TableNet: An Approach for Determining Fine-grained Relations for Wikipedia Tables

Besnik Fetahu[1], Avishek Anand[1], Maria Koutraki[1,2]

[1]L3S Research Center, Leibniz University of Hannover, Hannover, Germany

[2]FIZ-Karlsruhe Leibniz Institute, Karlsruhe Institute of Technology, Karlsruhe, Germany

{fetahu,anand,koutraki}@L3S.de

ABSTRACT

Wikipedia tables represent an important resource, where information is organized w.r.t table schemas consisting of columns. In turn each column, may contain *instance values* that point to other Wikipedia articles or *primitive values* (e.g. numbers, strings etc.).

In this work, we focus on the problem of interlinking Wikipedia tables for two types of table relations: *equivalent* and *subPartOf*. Through such relations, we can further harness semantically related information by accessing related tables or facts therein. Determining the relation type of a table pair is not trivial, as it is dependent on the schemas, the values therein, and the semantic overlap of the cell values in the corresponding tables.

We propose *TableNet*, an approach that constructs a knowledge graph of interlinked tables with subPartOf and equivalent relations. *TableNet* consists of two main steps: (i) for any *source* table we provide an *efficient* algorithm to find all *candidate* related tables with *high coverage*, and (ii) a neural based approach, which takes into account the table schemas, and the corresponding table data, we determine with *high accuracy* the table relation for a table pair.

We perform an extensive experimental evaluation on the entire Wikipedia with more than 3.2 million tables. We show that with more than 88% we retain relevant candidate tables pairs for alignment. Consequentially, with an accuracy of 90% we are able to align tables with subPartOf or equivalent relations. Comparisons with existing competitors show that TableNet has superior performance in terms of coverage and alignment accuracy.

1 INTRODUCTION

Wikipedia has emerged as one of the most reputable sources on the internet for a wide range of tasks, from question answering [8] or relation extraction [19]. One of the most notable uses of Wikipedia is on knowledge base construction.

Well known knowledge bases like DBpedia [2] or YAGO [28] are almost exclusively built with information coming from Wikipedia's infoboxes. Infoboxes have several advantages as they adhere to pre-defined templates and contain factual information (e.g. *bornIn* facts). However, they are sparse and the information they cover is very narrow. For most of the application use cases of Wikipedia, availability of factual information is a fundamental requirement.

Wikipedia tables on the other hand are in abundance. The current snapshot of Wikipedia contains more than 3.23M tables from more than 520k Wikipedia articles. Tables are rich with factual information for a wide range of topics. Thus far, their use has been limited, despite them covering a broad domain of factual information that can be used to answer complex queries. For instance, for complex queries like *"Award winning movies of horror Genre?"* the answer can be found from facts contained in *multiple tables* in Wikipedia. However, question answering systems [1] built upon knowledge base facts, in most cases they will not be able to provide an answer or will provide an *incomplete* answer.

The sparsity or lack of factual information from infoboxes can easily be remedied by additionally considering facts that come from Wikipedia tables. A rough estimate reveals that we can generate more than hundreds of millions of additional facts that can be converted into knowledge base triples [20]. This amount in reality is much higher, if we allow for tables to be aligned. That is, currently, tables are seen in isolation, and *semantically* related tables are not interlinked (i.e. equivalent table relations). Table alignments would allow to access tables fulfilling a specific criteria (e.g. *"List of All Movies"* by different Producers). Additionally, relations that can semantically describe tables as *supersets or subsets* (i.e. subPartOf relations) in terms of classic database *projection* or *selection* functions are missing (cf. Figure 1), which would enable queries to access semantically dependent tables (i.e., *"List of Award-Winning Movies"* and *"List of All Movies"* from a Producer). The presence of such fine-grained relations opens up for opportunities that can be used in question answering, knowledge base construction, and inferences of other facts from the facts that reside in equivalent or subPartOf aligned tables.

Determining the fine-grained table relations is not a trivial task. Table relations are dependent on the semantics of the columns (e.g. a column containing instance values of type Country), the context in which the column appears (e.g. *"Name"* can be an ambiguous column and it can only be disambiguated through other columns in a table schema), cell values etc. Furthermore, not all columns are important for determining the relation between two tables [27]. Finally, to be able to establish relations amongst all relevant table pairs, requires for efficient approaches that avoid exhaustive computations between all table pairs that can be cumbersome given the extent of tables in Wikipedia.

In this aspect, related work has focused mainly on table retrieval scenarios. The Google Fusion project [7, 27] retrieves top-k tables, where the query is a table and the notion of relatedness is in terms of table schemata (specifically subject columns). Recent work [30] focuses on ad-hoc table retrieval from keyword search. There are two main issues that are not addressed by related work: (i) top-k retrieval does not provide guarantees in terms of coverage, and (ii) the notion of relevance is in terms of *keyword queries*, and there is no distinction between the different *relation types*, specifically `equivalent` and `subPartOf` relations.

We propose *TableNet*, an approach with the goal of aligning tables with `equivalent` and `subPartOf` fine-grained relations. Our goal is to ensure that for any table, with high coverage we can find candidate tables for alignment, and with high accuracy determine the relation type for a table pair. We distinguish between two main steps: (i) efficient and high coverage table candidate generation for alignment, and (ii) relation type prediction by leveraging table schemas and values therein.

We perform an extensive evaluation of *TableNet* on the entire English Wikipedia with more than 3.2 million tables. Through our proposed approach we are able to retain table pairs that have a relation with a high coverage of 88%, and correspondingly predict the type of the relation with an accuracy of 90%.

We make the following contributions in constructing TableNet:
- we formally define the problem of fine-grained table alignment;
- we model tables with fine-grained information relying on column descriptions, instance values, and types and additionally take into account the context in which the columns appear for table alignment;
- a large ground-truth for table alignment with coverage guarantees with more than 17k table pairs;
- a large scale knowledge graph of aligned tables with more than 3.2 million tables.

2 RELATED WORK

In this section, we review related work, which we differentiate between three main categories that we describe in the following.

Wikipedia Tables. Bhagavatula et al. [4] propose an approach for finding relevant columns from target tables for a given source table, based on which the table pair can be joined. Similarly, in our table candidate generation process we employ graph relatedness measures in order to generate relevant table pairs for alignments. Nonetheless, our objectives are on finding tables for alignment that are semantically similar. As such our criteria for alignment does not adhere to their *join* definition where the objective is to construct a new table as a result of the joined tables. Additionally, for our task of fine-grained alignment, we argue that relying only in specific columns is not sufficient for alignment. The semantics of a table cannot always be defined from a single column, but rather the context in which the column occurs.

Web Tables. The work carried in the project *Google Fusion Tables* [7, 10, 27, 29] represents one of the most significant efforts in providing additional semantics over tables, and to the best of our knowledge, only some of the works carried in this project are most related to our work, against which we provide an optimal comparison [27].

Carafella et al. [7] propose an approach for table extraction from Web pages and additionally provide a ranking mechanism for table retrieval. An additional aspect they consider is the schema auto-completion for some input column, where they recommend other columns that would fit contextually to generate a "complete" schema. Our aim is different here, while we aim at providing more fine-grained representations of columns in a table schema, our goal is to use such information for the task of table alignment.

Das Sarma et al. [27] propose an approach for finding related tables, where as relatedness they consider two cases: (i) *entity complement* and (ii) *schema complement*. For (i), the task is to align tables that have the same table schemas, however, with complementary instances. This case can be seen as applying a *selection* over some table that has the union of instances from both tables. In (ii), the columns of a target table can be used to complement the schema of a source table, with the precondition that the instances (from *subject* columns) are the same in both tables. This case is seen as a *projections* operation over some table with the same *selection* criteria, thus, resulting in the same set of instances.

