type.

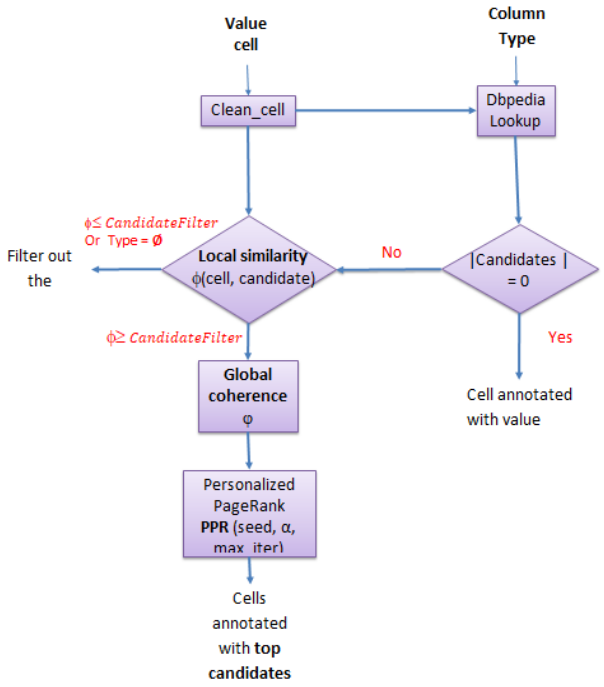

Figure 1: Approach workflow

3.2 Entity Linking

The Entity Linking (EL) task aims to match each mention of a table cell with KB entities, it includes a step that associate to table mention a set of KB candidate entities and a step that disambiguates entities by combining a graph-based collective strategy and distributional vector representations (a.k.a. embeddings) to select the most suitable entities. Figure 1 summarizes the different steps of our EL task.

3.2.1 Candidates generation. The EL task starts with a candidate generation step. Thus, for each potential mention in a table cell, we need to determine an initial set of candidate entities from the given KB. Then, we apply a feature-based ranking procedure based on contextual and string similarities calculated between each mention m_i and each entity in the candidate set E_{m_i}, in order to filter the most similar candidates and reduce the number of candidates as much as possible. Hence, we compute a score for each candidate entity based on the following three similarity values:

- **Semantic similarity** between a mention m_i and an entity $e_{i,j}$ calculated as the cosine distance between their vector representations as word and entity embeddings which are n-dimensional vectors of words representing entities in the target KB.
- **Contextual information** between mention context and the description of the candidate entity extracted from the target knowledge base, the context of a mention m_i represents all the mentions that exist in the same row and column of the target mention.
- **String similarity** represents the syntactic similarity between the string value of the mention m and the RDFS:LABEL value of the entity $e_{i,j}$.

Thus, we keep only the candidate entities having a similarity score exceeds the $CandidateFilter$ threshold. Once the candidate entities have been filtered using the similarity metrics, a set of candidate entities is associated for each mention m_i. The next step is the collective strategy of disambiguation and aims to select the most suitable entity for each mention.

3.2.2 Collective Strategy for Entity Disambiguation. It is at this stage that we describe how to choose an entity from the candidate set as the referent entity for a given mention in a Web table. Our proposed Entity Disambiguation approach is similar to that proposed in [32] which is a collective EL approach for text disambiguation. Therefore, in our approach, we examine the relatedness of entities using a coherence Graph such as candidate entities as vertices and edges reflecting the semantic similarity between each pair of entities that correspond to two different mentions in a Web Table i.e., there is no edge between the candidate entities of the same mention. The intuition behind this approach is that entities in the same column should be closed in the embedding space as they share semantic characteristics. In the disambiguation graph, there are only weighted directed edges between the nodes (entities). These edges have the following format $(e_1; e_2; etp(e_1; e_2))$ where $etp(e_1; e_2)$ represents Entity Transition Probability (ETP) which describes the likelihood to walk from node e_1 to node e_2. In our case, we use the cosine similarity to measure the ETP between two non-zero d-dimensional vectors corresponding to the target entities (e_1, e_2). Cosine Similarity is one of the most frequently used

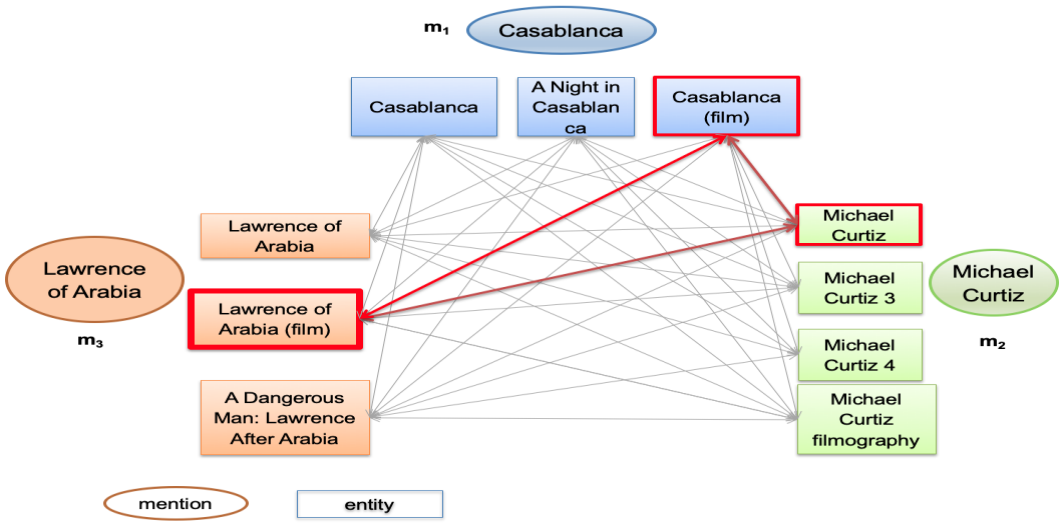

Figure 2: An example of an Entity-Entity disambiguation graph where both "Casablanca" and "Lawrence of Arabia" mentions have three possible candidates, while "Michael Curtiz" mention has four possible candidates. The weight $\varphi(e_i, e_j)$ edges is the cosine similarity between the vectors representations of the candidates entities e_i, e_j as well as the $Sameclass(e_1, e_2)$ function which aim to improve the score value if two target entities e_i and e_j have the same class.

metrics for words similarity [19]. Given two entities vectors $\vec{W_{vec1}}$ and $\vec{W_{vec2}}$, the cosine similarity is calculated as follows:

$$Cos\_Sim(\vec{W_{vec1}}, \vec{W_{vec2}}) = \frac{\vec{W_{vec1}}, \vec{W_{vec2}}}{\|\vec{W_{vec1}}\|, \|\vec{W_{vec2}}\|} \quad (3)$$

where the returned result of Cos_Sim varies between -1 and 1.

The details of the collective disambiguation strategy will be described in the followings steps:

(1) Measure the local similarity score between the mention m_i and the entity $e_{i,j}$ (as described in the candidate generation step)
(2) Estimate the global coherence score between each two entities based on the assumption that two entities are related if their vector embeddings are highly close.
(3) Create a weighted directed graph G=(V, E) where V is the set of candidate entities while E represents the edges between every pair of entities weighted by the ETP value.
(4) Given the weighted graph, we apply a Personalized PageRank (PPR) algorithm to determine a PPR score for each entity candidate.

Local similarity score $\phi(m_i, e_{i,j})$:

We adopt the similarity score as introduced in the candidate generation step as a local similarity score between a mention m_i and the candidate $e_{i,j}$. Therefore, the local similarity score determined using three different metrics, including the contextual similarity $CxtSim(M_i, E_i)$, where M_i is the set of all mentions appearing with m_i in the same row and which represents the context of the target mention mi while E_i is the set of words appearing in the short description of the entity $e_{i,j}$ extracted from the knowledge base which used as a context of the entity $e_{i,j}$. The String Similarity score $SSim(m_i, e_{i,j})$ whereas we apply two different approaches to calculate $SSim(m_i, e_{i,j})$ which are the levenshtein distance and the jaccard similarity, and finally the semantic similarity $SemSim(m_i, e_{i,j})$ between the candidate entity and the mention.

$$\phi(m_i, e_{i,j}) = \alpha * CxtSim(M_i, E_i) + \beta * SemSim(m_i, e_{i,j}) + \lambda * SSim(m_i, e_{i,j}) \quad (4)$$

We exploit the cosine similarity measure to estimate the Semantic similarity $SemSim(m_i, e_{i,j})$ between the mention m_i and the entity $e_{i,j}$ but here we adopt both word and entity embeddings since our experiments shows that joint modeling of words and entities in the same continuous space improves the quality of word and entity embeddings and benefits the task of entity disambiguation. Let T denote the target token where T="Saudi Arabia", we can consider T either as a set of two different words (e.g., Saudi, Arabia) or as an ENTITY (e.g., [Saudi_Arabia]).

Global coherence score (e_i , e_j):

Several previous studies in EL exploit the relatedness measure between the entities to generate a disambiguation context. Thus,in comparison with previous work, in our case, we use both word and entity embeddings similarities because the integration of entity embedding to estimate the semantic coherence between the concepts proved effective for entity disambiguation in recent works [6].

$$\varphi(e_i, e_j) = a \times [Sim(emb(e_1), emb(e_2))] + (1-a) \times Sameclass(e_1, e_2) \quad (5)$$

The global coherence score is computed using a cosine similarity between vector representations of two entities as well as a simple function called $Sameclass(e_1, e_2)$ which aim to improve the score value if the two target entities have the same class.

Collective Disambiguation Strategy

In our approach, we create a weighted and directed graph with edges connecting each two entities refereed to two different mentions. The edges are weighted by the score of the global context matching $\varphi(e_i, e_j)$ which describe the ETP of walking from one node to another. An example of an entity-entity disambiguation graph is illustrated in Figure 2 .