Our work is related to the case of *entity complement*, where the authors compute the *schema similarity* between two tables in order to decide if a table can be considered for complementing the instances in another table. The similarity of the schemas is considered as a max-weight bipartite matching approach, with weighted established between the column in the disparate schemas, and edge weight being the *string similarity* between columns and jaccard similarity between the column types (established from the values in a column through the `WebIsA` database).

Despite the fact that this approach is unsupervised, we adapt it such that we find the best threshold of the max-weight matching score between two schemas, and consider tables to be either *aligned* or *not-aligned*. We show that our approach outperforms the most closely related work from Google Fusion.

The aforementioned works rely on a structured table representation based on work by Venetis et al. [29]. The columns in a table are labelled based on a `isA` database, which consists of instances and the associated labels mined from a large Web corpus (e.g. *capital cities* for a column containing *Paris, Berlin*, etc.). In our case, we deal with Wikipedia tables in which instances are linked to Wikipedia articles, thus, we opt for using the Wikipedia category taxonomy and the additional information coming from knowledge bases for inferring a structured representation of a table schema, respectively for the individual columns in a table. Wikipedia categories are much richer than the `isA` database used in [29], which is flat, contrary to the categories which represent a taxonomy, thus, allowing us to leverage from coarse to fine grained information about columns.

Table Annotation. Work on table annotation [5, 16] focus specifically on linking cell values with entities, and columns with entity types that best describe the values in a column. We see these works as complementary, in which case we can employ them to further enrich tables where the cell values are not already linked to existing Wikipedia entities.

Another line of work is proposed by Munõz et al.[20]. In this work, Wikipedia tables, specifically table rows are used as an input for generating RDF triples, where the relations between two columns correspond to properties extracted from a target knowledge base like DBpedia. Slightly related to [20] is the work by Ritze

et al. [25], where the authors propose a system called *T2K Match*. T2K matches Web tables into a target knowledge base, that is, the columns are described in terms of classes from the target KB. The works in [20, 25] can be seen as complementary and they could be used as additional information for the table alignment task.

Schema Matching. In the database community, there has been extensive research in schema matching [13, 14, 21, 23]. However, works in schema matching tackle the problem of mapping individual columns from two database table schemas, whereas, in our case, the column alignments from two table schemas are only intermediary input into determining the actual fine-grained relation between two tables.

3 PRELIMINARIES AND OVERVIEW

3.1 Terminology

We consider Wikipedia articles $A = \{a_1, \ldots, a_n\}$; each article a is associated with a set of Wikipedia categories $\Psi_a = \{\psi_1, \ldots, \psi_n\}$. From all the categories we induce the category graph Ψ, which consists of *parent* and *child* relations between categories ψ_i `childOf` ψ^p. The parent/child relations allow us to establish a hierarchical graph in Ψ. The level of a category is denoted by λ_ψ.

Next, we define the tables from an article a as $T_a = \{t_1, \ldots, t_n\}$. A table t consists of a *table schema* (or *column header*) that we define as $C(t) = \{c_1, \ldots c_n\}$. Each column consists of a *textual description* and the set of all values $c_i = \langle desc, \{v_i^1, \ldots, v_i^n\}\rangle$ assigned to the corresponding column cells in the table rows $t_i(r) = \{r_i^1, \ldots, r_i^n\}$. More specifically, the cell value that is attributed to a specific *row* and *column* is indicated by v_i^k, where k is the row r^k and i is the column c_i. Cell values can point to existing articles in Wikipedia, that is $v_i^k = \langle a_k \rangle$, which we will refer to as *instance values*, or *primitive values* in cases of text, numbers, dates etc.

From the extracted tables $T = \{t_1, \ldots, t_n\}$ from A, we define two fine-grained types of relations between a table pair $\langle t_i, t_j \rangle$: (i) $t_i \vDash t_j$ where t_j is considered to be *semantically* a subPartOf of t_i, and (ii) $t_i \equiv t_j$ where t_i and t_j are *semantically* equivalent. We indicate the presence of relation with $r(t_i, t_j) \neq \emptyset$, and in the next section we precisely define the table relations.

3.2 Table Alignment Task Definition

In this section, we define the task of table alignment, and provide the definition for the fine-grained table relations.

Table Alignment. From the generated table pairs in the previous step, the task is to determine the *relation type* between any table pair $r(t_i, t_j)$ from the article pair $\langle a_i, a_j \rangle$. The relation types can be either subPartOf, equivalent or none (in case $r(t_i, t_j) = \emptyset$).

DEFINITION 1 (SUBPARTOF). *For a table pair* $r(t_i, t_j) = \{subPartOf\}$ *holds if the schema* $C(t_i)$ *can subsume either at the* **data value** *(i.e. cell-value) or* **semantically** *the columns from* $C(t_j)$ *(cf. Figure 1), that is,* $C(t_i) \supseteq C(t_j)$.

DEFINITION 2 (EQUIVALENT). *For a pair* $r(t_i, t_j) = \{equivalent\}$ *holds if both table schemas have* **semantically similar** *column representation (cf. Figure 1), that is,* $C(t_i) \approx C(t_j)$.

With the notion of *semantic similarity* we refer to cell *instance* values, whose similarity can be assessed at the Wikipedia category

level, e.g. the column $c = \langle$ *"Country"*, {*Germany*, *USA*, …}$\rangle$ contains values of type `Location`, thus c will be semantically similar to any column whose values are topically similar, despite the fact that at data level those values will not overlap. Similarly, is the case for the column descriptions, where *"Nation"* and *"Country"* usually refer to the same type of columns.

3.3 TableNet Overview

TableNet operates in a manner such that for any given *source* Wikipedia article $a_i \in A$, first, we generate article candidate pairs $\langle a_i, a_j \rangle$ ($a_j \in A$ and $a_i \neq a_j$) where the tables from the pair are likely to have an alignment relation (i.e. $r(t_i, t_j) \neq \emptyset$, where $t_i \in T(a_i)$ and $t_j \in T(a_j)$), which corresponds to the second step in TableNet. In the following, we describe the two main steps of TableNet:

(1) Article candidate pair generation
(2) Table alignment

4 ARTICLE CANDIDATE PAIR GENERATION

In the article candidate generation step, we address the problem of determining article pairs, whose tables will result in a table relation. In this case, we require our article candidate generation process to fulfill two main properties. First, we need to minimize the amount of *irrelevant article pairs*, whose tables do not result in a table relation. Second, the filtering out of article pairs (from the first property) should not affect the coverage in terms of retaining *relevant article pairs*, whose tables result in a table relation.

Thus, we define a function as shown in Equation 1 which provides optimal coverage of relevant article pairs, and at the same time minimizes the amount of irrelevant pairs.

$$A \times A \rightarrow r(t_i, t_j), \text{ where } t_i \in T(a_i) \wedge t_j \in T(a_j), \text{ and } a_i \neq a_j \quad (1)$$

This step is necessary as *naive* approaches which would enumerate over all article pairs to ensure maximal coverage result in a combinatorial explosion ($n!$ where n is the number of articles). Thus, in our candidate generation approach we circumvent this issue by defining features that fulfill the following desiderata:

- For an article pair whose tables will result in a table alignment, we expect the articles to be *semantically* or *topically* similar, which can be captured through the article abstracts or their category associations.
- For any two tables, whose parent articles fulfill the first criterion, we expect to find notions of similarity in terms of their schemas, such as column names

With these desiderata in mind, we define features that either operate at the article pair level or table level, and use them in two ways: (i) filter out irrelevant article pairs, and (ii) employ the features in a supervised manner to further filter out such pairs.

4.1 Features

Table 1 provides an overview of all the similarity features for the article candidate pair generation step.

Article Abstract. Wikipedia article abstracts contain a summary of the article, containing the most important information. For a *relevant article pair*, we expect that the abstracts will overlap in terms of abstract, specifically on *keywords*. For example, `Ariane Friedrich` and `Teddy Tamgho`, whose tables are in equivalent
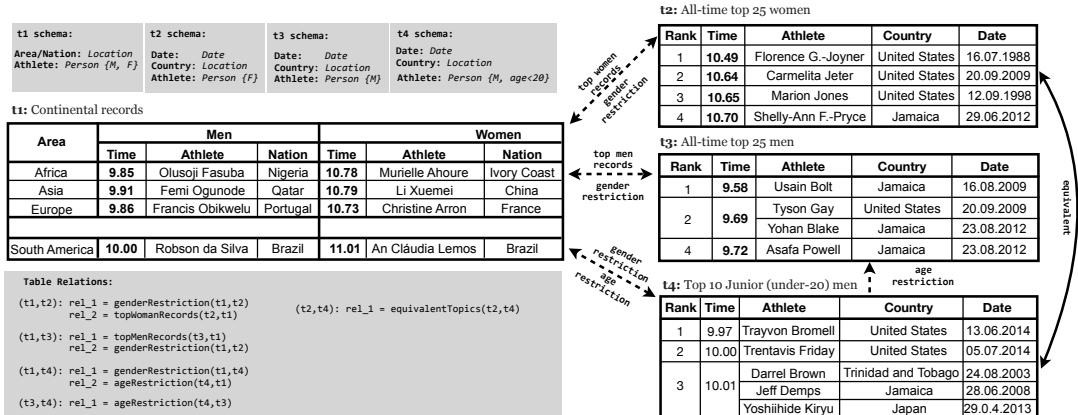
t1 schema:
Area/Nation: *Location*
Athlete: *Person {M, F}*

t2 schema:
Date: *Date*
Country: *Location*
Athlete: *Person {F}*

t3 schema:
Date: *Date*
Country: *Location*
Athlete: *Person {M}*

t4 schema:
Date: *Date*
Country: *Location*
Athlete: *Person {M, age<20}*

t1: Continental records

Area	Men			Women		
	Time	Athlete	Nation	Time	Athlete	Nation
Africa	9.85	Olusoji Fasuba	Nigeria	10.78	Murielle Ahoure	Ivory Coast
Asia	9.91	Femi Ogunode	Qatar	10.79	Li Xuemei	China
Europe	9.86	Francis Obikwelu	Portugal	10.73	Christine Arron	France
South America	10.00	Robson da Silva	Brazil	11.01	An Cláudia Lemos	Brazil

Table Relations:

(t1,t2): rel_1 = genderRestriction(t1,t2)
rel_2 = topWomanRecords(t2,t1)

(t1,t3): rel_1 = topMenRecords(t3,t1)
rel_2 = genderRestriction(t1,t2)

(t1,t4): rel_1 = genderRestriction(t1,t4)
rel_2 = ageRestriction(t4,t1)

(t3,t4): rel_1 = ageRestriction(t4,t3)

(t2,t4): rel_1 = equivalentTopics(t2,t4)

t2: All-time top 25 women

Rank	Time	Athlete	Country	Date
1	10.49	Florence G.-Joyner	United States	16.07.1988
2	10.64	Carmelita Jeter	United States	20.09.2009
3	10.65	Marion Jones	United States	12.09.1998
4	10.70	Shelly-Ann F.-Pryce	Jamaica	29.06.2012

t3: All-time top 25 men

Rank	Time	Athlete	Country	Date
1	9.58	Usain Bolt	Jamaica	16.08.2009
2	9.69	Tyson Gay	United States	20.09.2009
		Yohan Blake	Jamaica	23.08.2012
4	9.72	Asafa Powell	Jamaica	23.08.2012

t4: Top 10 Junior (under-20) men

Rank	Time	Athlete	Country	Date
1	9.97	Trayvon Bromell	United States	13.06.2014
2	10.00	Trentavis Friday	United States	05.07.2014
3	10.01	Darrel Brown	Trinidad and Tobago	24.08.2003
		Jeff Demps	Jamaica	28.06.2008
		Yoshiihide Kiryu	Japan	29.0.4.2013

top women records / gender restriction

top men records / gender restriction

gender restriction / age restriction

age restriction

equivalent

Figure 1: Table alignment example with `subPartOf` (dashed line), and `equivalent` relations (full line). `subPartOf` can be explained in terms of *age restriction* or *gender restriction*, whereas `equivalent` relation represents topically similar information.

	feature	description	group
f_1	*tfidf*	*tfidf* similarity between abstracts	
f_2	*d2v*	*doc2vec* similarity between abstracts	*abstract*
f_3	*w2v*	*avg.* word2vec abstract vectors similarity	
f_4	$sim(\Psi_{a_i}, \Psi_{a_j})$	similarity in embedding space between Ψ_a and Ψ_a^p categories for the article pair	
f_5	$\bigcap_{a \in \langle a_i, a_j \rangle} \Psi_a$	direct and parent categories overlap	Ψ & KB
f_6	$sim(a_i, a_j)$	embedding similarity of the article pair	
f_7	*type*	type overlap	
f_8	$sim(\psi_i, \psi_j)$	column title similarity (f_8^l) and column distance (f_8^d) between the schemas in a table pair	*tables*
f_9	$\lVert \gamma(\psi_i) - \gamma(\psi_j) \rVert$	category representation similarity γ	

Table 1: Article candidate pair features.

relation, both contain snippets indicating that the corresponding persons are ***athletes*** and ***jumpers***. This *topical similarity* is in line with the definitions of table relations in Section 3.

The features in Table 1 in the *abstract* group capture exactly such *topical* similarities. Feature f_2 computes a *doc2vec* [15] embedding for each article and measures the cosine similarity between those embeddings. Doc2Vec embeddings have the advantage that they take into account a broader context when compared to standard word2vec [17]. Additionally in f_3, we compute an *average embedding* from *word2vec* embeddings from the tokens in the abstract. We use GloVe pre-trained embeddings Wikipedia [22]. Finally, since none of the embedding can account for the *salience of tokens* in an abstract, we additionally compute the cosine similarity of the *tf-idf* (feature f_1) vectors from the article abstracts.

Categories & Knowledge Bases. Wikipedia categories are indicators that two articles are semantically similar in either *temporal*, *geographical*, or *topical* dimension (categories are created to indicate either *temporal* or *geographical* grouping, or *topical* similarity). Since articles are associated manually to categories, this association is prone to noise. As we show later in our evaluation, if we consider as pairs only articles assigned to the same categories, the resulting coverage will be low.

To circumvent this problem, we compute *graph embeddings* for Wikipedia categories based on the category graph Ψ. We use different embedding approaches like RDF2Vec [24] and Node2Vec [11]. This allows us to elevate the category comparisons from the link based structure in Ψ to the embedding space, and consider as candidates articles whose categories are close in the embedding space.