Algorithm 1 Graph-based EL in web tables

Input: N mentions$(m_1, ..., m_N)$ each mention m_i has a list of candidate Entities $CE_{m_i} = [e_1, ..., e_{|CE|}]$
Output: $\Gamma^* = (e_1, ..., e_N)$ with e_i being the entity matching to m_i

Begin
// Local similarity candidate Filter
for $m_i \in M$ *and* $|CE_{m_i}| >= 1$ **do**
  FinalCan=ϕ
  if $localSimilaityScore(m_i, e_i) >= thresholdSim_{m,e}$ *and* $class_{e_i} \neq \phi$ **then**
    FinalCan= FinalCan $+e_i$
  if *FinalCan*== ϕ **then**
    $CE_{m_i} \leftarrow NIL$
  else
    $CE_{m_i} \leftarrow FinalCan$

// Create a Weighted Direct Graph WDG

$nodes = \phi$
for m_i *in* M **do**
  for $e_j \in CE_{m_i}$ **do**
    $nodes \leftarrow nodes + e_j$
WDG=Create_DiGraph(nodes, GlobalSimilarity$(node_{e_i}, node_{e_j})$)

$Matrix = Convert\_to\_Matrix(WDG)$
// Apply Personalized PageRank PPR
for m_i *in* M **do**
  PPR= Personalized_PageRank($Matrix$)
  Rank *entities*
  if $CS_{mi} >= 1$ **then**
    select for each m_i the entity with the highest score
  else
    m_i annotate with NIL

Based on the disambiguation graph, we perform a random walk simulated by a PageRank procedure which aims to encode the structure of the graph in the form of a transition matrix in order to determine a ranking list of relevance scores for each entity candidate. Depending on this list, our approach decides which entity candidate is the correct entity that corresponding to the target mention.

In our approach, we use Personalized PageRank (PPR) [9] which is more general than PageRank version. The personalized PageRank based on two parameters, the jumping constant α and the seed s. In fact, a seed can be considered as either a vertex or a probability distribution on vertices i.e. a probability distribution on a seed node set which serves as the personalization context. On this way, given a weighted direct graph G(V, E), a damping factor α and a set of nodes $S \in V$ denoted as the seed vector $\vec{s}$, the personalized PageRank score PPR_s of an entity c_i in V is defined as follow:

$$PPR_s(c_i) = (1-\alpha)\vec{s} + \alpha \times \sum_{c_j \in C(c_i)} \frac{\varphi(c_i, c_j)}{\sum_{c_k \in C(c_j)} \varphi(c_j, c_k)} PPR_s(c_j) \quad (6)$$

where $\varphi(c_i, c_j)$ is the coherence score between two candidates c_i and c_j calculated using equation 7. $\vec{s}$ is a re-seeding vector, such that if $v_i \in S$, then $\vec{s}[i] = \frac{1}{\|S\|}$ and $\vec{s}[i] = 0$, otherwise. Therefore, rather than leaping to a random node in V with probability α, the random walk jumps to one of the nodes from the set of seed S.

3.3 Columns Predicate Annotation

The structure of a web table inherently provides high quality semantic relations between its columns. However, extracting these relations is a challenging task since it requires the identification of unique structural information of each table. In the perspective of the knowledge base, these relations are considered as properties or entities attributes.
In general, every relational web table has a subject column which is a key of the web table and hence every other column represents a binary relation with the subject column.

3.3.1 *Subject Column Detection (SCD).*

Subject column, also called entity label column or key column, contains the names of the entities described by the web table. Assume that each web table has exactly one subject column, to detect the latter we apply several heuristic as shown in Algorithm 2.

Algorithm 2 Pseudocode to find the subject column

SubjColm= None
// SubjColm is the most unique column in the table
for *c in C* **do**
  if *type(header(c))==str* **then**
    if *len(mention)>=3 and len(mention)<=200* **then**
      if *"name" or "title" in header(c)* **then**
        if *confidenceScore(c) >= 0.3* **then**
          SubjColm= c
      else if *confidenceSc(c) == max(confidenceSc(C))* **then**
        SubjColm= c

First of all, the column needs to be of data type `String` and its cells have an average length between 3 and 200 characters. Since the table header is not always available and that arbitrary terms can be used, we can not rely solely on the header to detect the subject column. To the best of our knowledge, we calculate a confidence score which is defined as the ratio of annotated mentions to

all mentions of a column, reduced by the ratio of non-annotated mentions, e.i. mentions mapped to value "NIL" :

$$confidenceScore(c) = \frac{\#annotatedMentions}{\#totalMentions} \quad (7)$$

The confidence score is important for detecting the correct subject column. It rewards columns with many annotated cell and penalises columns with many "NIL" annotations. Hence, two rules decide which column is selected as subject column: (1) If the column header contains either the term "name" or "title" and the confidence score above 0.3, this column chosen as subject column; (2) otherwise, the column with the highest confidence score is chosen and in case of a tie, the left-most candidate is chosen.
Note that the purpose of SCD is not limited to identify the main class of web tables, but also used by later processes such as columns predicate annotation.