As features we consider the cosine similarity between the *directly associated categories* for an article pair, and additionally their *parent categories*. We also consider the similarity of articles in the same embedding space (Equation 2), and the jaccard similarity in terms of types in DBpedia.

$$sim(a_i, a_j) = \frac{emb(a_i) \cdot emb(a_j)}{\lVert emb(a_i) \rVert \lVert emb(a_j) \rVert} \quad (2)$$

Tables. The article pair features capture a coarse grained similarity for the tables in the articles themselves. We compute a set of light-weight features between the tables schemas for tables $t_i \in a_i$ and $t_j \in a_j$ from an article pair. The similarity corresponds to the *average word embedding* of the column description for two columns in the schemas $C(t_i)$ and $C(t_j)$. We consider only the highest matching column as measured in Equation 3. In addition to the similarity, we capture also the *positional index* difference between the highest matching columns, and the maximal matching in terms of a *column representation* which we explain below.

$$\max_{c_l \in C(t_j)} sim(c_k, c_l), \quad (3)$$

where $sim(c_k, c_l)$ is computed similarly as in Equation 2.

Column-Representation. In case a column c_i consists of instance-values, we compute a representation for c_i based on the attributes associated with the instances $\{v_i\}$ (where an instance points to a Wikipedia article), e.g. $v_i^1 =$ "*George Lucas' **bornIn** "Modesto, California, U.S.*". More specifically, since there may be multiple instances $|v_i| > 1$, we find the *lowest common ancestor* ψ_L category from $\{v_i\}$ by following the article-category associations in Ψ. This provides an abstraction over the values and is seen as a *type* of instances in $\{v_i\}$. By considering ψ_L instead of the individual $\{v_i\}$, we can summarize the column representation in terms of the most discriminative attributes in overall for ψ_L. In this way, for a

table pair, we compare the column representations, and in cases of a high match we assume the columns to be semantically similar.

The representation of c_i is computed as in Equation 4. We weigh the importance of *attributes* based on how discriminative they are for ψ_L, e.g. an attribute associated with articles directly belonging to category ψ_L are exclusive for ψ_L, and thus are weighted high. For an attribute p, the weight for ψ_L is computed as following:

$$\gamma(p, \psi_L) = \frac{\lambda_{\psi_L}}{\max \lambda_{\psi}} * \left(-\log \frac{|\bigcup o| : \forall \langle a, p, o \rangle \wedge a \in \psi_L}{|o| : \forall \langle a, p, o \rangle \wedge a \in \psi_L} \right) \quad (4)$$

where, the first part of the fraction weighs p by taking into account the level of λ_{ψ_L} and the deepest category where p is present in a target KB $\max \lambda_{\psi}$. $|\bigcup o|$ represents the number of distinct values assigned to attribute p from $a \in \psi_L$, whereas $|o|$ is the total number of assignments of p in ψ_L.

Through $\gamma(\psi_L)$ we capture the most *important* and *descriptive* attributes for a column c_i. For two columns in two table schemas, a high similarity $\|\gamma(\psi_i) - \gamma(\psi_j)\|$ is an indicator that the columns are semantically similar, which we use as a *table feature*.

4.2 Filtering & Classification

We use the features in Table 1 in two ways: (i) filter out article pairs that are *unlikely* to yield a table relation, and (ii) train a supervised model and classify article pairs as either *relevant* or *irrelevant*.

Filtering. We consider a *conjunction* of filtering criteria based on empirically evaluated thresholds for the individual features. Our main goal is to retain a high coverage of *relevant article pairs*, and at the same time filter out drastically *irrelevant pairs*. For thresholds we consider the *mean* value of a particular feature. This ensures that for a pair, if the score is below the mean value, that is an indicator that the pair is unlikely to yield any table relation. In Section 7 we show that we are able to drastically reduce the number of pairs by simply applying such thresholds.

Classification. From the remaining pairs we train a classification model and classify pairs as being either *relevant* or *irrelevant*. We consider as positive instances all table pairs from the article pair $\langle a_i, a_j \rangle$ which have *at least one* table relation, i.e, $r(t_i, t_j) \neq \mathsf{none}$, where $\exists (t_i \in a_i \wedge t_j \in a_j)$.
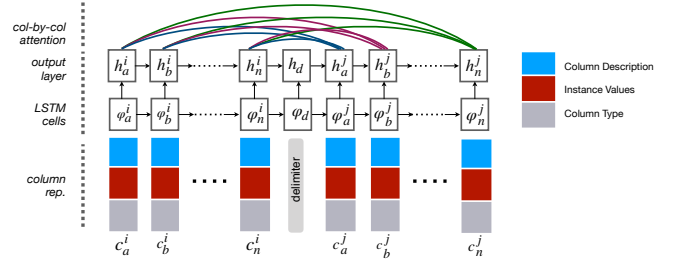
For classification we use Random Forests (RF) [6]. RFs allow to set *minimal amount of samples* that are allowed for a node in the tree to be split. This has direct implications in the accuracy of a classifier, however, this allows us to retain high coverage. Setting this number high makes the leafs of the different trees to be impure containing relevant and irrelevant article pairs. Our classifier will classify such impure leafs as relevant, at the cost of accuracy, however, in this way we retain a high recall. Section 7 shows that we can maintain high coverage of relevant pairs and at the same drastically reduce the amount of irrelevant pairs.

5 TABLE ALIGNMENT

TableNet is a bidirectional recurrent neural network (RNN), which for any table pair $r(t_i, t_j)$ learns to classify them into one of the relations $\mathsf{equivalent}$, $\mathsf{subPartOf}$, or none. For a model to accurately align table, the order of columns in their schemas needs to be taken into account. Additionally, the matching columns in the two schemas need to fulfill two main criteria: (i) *the context* in which the

columns occur needs to be semantically similar, and (ii) *the positions* in which the columns appear needs to be comparably similar [27].

Figure 2 shows an overview of the proposed alignment model. In the following we describe in details the means of representing tables, and the proposed architecture for the alignment task.

Figure 2: TableNet uses *BiLSTMs* to encode the tables as a sequence of columns. Each column can be represented in terms of its *description, instance values,* and its *column-type* (indicated by the different colors). The *column-by-column* captures soft-alignments between columns in the corresponding table schemas. (better viewed in color)

5.1 Table Representation

How we represent columns is key towards an accurate alignment model. A column in table schema consists of the following information $c_i = \langle desc, \{v_i^1, \ldots v_1^n\} \rangle$ (see Section 3).

Column Description. For a column c its description is a strong indicator of the cell-values v_i. We represent the column description tokens based on their word embeddings, specifically we use pre-trained Glove word embeddings [22]. In the case of multiple tokens, we average the respective word embeddings.

One disadvantage with this representation is that column descriptions can be ambiguous. For instance, *"Title"* can refer to various different values, e.g. $\mathtt{Movies, Books}$ etc. Thus, relying solely on column titles is not optimal for the table alignment process.

Instance–Values. In case c_i contains *instance values*, we represent c_i through the average embeddings of the individual cell-values v_i based on pre-computed graph embeddings. In our experimental evaluation, we used *node2vec* embeddings [11], which we trained on the Wikipedia *anchor graph*[1].

The combination of column *description* and *instance values* improves the representation of a column and reduces its ambiguity. This can be mostly attributed to the cases where c_i consists of more than one single instance, whereby by averaging the embeddings we establish the context in which such values can appear.

Column–Type. Representing the columns based solely on instance values poses a risk of biasing the column representation towards articles that are often linked together in the Wikipedia anchor graph, and thus it may ignore the topic information that is present in such articles based on their category associations.