Within this context, we aim to identify the semantic relations of the subject column in a table with other columns (both Named entity and literal-columns) in the same table using the linkable table. In others terms, with the help of EL and CTA matching tasks, we turn to determine which relation might exist between the subject column and the attribute columns in order to set the overall meaning of the table.
In our case, we firstly start by genereating a set of candidate properties using the following three matchers for the columns predicate annotation task:

Outgoing-Properties.
We introduce a subject column based search mechanism for the property selection. Indeed, for each entity in the subject column, we query the DBpedia knowledge base to find all the possible relationships emerging from that entity. we ask for all properties in the DBpedia knowledge base that corresponds to the relations that a subject column expect from the other columns of the table. For example, the subject column c_0 of table in Figure 3 can have the outgoing properties *dbp.areaTotal*, *dbp.averageDepth*, *dbp.elevation*, *dbp.length*, *dbp.location*

Listing 1: SPARQL query used in the Outgoing-Properties extract

```
PREFIX dbr: <http://dbpedia.org/resource/>
select distinct ?predicate ?object where{
%URI% ?predicate ?object .}
```

Incoming-Properties.
Incoming-Properties refers to all properties coming into all entities in the Named Entity column except the subject column, in the perspective of other entities. it define the behavior of an entity in relation to all subject entities. Given e is an entity of a NE-column is derived from the triples containing e as object, denoted by $<?subject, ?predicate, e>$.

Listing 2: SPARQL query used in the Incoming-Properties extract

```
PREFIX dbr: <http://dbpedia.org/resource/>
select distinct ?subject ?predicate where{
?subject ?predicate %URI%.}
```

Extracted-Properties.
Here, we aim to extract all possible relations between a subject column and a NE-column. For this purpose, we query Dbpedia for the properties between all pairs of entities for each row in the primary and secondary columns, $< \%URI1\%, ?predicate, \%URI2\% >$, where $\%URI1\%$ and $\%URI2\%$ are entities in the subject column and the NE-column respectively. As instance, the Extracted-Properties between column c_0 and column c_1 are: *dbp.location*

Listing 3: SPARQL query used in the Extracted-Properties

```
PREFIX dbr: <http://dbpedia.org/resource/>
select distinct ?predicate where{
%URI1% ?predicate %URI2% .}
```

At the end of the properties extraction process a set of candidate properties is generated. In next step, we rank these candidate properties by importance in order to determine the best possible candidate relation between two given columns. To do so, we assign a score for each candidate relation $< subject\_column, property, object\_column, score >$ by utilizing the following metrics:

- **Similarity.** For each candidate in the Outgoing-Properties set, we compute a string similarity(Levenshtein) and a cosine similarity(words embeddings) between the name of the property candidate and the header text of the object column (if exists).
- **Property validity.** We determine the validity of a property by checking whether a candidate property is apparent in both the outgoing-properties and in the incoming-property. If so, a property validity function returns 1 or 0 otherwise. The purpose of this function is to identify one or more valid properties that can link the subject column to the object column.
- **Property confidence function** specifies how likely the relation holds. This function based on the frequency of occurrences of each candidate in the Extracted-Properties. It varies between 0 and 1 while 1 declares that the property of the greatest frequency is most likely to hold.

Finally, we aggregate all metrics listed above to predict a final score S_p for each candidate property, i.e.,

$$S_p = \alpha * Sim(p, h_c) + \beta * V_p + \gamma * Conf_p \quad (8)$$

Where *Sim* include the Levenshtein similarity and the cosine similarity between the header text of a column h_c and the name of a candidate concept p, V_p is the validity function of p and $Conf_p$ is the Property confidence function.
Finally, the candidate property with the highest score S_p is chosen as the best binary relation associating a subject column with a secondary column.

4 EXPERIMENTS

Our matching method is fully implemented in python. We exploit DBpedia as knowledge graph, T2D gold standard (version 1) as datasets of web tables and Wikipedia2vec as a Pre-trained model of Words and Entities Embeddings.

DBpedia[14] is one of the most popular knowledge bases. DBpedia is a very popular knowledge graph that is located in the center of the LOD cloud because it is highly interconnected with many other datasets in the Linked Open Data cloud. Furthermore, The DBpedia knowledge graph presents several advantages compared to the existing knowledge bases: it covers several areas and

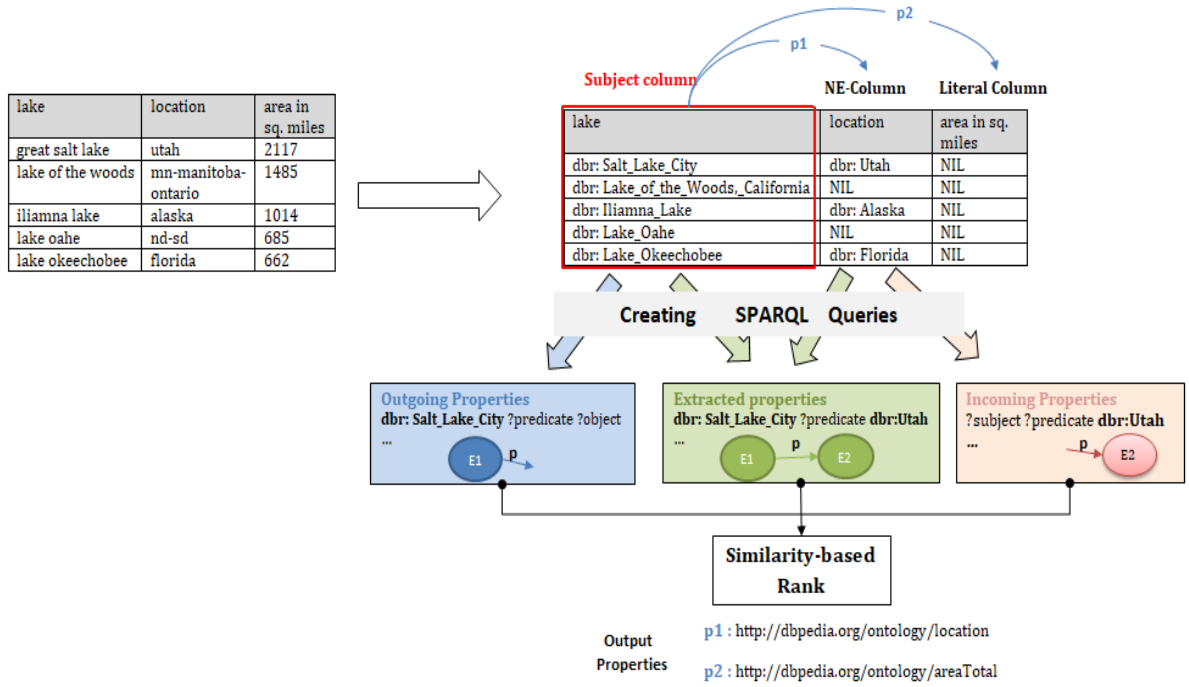

Figure 3: Columns Predicate Annotation

is known as the cross-domain knowledge base; it evolves automatically as Wikipedia changes, and it is truly multilingual. Indeed, DBpedia provides 125 languages versions which describe together 38.3 million things. Especially, the English version of the DBpedia knowledge base describes 4.58 million entities. [1].

The T2D gold standard[2] was developed for several goals. It contains 233 Web tables, with manual annotations of 26,124 entities extracted from the 2014 version of DBpedia. T2D has an average of 123 rows per table, and 4,95 columns per table. It is highly structured, meaning that very few cells are empty in tables. It includes cross-domain tables from multiple websites covering a wide range of challenges (figure 4).

Wikipedia2vec.[3] To the best of our knowledge, we use Wikipedia2Vec, a model for obtaining embeddings of entities and words simultaneously. The strong point of this method is that it based on word2vec and aims to jointly place words and entities in the same continuous vector space. Indeed, similar words and entities are close together in the same vector space [26]. This is what allows us to easily calculate the cosine similarity between any pair of items (entity-entity, word-entity or word-word). However, Wikipedia2Vec has been used in several recent works such as: Entity Linking [4], Named Entity Recognition [12] and Entity Typing [26].

Evaluation Measures. To evaluate the performance of the different tasks of our system, we report the results using standard measures: precision, recall and F-measure.

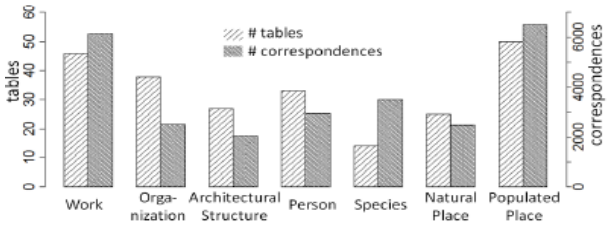

Figure 4: Number of class and entity correspondences per category.

$$Precision = \frac{TP}{(TP + FP)} \qquad Recall = \frac{TP}{(TP + FN)}$$

$$F1 = \frac{(2 * Precision * Recall)}{(Precision + Recall)}$$

4.1 Experiment Settings

Our approach offers several parameters to improve the overall results. In what follows, we will only mention those who have the most impact on results.

Semantic Embeddings. In terms of word2Vec, we use a feature space of d = 300 dimensions.