Hence, for columns that contain instance values, we additionally represent it through its *type* or *category*. That is, for all the

---

[1]The anchor graph consists of nodes (Wikipedia articles and categories), while the edges correspond to the anchor text and the category-article associations.

instance values in v_i for c_i, we extract their lowest common ancestor category from Ψ. A similar idea was employed by Das Sarma et al. [27], where the columns are represented by the topics they are most often associated with, an information extracted from Google's query logs (e.g. *India* `Asian country`). Similar as for *instance-value* representation, here too, we represent the LCA category through graph embeddings. As we will see later in our experimental setup (see Section 6), since we we only ensure that Ψ is a consistent hierarchical graph, in cases where the LCA categories can be more than one, then we average their corresponding representations.

5.2 Table Alignment Architecture

In our model, we differentiate the input from the different tables through a *delimiter*. The model is an adoption of the one proposed by Rocktäschel [26] in the task of textual entailment. For a table pair $r(t_i, t_j)$ the model predicts the relation $r \in \{$ `equivalent`, `subPartOf`, `none` $\}$.

The alignment model corresponds to an RNN with LSTM cells[12], in that we read the sequence of columns for the table pair in both directions. Additionally, on top of the output layer from the RNN model, we compute a *column-by-column* attention, which helps us generate soft-alignments between columns in the table schemas, and thus further improve the alignment accuracy. In the following we describe the encoding of the column tables, and the intuition behind the attention mechanism.

Table Encoding. Since we have two separate tables, a precondition for accurate alignment is the encoding of the sequence of columns. Our model provides a *conditional encoding*, in that it first reads the columns from $C(t_i)$, then the cell state c_d, which is initialized with the last state of t_i (in this case c_n^i) is used to conditionally encode the sequence of columns in $C(t_j)$.

The advantage of the conditional encoding is that by encoding table t_j with initial cell state that corresponds to the last column cell state of t_i, we bias the model to learn encodings that are optimal for the task of table alignment. That is instead of trying to encode all columns, it will learn to encode the columns of t_j such that it can best predict the relation for the table pair. Since we have a bidirectional LSTM, we encode in a similar fashion the table t_i by conditioning it on the last state of t_j.

Attention Mechanism. In our case, for a table pair $r(t_i, t_j)$ to be aligned with either `equivalent` or `subPartOf` relation, we expect that the most important columns in each of the tables to have their corresponding matches in the respective schemas. This follows the intuition that not all columns in a table are equally important [27].

In this case, if we use the last cell state of the encoded table pair for classification, we enforce the model to weigh equally all the columns for determining the relation type for the table pair. Furthermore, for larger tables, the last cell state is expected to capture the information from all the previous states. A common workaround in such cases is to consider RNNs with attention mechanism [3]. In such cases the models are able to attend over all sequences with a specific *attention weight*. This has the advantage in that classification task is not carried solely based on the last state in a sequence, but instead the sequences are weighed based on their importance.

Column-by-Column Attention. In TableNet, we employ a more sophisticated attention mechanism, which addresses several disadvantages from the global attention mechanism [3]. The *column-by-column* attention mechanism works as following. After having encoded the last column from t_i, we process the columns in t_j individually and generate the attention weights w.r.t the columns in t_i. As a consequence, for each column in t_j we generate soft alignments to highest matching columns in t_j. After having processed all the columns in t_j and computing the corresponding attention weights (the upper part in Figure 2), for classification of the table pair $r(t_i, t_j)$ we will use a non-linear combination of the weighted representation of the last column c_n^j in t_j. We use *softmax* classification function for determining the label for $r(t_i, t_j)$.

The advantages of the *column-by-column* attention, is that it allows to encode all the desired table relation semantics, and additionally not enforce for two tables to have the same set of columns, given that not all columns are important for alignment. Thus, the alignment model in TableNet has the following properties:

- we can distinguish between columns from the different table schemas $C(t_i)$ and $C(t_j)$,
- for each column in $C(t_j)$ we can compute alignment weights to the column in $C(t_i)$ which function as *soft alignments* between columns in the respective schemas, and

6 EXPERIMENTAL SETUP

Here we describe the experimental setup for evaluating *TableNet*. First, we introduce the evaluation datasets, and then describe the setup for: (i) candidate generation, and (ii) table alignment.

The evaluation datasets and the code developed for all stages of TableNet are available for download[2].

6.1 Datasets

The main dataset in our experimental setup is Wikipedia. We use the entire set of Wikipedia articles from the snapshot of `20.10.2017`, with 5.5 million articles. Additionally, we use the Wikipedia categories, with nearly 1 million categories.

Wikipedia Tables. We extract tables from the HTML content of Wikipedia articles. From the entire snapshot of Wikipedia, only 529,170 Wikipedia articles contain tables. This resulted in a total of 3,238,201 Wikipedia tables. On average there are **6 tables per article** with an average of **6.6 columns**, and with an average of **10 rows** per table.

In more details, if we consider the composure of columns in the table schemas, more than 20% of columns in total consist of cell-values that are *instances* (see Section 3). Furthermore, if we consider the number of tables that contain columns with instance values, this number is significantly higher with 85%. This shows that in the vast majority of cases, we can represent tables, specifically the columns with highly rich semantic representations.

Wikipedia Categories. The category graph Ψ consists of nearly 1M distinct categories, organized in a *parent-child* graph. However, there are two main issues with using Ψ as is. First, it contains cycles, and second, categories are not depth-consistent, that is, the parents of a category do not belong to the same depth in Ψ.

---

[2]https://github.com/bfetahu/wiki_tables

We resolve these two issues, by first breaking any cycle in Ψ, and establish a depth-consistent graph s.t. for every Wikipedia category we remove any edge to its parents (ψ childOf ψ_p), where the level of the parent category $\lambda_{\psi_p} < \max_{\psi' \in \psi_p} \lambda_{\psi'}$, where with Ψ_p we denote all the parent categories of ψ. Removing such edges does not incur any loss in terms of *parent-child* relations between categories, as such categories can be reached through intermediate categories in Ψ. This process is performed iteratively from the root category until we have reached the leafs of Ψ.

6.2 Table Alignment Ground-Truth

We are the first to generate a large scale ground-truth for table alignment. Additionally, we are the first to distinguish between *fine-grained relations*, and additionally provide *coverage* guarantees for any given table in terms of its relations.

Our ground-truth consists of a sample of 50 source Wikipedia articles from which we construct article candidate pairs. Since the *naive* approach would generate 26.5M pairs, we apply a set of *filtering keywords* to filter out irrelevant article pairs. We filter articles by checking if a keyword appears *anywhere* in the article's content. The filtering keywords are chosen to fulfill two main criteria:

- *keywords are **generic**, in order not to filter out relevant pairs,*
- *topical keywords are **broad** s.t they can capture both coarse/fine grained topics (e.g. "athletics" vs. "jumper").*

We manually inspect a random sample of pairs that are filtered out, and assess if we remove pairs that should be considered relevant, and consequentially refine the keywords. For article pair that remain after filtering, we check if they can be seen as *false positives* and similarly refine our filtering keywords to remove such cases. We iteratively apply the refine and filtering steps, until we are left with an initial set of article pairs that we deploy for evaluation through crowdsourcing. For the remainder of article pairs, we construct all table pairs and rely on crowdsourcing to assess the table relations. Table 2 shows the stats for the three filtering iterations w.r.t the 50 source articles.

From the resulting 3.7k pairs, we have a set of 17k table pairs which we evaluate by means of crowdsourcing.

all pairs	iter-1	iter-2	iter-3
26.5M	416506 (▼63×)	10701 (▼38×)	3702 (▼2.9×)

Table 2: For 50 random source articles we applied three iterations of refine and filtering steps based on manual inspection. The reduction shows the factor with which the filtering reduces the pairs between each consecutive step is w.r.t the previous step. The final set of article pairs, whose table pairs we evaluate through crowdsourcing is 3.7k article pairs.