[1] http://wiki.dbpedia.org/about
[2] http://webdatacommons.org/ webtables/goldstandard.html
[3] https://wikipedia2vec.github.io/wikipedia2vec/

Table 2: Results of the CTA task using different features

Method	Precision	Recall	F-measure
String Similarity	0.65	0.34	0.45
frequency-based	0.81	0.78	0.79
Embedding-baseline	0.83	0.81	0.82
All	0.90	0.85	0.87

Table 3: Results of the three matching tasks

Task	Precision	Recall	F-measure
CTA	0.84	0.91	0.87
EL	0.87	0.86	0.87
CPA	0.69	0.71	0.70

Starting by the CTA task, since the frequency-based and the embeddings method are the most important features for the class matching task as illusrated in table 2: Considering only the string similarity (Levenshtein) between the candidate class and the column header to find the the correct column class, the precision is 0.65 and the recall 0.34. Meaning that for only fewer than half of the tables, the correct class is assigned, due to the Levenshtein similarity that compares the sets of tokens from two strings or applies a character-based comparison. On the other hand, by only using the frequency-based matcher to find the correct column class, an F-measure of 0.79 can be achieved. In order to see the importance of applying methods that rely on context features,we evaluate the embeddings method independently of the others, an F-measure of 0.82 is obtained which is greater than the result of the frequency-based and the string similarity methods. we can therefore conclude that the use of embeddings is the most important feature for the class matching task and hence we adjust α to 0.4 and β to 0.5 according to the best averaged F1 values throughout the experiments.

For the EL task, two main parameters have a strong influence on the results: Similarity threshold and Page rank.

- **Similarity threshold:** we apply a *CandidateFilter* threshold to keep all candidates above a configurable threshold as final candidates i.e., if the score of the entity is below the given threshold, the entity will be removed from the list of candidates. Indeed, our experiences shows that the EL task can achieve high results only if the similarity threshold is equal to 0.4.
- **Personalized PageRank:** we perform Personalized Page rank on an entity-entity graph and we use the output scores for disambiguation. We choose iter=50 iterations because the results do not change with more iterations. In terms of the PPR jumping constant α and the seed s, we choose α = 0.75 and s= 4.

After evaluating EL task, we now turn to evaluating the CPA task. For this task, we use the same evaluation metrics as for CTA and EL above. We mention the candidate filter threshold γ which has the greatest impact on the final result. Indeed, we filter the candidate properties that have similarity scores less than γ. We select the value γ = 0.6 according to the best averaged F1 values throughout the experiments.

4.2 Overall Results

Table 2 reports the overall results of our system for three matching tasks in order to map the ambiguous tables from T2D to DBpedia KG. Particularly, For the Subject Column Detection (SCD), an F-measure of 0.97 is observed, this means that for almost all tables the correct subject column is determined. This is an important finding since an incorrect subject column mostly results in incorrect correspondence for the properties correspondence task. For the class matching task, an F-measure of 0.86 can be be achieved, followed by the performance for entities with 0.86 and properties with 0.70.

4.3 Comparison with State-of-the-Art

In this section, we aim to estimate the performance of our system by comparing our results with other recent matching systems like DBpedia Lookup, (Efthymiou et al., 2017) and T2K Match. Table 3 summarise the comparison between these systems tested on the T2D corpus.
As illustrated in table 3, our EL method outperforms the DBpedia lookup, (Efthymiou et al., 2017) and the T2K match method, with an increase of 0.1 in F-measure over the Dbpedia lookup, 0.04 over T2K Match and an improvement of only 0.02 over the (Efthymiou et al., 2017). In addition, our CPA method achieves an F-measure of 0.7 similar to that of the T2K Match method. However, our CTA result is lower than the T2K correspondence result because we perform the class matching task at the beginning of our system instead of executing it after the EL task. Hence, we can say that our system considered to be one of the best web table annotation system.

5 RELATED WORK

In this section, we first present the semantic embeddings and then we discuss recently proposed methods in the area of semantic table interpretation.

Semantic Embeddings. Despite the performance of cosine similarity in semantic measurement, the quality of the results may differ depending on the method used to determine the vector representations of terms. Indeed, there are a lot of techniques used to mapped terms to n-dimensional vectors like One-hot vector representations which generally do not provide any information on the meaning of term [3] and TF-IDF (term frequency- inverse document frequency) technique where TF is how often the term appears in the document whereas IDF is the number of documents in which the term appears [24]. However, word to be one of the most popular and beneficial methods in several works [6, 23]. With Word2Vec framework, words are represented in a continuous vector space by associating related words to relatively close points to display their semantic similarity. In order to learn word embeddings, [18] proposed two popular models which are Continuous Bag of Words (CBOW) that used to predict a word using a set of words constituting its surrounding context and Skip-gram that used to predict the context of each word.

Table 4: Comparison with other matching systems over T2D gold standard

	CTA			EL			CPA		
System	Precision	Recall	F1-measure	Precision	Recall	F1-measure	Precision	Recall	F1-measure
DBpedia Lookup	-	-	-	0.79	0.73	0.76	-	-	-
T2K Match	0.94	0.94	**0.94**	0.90	0.76	0.82	0.77	0.65	**0.70**
[Efthymiou et al., 2017]	-	-	-	0.87	0.83	0.85	-	-	-
Our method	0.84	0.91	0.87	0.87	0.86	**0.87**	0.69	**0.71**	**0.70**

Table interpretation. covers methods such as column type identification [5, 7, 15, 31], entity linking [1, 6, 23, 29, 31] and relation extraction [23, 31] which aim to annotating web tables with the help of existing knowledge bases and to make tabular data readable by machines.

A majority of recent works based their approaches on keyword lookup method. In other words, these approaches search for a particular keyword using a lookup method with a public API of the target KG, like Ritze et al. [23] which use the DBpedia lookup service, Zhang [27] use the Freebase lookup service while Efthymiou et al. [6] use the wikidata lookup service or a manually created search index [30]. Generally these approaches are limited to exact matches and regular expression queries. T2K Match [23] annotates web tables by associating web tables to classes, rows to entities and columns to properties from DBpedia knowledge graph. This lookup-based method begins with a candidate generation phase, and then uses the entity types, relationships, and terms common in descriptions of candidate matches to refine or even extend candidate matches. TableMiner [27] an incremental approach characterized by an efficient boot-strapping annotation method used to effectively matching columns and cells to entities in the knowledge graph. Efthymiou et al.[6] adopt an unsupervised hybrid annotation method based on three alternative approaches for matching entities whose contextual information may vary from poor (in Web tables) to rich (in KBs). The first is a lookup-based method, which relies on the minimal entity context from the tables to discover correspondences to the knowledge base. A second method exploits a vectorial representation of the rich entity context in a knowledge base to identity the most relevant subset of entities in web tables. The third method is based on ontology matching, and exploits schematic and instance information of entities available both in a knowledge base and in a web table. MTab [20] a probability-based method which combines the voting algorithm and the probability models to solve two major problems: (a) DBpedia lookup does not usually get relevance entities for non-English queries and (b) mapping cell values to corresponding values in a KG is less effective because the corresponding value in KG is rarely equal with a query value. This deep learning-based method optimizes the relations between mentions based on mention-entity hyperlink count statistic.

Another study of web table interpretation [10] with the objective of overcoming the problem of matching and aggregating tables by giving meaning to entities and quantities in web tables. In this study, authors map the table elements of table headers, entity tables cells, and numeric table cells to different knowledge bases.

Recently, works has tended to apply supervised learning to the semantic table annotation process where supervised task infers outputs data from labeled training data. For instance, TabEL a supervised machine learning approach for Semantic table interpretation proposed by [1]. This system takes the problem of annotating tables as an EL task which aims to match each value cell in the table to the concept of ontology. Furthermore, ColNet[5] a supervised neural network based CTA framework which is able to integrate KB reasoning and lookup with machine learning and can automatically train Convolutional Neural Networks for prediction.

A lot of deep learning methods have been implemented for web table interpretation in recent years like [13]; an end-to-end entity linking method that treats relations between mentions as latent variables. It induce the relations without any supervision while optimizing the entity-linking system in an end-to-end fashion.

6 CONCLUSION

In this paper, we present our unsupervised approach for semantic interpretation of web tables by solving three matching tasks namely Column Type Annotation (CTA), Entity Linking (EL) and Columns Predicate Annotation (CPA) using DBpedia KB. More precisely, CTA, aims to match table columns with widely recognized concepts like semantic classes of a KB. We mainly combined two baselines which are semantic embeddings and majority voting. The EL task builds a weighted directed graph to find the best candidate mapping between a mention in the Web table and the corresponding entity in the KB. This collective strategy based on two main features concerning the local similarity between mentions and entities as well as the global coherence between entities using words and entities embeddings representations. The last task, CPA, presents a novel method to analyse the behaviour between row entities to collectively deduce the possible relations between table columns. For this task, we categorize the properties generated between columns into three groups: (1) Outgoing-Properties, (2) Incoming-Properties, and (3) Extracted-Properties.

To evaluate the performance of our system, we exploit a corpus of 233 web tables extracted from T2D gold standard which captures correspondences for all the three matching tasks between tables and DBpedia KG. By observing the results of the evaluation, our system obtained relatively high quality results for the EL task and acceptable results for the other two tasks compared to state-of-the-art Web table annotation methods. For future work, we aim to improve our approach especially the CTA task. We also plan to extend our proposed method to other related tasks such as the discovery of new entities in Web tables for KB population, augmentation and refinement.

REFERENCES

[1] Chandra Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: Entity Linking in Web Tables. In *Proceedings of the 14th International Semantic Web Conference (ISWC'15)*. 425–441.

[2] Michael J. Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. 2008. Uncovering the Relational Web. In *Proceedings of the 11th International Workshop on Web and Databases (WebDB 2008)*. Vancouver, Canada.

[3] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. 2018. Similarity encoding for learning with dirty categorical variables. *Machine Learning* 107 (2018), 1477–1494.

[4] Haotian Chen, Sahil Wadhwa, Xi David Li, and Andrej Zukov Gregoric. 2019. YELM: End-to-End Contextualized Entity Linking. *ArXiv* abs/1911.03834 (2019).