6.2.1 Evaluation Protocol. The table alignment can be of three categories: (i) equivalent, (ii) subPartOf, and (iii) none. To get reliable judgments, we provide detailed instructions and examples to the crowdworkers on how to determine the correct alignment relation. We guide the crowdworkers through the steps below:

(1) Find *important* columns that can be interpreted in isolation from the remaining columns in a table schema (such columns are known also as *subject columns*).

(2) Find matching columns in the two tables (that can be considered abstractly as *join keys*), where a match is considered if the columns contain *topically* similar information or the *same* information, and that their column descriptions are matching (e.g. *Nation* is equivalent to *Country*).

(3) If the previous two conditions are met, a table is considered as equivalent if the two tables contain similar or the same information, and where for the important columns in a table there are corresponding columns in the candidate table.

(4) The alignment subPartOf holds if one of the tables is a super set in the sense that it *contains* the information contained in the other table, or if *semantically* it is the superset of the other table (e.g. *"List of All Movies"* vs. *"List of Award Winning Movies"* for the same movie director). Alternatively, this can be seen as a table generated as a result of a selection over the superset table.

6.2.2 Evaluation Efforts. We ensure the quality of the labeling process by using only the highest level workforce in FigureEight. We follow guidelines on how to avoid unreliable workers by establishing a set of *test questions* that we generate manually [9]. Every crowdworker needs to pass them successfully with an accuracy of above 70%. Finally, if a crowdworker takes time less than an estimated *minimum amount of time* for completing the task, we discard their judgments.

On average, it took **2 mins** for the crowdworkers to judge 5 table pairs. This resulted in a total of **1,162 hours of work** for the entire ground-truth, for which we payed crowdworkers according to the minimum wage in Germany.

6.2.3 Ground-Truth Statistics. From **17,047 table pairs**, after labeling our ground-truth consists of 52% table pairs marked with noalignment relation, 24% marked with as having equivalent alignment, and the remaining 23% with subPartOf relation. The 47% portion of table pairs with a relation, result from **876 article pairs**, which presents a further ▼4.2× reduction of article pairs from our initial filtering step.

The *average agreement rate* amongst crowdworkers for table pairs is 0.91, which is measured as a combination of the worker's confidence score and the agreement rate.

6.3 Baselines and TableNet setup

We compare *TableNet* in two main aspects: (i) efficiency in candidate pair generation, and (ii) table alignment.

6.3.1 Candidate Generation Baselines. In the candidate generation phase we first look for article pairs whose tables are likely to yield an alignment $r(t_i, t_j) \neq \emptyset$.

Greedy – G. For each article we consider as pairs all other articles containing a table. It has maximal coverage, however the amount of irrelevant pairs is extremely high.

Direct Categories – C1. We consider as pairs articles that are associated with the same *directly connected* categories. Due to the noisy article-category associations, there is no guarantee that we will have maximal coverage of relevant pairs.

Deepest Category – C2. Wikipedia articles are associated with categories that belong to different levels in the category hierarchy.

As pairs we consider all articles that belong to the *deepest category* in the hierarchy in C.

Parent Categories – PC. To increase the coverage of relevant pairs, we consider as pairs, articles that have the same *parent categories* based on their directly associated categories.

Milne-Witten – MW. In MW we consider as pairs all articles that are related (for some threshold τ) based on the Milne and Witten relatedness score [18]. We compute the relatedness score on the Wikipedia anchor graph.

6.3.2 Table Alignment Baselines. We consider the following baselines for the table alignment step.

Google Fusion. The work in [27] finds related tables for a given table by computing a set relatedness scores against all possible table candidates. Two tables are related if their schemas are related based on max-weight bipartite graph matching score (see Section 2 for a detailed discussion). Google Fusion is unsupervised, thus, we use a threshold τ (we fine tune τ s.t we find the best F1 score) to classify table pairs as either having a relation or not.

TableNet$_{LR}$. Here, we consider as a competitor a standard supervised model based a logistic regression model, which we train using the features in Table 1. Here, our aim is to show the necessity of using more computationally heavy approaches like RNNs.

LSTM and BiLSTM. We use standard long-term short-memory networks to train a strong baseline for table alignment. Similarly, as in TableNet, here too, we will use the different column representations introduced in Section 5. Similarly, we use a bidirectional LSTM as a baseline.

Setup: LSTM, BiLSTM & TableNet. We set the number of dimensions for the *hidden layer* to be 100. We train the models for 50 epochs and use 60% of the data for training, 10% for validation, and the remaining 30% for testing.

We represent columns based on three main representations, which we explained in Section 5. The simplest representation is based on the column description which we mark with TableNet^{desc}, and then incrementally add the instance-value representation which we denote with TableNet^{+val}, and finally add the *type* representation denoted with TableNet^{+type}. In the cases where we represent a column through more than one representation, we simply add up the different representations. Similarly are represented the baselines LSTM and BiLSTM.

For classification we use the *softmax* function, and optimize the models to minimize the *categorical cross-entropy* loss.

6.4 Evaluation Metrics

We distinguish between two sets of evaluation metrics, aimed at measuring the performance of the candidate generation process, and the table alignment.

Candidate Generation. The main aim is to *minimize* the amount of *irrelevant article pairs* $\langle a_i, a_j \rangle = \emptyset$, and at the same time retain pairs whose tables have an alignment. We compute Δ as the metric measuring the amount of reduction we achieve w.r.t the *greedy* approach in generating article pairs.

$$\Delta = 1 - \frac{\langle a_i, a_j \rangle}{k * |A|} \text{ where } a_i \neq a_j \wedge a_i, a_j \in A \qquad (5)$$

where, k is the number of source articles.

Coverage we measure through *micro* and *macro* recall indicated with R_μ and R, respectively. R_μ represents the recall in terms of all table pairs from all the source articles, whereas macro recall measures the average recall scores from all source articles.

Table Alignment. We rely on standard evaluation metrics, such as *precision* (P), *recall* (R), and *F1 score* (F1).

7 EVALUATION RESULTS

In this section, we present in details the evaluation results for TableNet and our competitors in terms of candidate generation efficiency and coverage, and the performance in table alignment.

7.1 Candidate Generation

Here we discuss the evaluation results in terms of efficiency in generating relevant article candidate pairs, and compare w.r.t Δ against *greedy approach*. Additionally we show the recall scores in retaining relevant article pairs.

Baselines. Table 3 shows the efficiency and coverage results for the baseline strategies. From the baselines we notice that the use of the Wikipedia category graph Ψ reduces the amount of irrelevant pairs drastically. In terms of recall, baseline **PC** maintains high recall with $R = 0.83$, and at the same time reduces by $\Delta = 87\%$ the amount of irrelevant pairs when compared to *greedy* approach. **MW** and **C2** achieve the highest reduction rate Δ. However, for **MW** the coverage of relevant article pairs is very low with $R = 0.49$.

The results in Table 3 show that despite the high reduction rates for the different baselines, we still face the issue of either having a highly imbalanced ratio of relevant and irrelevant pairs, or in some cases like **C2** where the reduction rate is the highest, the recall is low $R = 0.49$. Thus, the balance between coverage and efficiency is not maintained. We show that we can improve the deficiencies of the baseline approaches through our feature set in Table 1.

	$|\langle a_i, a_j \rangle|$	Δ	*rel. pairs*	R
G	26,500,000	-	876	1.0
PC	3,486,031	0.87	724	0.83
C1	792,701	0.97	571	0.65
MW	33,890	0.99	429	0.49
C2	6,738	0.99	440	0.50

Table 3: Reduction rate for baselines. Higher Δ means that there are less irrelevant pairs for the table alignment step.

TableNet: Filtering & Classification. We first filter out article pairs whose tables are unlikely to yield a relation and then classify the remaining pairs to further filter out irrelevant pairs.

Filtering. The filtering step uses the features in Table 1 to remove irrelevant article pairs. Figure 3 shows the impact of the different features in reducing the amount of article pairs w.r.t *greedy* approach. For instance, the f_2 feature, which computes the similarity of article abstracts based on their *doc2vec* representation, provides a high reduction with $\Delta = 0.91$. This feature follows our intuition on generating the article pairs for the ground-truth (see
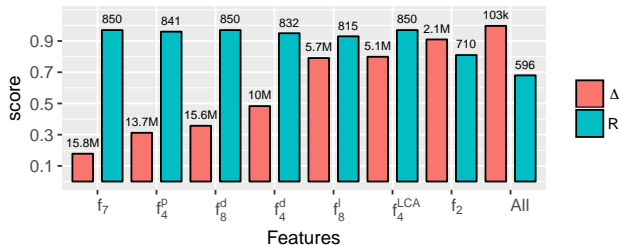
Section 6), where the *topic* and other semantic similarities for an article pair can be extracted from the article's content.

In terms of recall, we see that in majority of the cases the individual features have $R \geq 0.80$ coverage.

Since different features capture different notions of similarity, we apply them in *conjunction*, resulting in very high reduction rate of article pairs with $\Delta > 0.99$, and at the same time retaining a relatively high coverage with $R = 0.68$. The reduction compared to the greedy approach is more than ▼255 times less pairs.

We believe that this high reduction rate and at the same time the relatively high recall of relevant pairs, when compared to the baseline approaches can be attributed to the fact that we consider the similarities of articles, and their corresponding categories and articles' content in the embedding space. This allows us to capture implicit semantics that cannot be capture for instance through the simple link-based structure in the category graph Ψ.

Figure 3: Feature impact in terms of reducing the amount of irrelevant pairs and the coverage of $\langle a_i, a_j \rangle_r$ pairs. On top of each bar we show the corresponding number of article pairs (for Δ) and the number of retained relevant pairs (for R).

With the precomputed features we train a classifier and further filter out irrelevant pairs from the filtered articles in Figure 3.
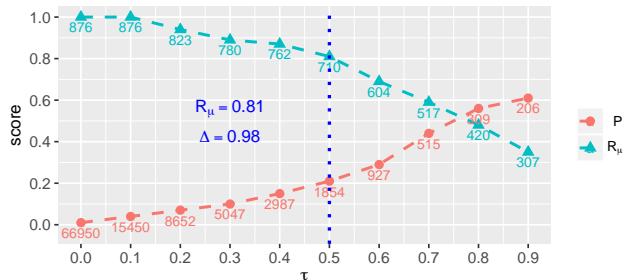
Classification. Determining whether a pair of articles, specifically, if their corresponding tables will result in a alignment relation is a difficult learning task.

From the previous filtering step, irrespective of the high reduction rate from 26M pairs to only 103k pairs, the amount of irrelevant pairs is still too high for any supervised approach to be able to learn models that predict with great accuracy the table relations. Thus, based on the configured RF model for high coverage (see Section 6), we train it on the feature set in Table 1 to further classify irrelevant pairs and filter them out.

Figure 4 shows the evaluation results for varying confidence thresholds of the RF model. With increasing threshold τ we can predict with higher accuracy pairs into their respective classes, whereas with lower thresholds we allow for more misclassifications. The increase of the confidence τ is directly proportional with the decrease in the amount of relevant pairs. This is intuitive as from the 103k pairs, only 876 pairs are actually relevant. However, based on the configuration of the RF (see Section 6), we are able to retrieve relevant pairs with high coverage, by slightly allowing irrelevant pairs to pass through.

We choose the confidence score to be $\tau = 0.5$, as it shows the best trade-off between the coverage of relevant pairs, and the amount of irrelevant pairs that are passed onto the table alignment step.

We achieve a high reduction rate of $\Delta = 0.982$ leaving us with only 1.8k pairs, and with a recall of $R_\mu = 0.81$.

Figure 4: For varying classification confidence τ in the x-axis, we show the scores for R_μ (turquoise line), and the precision P (red line). For each confidence score τ we show the corresponding amount of relevant pairs we retain in the case of recall, whereas for precision we show the amount of total pairs. We choose $\tau = 0.5$ which results in $R_\mu = 0.81$ and a $\Delta = 0.982$ w.r.t the 103k pairs from the filtering step.

Furthermore, if we compare this to the original amount of pairs from the greedy approach with 26M pairs, 103k from our pre-filtered candidates, and finally this drops to only 1.8k pairs in total after the classification step. Depending on the use case, one can use higher thresholds and thus have a higher ratio of relevant pairs, which makes the alignment task more efficient.

7.2 Table Alignment

In this section, we show the results for the table alignment step. From the article pairs marked as relevant in the previous step, we classify the corresponding tables into their corresponding alignment relation, $r(t_i, t_j) \rightarrow \{\text{equivalent}, \text{subPartOf}, \text{none}\}$.

Performance. Table 4 shows the alignment evaluation results for TableNet and all the competitors. Apart from the *Google Fusion* baseline, all the baselines are supervised models. In the case of *Google Fusion*, we consider a table pair to be related if their matching score is above some threshold that we determine empirically s.t we have the highest F1 score.

Google Fusion. This baseline has a reasonably high accuracy in determining whether a $r(t_i, t_j) \neq \emptyset$. Here we cannot distinguish between the different classes as the approach is unsupervised. In terms of recall it has the lowest score. This is due to the fact that the matching is performed by considering only the *column type* and the *column title* similarity. Additionally, the *bipartite* matching algorithm cannot retain the order of the columns, which is highly important for determining the alignment relation.

TableNet$_{LR}$. In this baseline we trained a *logistic regression* (LR) model with the feature set in Table 1, which classifies the equivalent and subPartOf relations with $F1 = 0.804$ and $F1 = 0.648$, respectively. When compared to Google Fusion, it achieves a 52% relative improvement in terms of F1 score for equivalent class, whereas if we take the average of both classes equivalent and subPartOf then the F1 improvement is 37%.

	equivalent			subPartOf			noalignment				
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Acc**	**R**	**F1**
Google Fusion	0.809	0.405	0.540	-	-	-	-	-	-			
TableNet$_{LR}$	0.824	0.790	0.804	0.612	0.688	0.648	0.754	0.730	0.742	0.730	0.723	0.731
LSTM^{desc}	0.851	**0.926**	0.887	**0.696**	0.816	0.751	0.870	**0.770**	0.817	0.806	0.837	0.818
LSTM^{+val}	0.865	0.913	0.888	0.668	0.977	0.794	0.936	0.722	0.815	0.823	0.871	0.832
LSTM^{+type}	0.839	0.935	0.884	0.547	0.976	0.701	0.933	0.564	0.703	0.773	0.825	0.763
BiLSTM^{desc}	0.883	0.891	0.887	0.684	0.960	0.799	0.918	0.752	0.827	0.828	0.868	0.838
BiLSTM^{+val}	0.877	0.871	0.874	0.684	0.975	0.804	0.915	0.747	0.823	0.826	0.864	0.834
BiLSTM^{+type}	0.854	0.908	0.880	0.690	0.957	0.802	0.925	0.741	0.823	0.823	0.869	0.835
TableNet^{desc}	**0.888**	0.884	0.886	0.686	0.947	0.796	0.909	0.759	0.827	0.828	0.863	0.836
TableNet^{+val}	0.856	**0.926**	**0.890**	0.675	**0.993**	0.804	**0.952**	0.719	0.819	0.828	**0.880**	0.838
TableNet^{+type}	0.872	0.903	0.887	0.692	0.961	**0.805**	0.925	0.752	**0.829**	**0.830**	0.872	**0.840**

Table 4: Evaluation results for the tasks of table alignment for the different competitors and TableNet. The evaluation results correspond to our manually constructed ground-truth dataset.