[5] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles A. Sutton. 2018. ColNet: Embedding the Semantics of Web Tables for Column Type Prediction. *CoRR* abs/1811.01304 (2018). arXiv:1811.01304 http://arxiv.org/abs/1811.01304

[6] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. 2017. Matching Web Tables with Knowledge Base Entities: From Entity Lookups to Entity Embeddings. In *International Semantic Web Conference, 260–277*. Springer.

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

[8] Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. TableNet: An Approach for Determining Fine-grained Relations for Wikipedia Tables. *WWW '19: The Web Conference on The World Wide Web Conference WWW 2019* (2019), 2736–2742.

[9] Shengyu Huang, Xinsheng Li, K. Selçuk Candan, and Maria Luisa Sapino. 2014. "Can you really trust that seed?": Reducing the impact of seed noise in personalized PageRank. *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)* (2014), 216–223.

[10] Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. 2016. Making Sense of Entities and Quantities in Web Tables. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*.

[11] San Kim, Guoliang Li, Jianhua Feng, and Kaiyu Li. 2018. Web Table Understanding by Collective Inference. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)* (2018), 217–226.

[12] Alicia Lara-Clares and Ana M. García-Serrano. 2019. LSI2_UNED at eHealth-KD Challenge 2019: A Few-shot Learning Model for Knowledge Discovery from eHealth Documents. In *IberLEF@SEPLN*.

[13] Phong Le and Ivan Titov. 2018. Improving Entity Linking by Modeling Latent Relations between Mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 1595–1604. https://doi.org/10.18653/v1/P18-1148

[14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6 (2015), 167–195.

[15] Oliver Lehmberg and Christian Bizer. 2016. Web table column categorisation and profiling. In *Proceedings of the 19th International Workshop on Web and Databases (WebDB '16)*. 1–7. https://doi.org/10.1145/2932194.2932198

[16] Xusheng Luo, Kangqi Luo, Xianyang Chen, and Kenny Q. Zhu. 2018. Cross-Lingual Entity Linking for Web Tables. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 362–369. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16490

[17] Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. 2014. A knowledge base from multilingual Wikipedias-YAGO 3 Une base de connaissances des Wikipédias plurilingues – YAGO 3.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems* abs/1310.4546 (10 2013). arXiv:1310.4546 http://arxiv.org/abs/1310.4546

[19] Jose G. Moreno, Romaric Besançon, Romain Beaumont, Eva D'hondt, Anne-Laure Ligozat, Sophie Rosset, Xavier Tannier, and Brigitte Grau. 2017. Combining Word and Entity Embeddings for Entity Linking. In *The Semantic Web (ESWC)*. Springer International Publishing, Cham, 337–352.

[20] Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. 2019. MTab: Matching Tabular Data to Knowledge Graph with Probability Models. In *OM@ISWC*.

[21] T. T. Nguyen, Q. V. H. Nguyen, M. Weidlich, and K. Aberer. 2015. Result selection and summarization for Web Table search. In *2015 IEEE 31st International Conference on Data Engineering*. 231–242.

[22] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8(3) (2017), 489–508.

[23] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching HTML Tables to DBpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics WIMS '15*. 1–6. https://doi.org/10.1145/2797115.2797118

[24] Stephen E. Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation* 60 (2004), 503–520.

[25] Ahmet Uyar and Farouk Musa Aliyu. 2015. Evaluating search features of Google Knowledge Graph and Bing Satori: Entity types, list searches and query interfaces. *Online Information Review* 39 (2015), 197–213.

[26] Ikuya Yamada, Hiroyuki Shindo, and Yoshiyasu Takefuji. 2018. Representation Learning of Entities and Documents from Knowledge Base Descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 190–201. https://www.aclweb.org/anthology/C18-1016

[27] Zhang and Ziqi. 2014. Towards Efficient and Effective Semantic Table Interpretation. In *The Semantic Web – ISWC 2014*, Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble (Eds.). Springer International Publishing, Cham, 487–502.

[28] Shuo Zhang and Krisztian Balog. 2019. Auto-completion for Data Cells in Relational Tables. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019).

[29] Shuo Zhang, Edgar Meij, Krisztian Balog, and Ridho Reinanda. 2020. Novel Entity Discovery from Web Tables. *Proceedings of The Web Conference 2020 (WWW '20)* (Apr 2020), 11. https://doi.org/10.1145/3366423.3380205

[30] Xiaolu Zhang, Yueguo Chen, Jinchuan Chen, Xiaoyong Du, and Lei Zou. 2013. Mapping Entity-Attribute Web Tables to Web-Scale Knowledge Bases. In *Database Systems for Advanced Applications*, Weiyi Meng, Ling Feng, Stéphane Bressan, Werner Winiwarter, and Wei Song (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 108–122.

[31] Ziqi Zhang. 2017. Effective and efficient Semantic Table Interpretation using TableMiner+. *Semantic Web, vol. 8, no. 6, pp. 921-957* (2017). https://doi.org/10.3233/SW-160242

[32] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2016. Robust and Collective Entity Disambiguation through Semantic Embeddings. In *SIGIR '16*. 425–434. https://doi.org/10.1145/2911451.2911535