This shows that the proposed feature set is able to capture relations of type equivalent with high accuracy. However, it often misclassifies the subPartOf into the none and equivalent classes. One reason for this misclassification is since the features and the LR model cannot capture the sequence of columns, and it is not trivial to incorporate the information from the cell-values into the model for classifying the table relation.

LSTM and BiLSTM. One key motivation in this work is the hypothesis that through sequence based models, we can retain the order of columns in their respective schemas, an important aspect in determining the table alignment relation. The LSTM and BiLSTM approaches represent very competitive baselines. An additional advantage which addresses a deficiency in the standard supervised models, is that we jointly encode the different representations of a column for the classification task. Representing the columns as a combination of their description in the word embedding space, and the type and instance values through graph embeddings, we can capture complex relationship between the column description and the underlying cell-values.

For equivalent relations, LSTM^{+val} and BiLSTM^{+desc} achieve the highest F1 scores with $F1 = 0.886$ and $F1 = 0.887$, respectively. For subPartOf relations, the results look slightly different, with LSTM^{+val} still having the highest F1 score, whereas for BiLSTM, BiLSTM^{+val} the representation based on the column description and instance values achieves the highest F1 scores. The introduction of the column type in BiLSTM^{+type} provides a further boost in the accuracy of determining subPartOf relations. One conclusion we draw from the comparison between the two relations and two models, is that for subPartOf relations the *column type* provides additional power on determining the table alignment relation, whereas for equivalent it does not provide an additional advantage. These findings are inline with [27], where column type can provide important information in finding related tables. Comparing the LSTM and BiLSTM baselines against TableNet$_{LR}$, we gain 10% relative improvement in terms of F1 score for equivalent relation, and with 22% in the case of subPartOf relation. While for *Google Fusion* we have a 64% improvement for equivalent relation.

TableNet. In our approach, we addressed several deficiencies from the related work. Through our *column-by-column* attention mechanism, we can compute soft alignments between columns in the respective table schemas and thus take into account the position of the matching columns in the corresponding schemas. Additionally, the column representations allow us to capture the similarity between columns and the schema context in which they appear, and additionally the representation context based on their description, type and its instance values.

The evaluation results reflect this intuition. Comparing our best performing setup, TableNet^{+type} achieves an overall $F1 = 0.840$ across all three classes. We achieve a relative improvement of 64% when comparing F1 scores for the equivalent class against *Google Fusion*, or 56% if we compare the average F1 score for both alignment relations (equivalent and subPartOf). Against TableNet$_{LR}$ we observe high improvements for both alignment relation classes with a relative increase of 10% and 24% in terms of F1 score, for equivalent and subPartOf, respectively.

LSTM and BiLSTM are two strong competitors. They are able to capture the sequence information in the table schemas, and additionally provide the means to capture the contextual similarity between the column description, type and instance cell-values. TableNet^{+type} outperforms both approaches on average F1 score across all classes. For the individual classes, we note a variations amongst the different configurations of TableNet that perform best (marked in bold). The relative improvements are not as high as when compared against Google Fusion and TableNet$_{LR}$, however, they are consistent in nearly all cases. This confirms the usefulness of the attention mechanism for the alignment task, where we achieve an overall better performance in terms of F1 score.

8 CONCLUSIONS AND FUTURE WORK

In this work, we presented TableNet, an approach for table alignment by taking into account the coverage of table relations by providing an efficient approach for generating article pairs, whose tables we consider for alignment with a high accuracy. We provide

fine-grained relations `equivalent`, `subPartOf`, or `none`, a significant improvement over existing works.

We constructed an exhaustive ground truth for a random sample of 50 Wikipedia articles for which we labeled all possible table pairs, providing a dataset against which we can measure the coverage of table relations, and additionally provide high quality labels for more than 17k table pairs in our ground-truth.

In terms of *efficiency*, we show that from a naive approach which produces 26.5M pairs we can provide an efficient means that guarantees a high coverage of more than 68% and at the same time reducing the amount of pairs by ▼255 times. In terms of *table alignment*, we show that we can improve over strong baselines. We showed relative improvement of 56% when compared to *Google Fusion*, and with 17% when compared against TableNet$_{LR}$, a standard feature based model. If we compare against LSTM^{+type} and BiLSTM^{+type}, we again achieve improvements in terms of F1 score, thus, validating our hypothesis that a *column-by-column* attention mechanism provides soft alignments for columns across table schemas.

Future Work. As future work we foresee the task of *relation typing* s.t we can provide a attribute-based explanation of the relations in the case `equivalent` alignment, and for `subPartOf` provide attribute restrictions, i.e., in terms of the semantics of the respective tables, or in the form of *selection* criteria that may apply to generate a sub or superset from a table s.t. the `subPartOf` alignment holds.

REFERENCES

[1] A. Abujabal, R. S. Roy, M. Yahya, and G. Weikum. Never-ending learning for open-domain question answering over knowledge bases. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1053–1062, 2018.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735, 2007.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] C. S. Bhagavatula, T. Noraset, and D. Downey. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, IDEA@KDD 2013, Chicago, Illinois, USA, August 11, 2013*, pages 18–26, 2013.

[5] C. S. Bhagavatula, T. Noraset, and D. Downey. Tabel: Entity linking in web tables. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, pages 425–441, 2015.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] 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.

[8] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879, 2017.

[9] U. Gadiraju, R. Kawase, S. Dietze, and G. Demartini. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, pages 1631–1640, 2015.

[10] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon. Google fusion tables: web-centered data management and collaboration. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 1061–1066, 2010.

[11] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864, 2016.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] M. Koutraki, N. Preda, and D. Vodislav. SOFYA: semantic on-the-fly relation alignment. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.*, pages 690–691, 2016.

[14] M. Koutraki, N. Preda, and D. Vodislav. Online relation alignment for linked datasets. In *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, pages 152–168, 2017.

[15] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196, 2014.

[16] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347, 2010.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[18] D. N. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 509–518, 2008.

[19] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011, 2009.

[20] E. Muñoz, A. Hogan, and A. Mileo. Using linked data to mine RDF from wikipedia's tables. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pages 533–542, 2014.

[21] B. P. Nunes, A. A. M. Caraballo, M. A. Casanova, B. Fetahu, L. A. P. P. Leme, and S. Dietze. Complex matching of RDF datatype properties. In *Database and Expert Systems Applications - 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings, Part I*, pages 195–208, 2013.

[22] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014.

[23] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[24] P. Ristoski and H. Paulheim. Rdf2vec: RDF graph embeddings for data mining. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, pages 498–514, 2016.

[25] D. Ritze, O. Lehmberg, and C. Bizer. Matching HTML tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS 2015, Larnaca, Cyprus, July 13-15, 2015*, pages 10:1–10:6, 2015.

[26] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kocisky, and P. Blunsom. Reasoning about entailment with neural attention. In *International Conference on Learning Representations (ICLR)*, 2016.

[27] A. D. Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 817–828, 2012.

[28] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706, 2007.

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

[30] S. Zhang and K. Balog. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1553–1562, 2018.