Natural Key Discovery in Wikipedia Tables

Leon Bornemann
leon.bornemann@hpi.de
Hasso Plattner Institute, University of Potsdam
Berlin, Germany

Tobias Bleifuß
tobias.bleifuss@hpi.de
Hasso Plattner Institute, University of Potsdam
Berlin, Germany

Dmitri Kalashnikov
dvk@research.att.com
AT&T Labs – Research
Bedminster, New Jersey

Felix Naumann
felix.naumann@hpi.de
Hasso Plattner Institute, University of Potsdam
Berlin, Germany

Divesh Srivastava
divesh@research.att.com
AT&T Labs – Research
Bedminster, New Jersey

ABSTRACT

Wikipedia is the largest encyclopedia to date. Scattered among its articles, there is an enormous number of tables that contain structured, relational information. In contrast to database tables, these webtables lack metadata, making it difficult to automatically interpret the knowledge they harbor. The natural key is a particularly important piece of metadata, which acts as a primary key and consists of attributes inherent to an entity. Determining natural keys is crucial for many tasks, such as information integration, table augmentation, or tracking changes to entities over time.

To address this challenge, we formally define the notion of natural keys and propose a supervised learning approach to automatically detect natural keys in Wikipedia tables using carefully engineered features. Our solution includes novel features that extract information from time (a table's version history) and space (other similar tables). On a curated dataset of 1,000 Wikipedia table histories, our model achieves 80% F-measure, which is at least 20% more than all related approaches. We use our model to discover natural keys in the entire corpus of Wikipedia tables and provide the dataset to the community to facilitate future research.

CCS CONCEPTS

• **Information systems → Information integration**; **World Wide Web**; **Entity resolution**.

KEYWORDS

Webtables, key discovery, natural key, information integration

ACM Reference Format:
Leon Bornemann, Tobias Bleifuß, Dmitri Kalashnikov, Felix Naumann, and Divesh Srivastava. 2020. Natural Key Discovery in Wikipedia Tables. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3380039

1 MOTIVATING NATURAL KEYS

Tables on the web are a significant source of structured information. In particular, just the English version of Wikipedia contains more than 1 million tables as of 11/2017, many of which are relational. When interpreting such webtables, it is especially important to identify the contained entities, as these are needed for many important applications, such as information integration [10], extending knowledge bases [18] or change exploration [4].

Prior work has frequently used the notion of *subject column* to identify entities in relational webtables. The subject column is commonly understood to be "a column that lists the entities the table is about" [3]. There exist several approaches to discover subject columns in a relational webtable [3, 14, 25]. While these approaches achieve high accuracy, using only subject columns to identify entities imposes a limitation on the types of identifiers that can be discovered. For example, subject column discovery approaches usually discard numeric columns, which in practice can be meaningful identifiers, such as years or uniform numbers in sport events. Furthermore, entity identifiers may be split into multiple columns, which by definition cannot be detected by subject column discovery approaches. Examples of such multi-column identifiers are name and year for scientific conferences, or athlete and event for the results of a nation at the Olympic Games.

To solve these shortcomings, we propose to use the natural key as the main tool to identify entities in tables found in Wikipedia articles (in the following called Wikipedia tables). We formalize our understanding of a natural key below. Until then, the notion of a natural key can intuitively be understood as an extension of the subject column concept to allow for multi-column identifiers and context-aware identifiers. Besides identifying entities, the natural key is also an important metadata, that can facilitate many important tasks, such as information integration and table augmentation. For example, they serve as points of reference for creators of other tables, and enable reference discovery. It should be noted that the problem of natural key discovery in webtables is not solved by well-known knowledge bases, such as Wikidata or Yago, as these knowledge bases use surrogate keys to identify their contained entities and also do not use any automatic import from tabular data in Wikipedia. Thus, many facts from relational tables in Wikipedia are not contained in commonly used knowledge bases.

While we are not aware of any formal definition of natural keys in scientific literature, there are commonly accepted definitions for the natural key. For example, Wikipedia defines a natural key as a *"type of unique key in a database formed of attributes that exist and are used in the external world outside the database"*[1]. Even though this definition is good at providing the initial intuition, we, however, claim that it is not sufficient. For example, the exact box-office gross of a movie is a real-world attribute and can be expected to be unique for all notable movies. Such an attribute is not a good choice for a natural key, because its values are constantly changing while the movie is in theaters. Even for movies that are no longer in theaters, it would be difficult to look up more information about a movie given only its box-office gross (instead of its title, for example). We thus propose the following, more useful formal definition:

Definition 1.1. A *global natural key* of a table T is a set of columns C that has the following properties

(1) **Keyness**: C has all syntactical properties of a primary key in T and each value of C identifies an entity globally.
(2) **Temporal Stability**: Values of C are stable across time.
(3) **Spatial Identifiability**: Independent sources identify the values of C as identifiers.

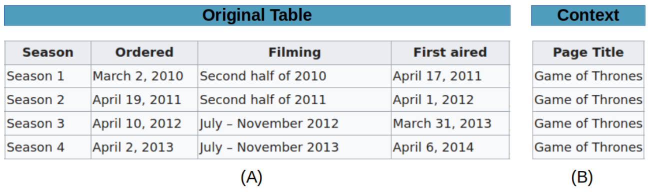
The two additional aspects of temporal stability and spatial identifiability distinguish our definition from existing definitions. Temporal stability demands that a global natural key must also be a key in all past or future versions of T with the same schema, essentially ruling out candidates that are unique by chance. Spatial identifiability in a corpus of tables means that the values of a natural key actually need to be used by different sources, for example as keys or foreign keys of other, independently created tables.

Definition 1.1 specifies the concept of a natural key abstractly, but limits natural keys to unambiguous, global identifiers. This is problematic, because many tables in Wikipedia (and webtables in general) contain only local identifiers. An example of such a table is shown in Figure 1(A), which contains information about the seasons of the TV show "Game of Thrones". Within the context of the page, Season is a good identifier, but by itself cannot serve as a global natural key: its values could identify the seasons of any TV show or even the seasons of the year. However, by adding contextual information as constant columns to the table (Figure 1(B)), a global natural key can be found: the combination of Season and Page Title. This leads us to the notion of a *local natural key*:

Definition 1.2. Given a table T and a set of constant columns C', taken from T's context, C is a *local natural key* of T, if $C \subseteq T$ and there exists a subset of all context columns $\tilde{C} \subseteq C'$, such that $C \cup \tilde{C}$ satisfies all conditions of a global natural key.

Intuitively, this means that local natural keys can become global natural keys by adding the right context. The context of a table can typically include its caption, section header, and page title or URL. It is clear that every global natural key is also a local natural key. Hence, whenever we mention local natural keys in the rest of this paper, we implicitly mean local natural keys that are not also global natural keys. If we simply mention natural keys we implicitly mean both local and global natural keys.

[1]https://en.wikipedia.org/wiki/Natural_key (as of October 7, 2019)



Original Table				Context
Season	Ordered	Filming	First aired	Page Title
Season 1	March 2, 2010	Second half of 2010	April 17, 2011	Game of Thrones
Season 2	April 19, 2011	Second half of 2011	April 1, 2012	Game of Thrones
Season 3	April 10, 2012	July – November 2012	March 31, 2013	Game of Thrones
Season 4	April 2, 2013	July – November 2013	April 6, 2014	Game of Thrones

(A) (B)

Figure 1: A table on the Wikipedia page about Game of Thrones (the TV show).

It is easy to see that natural keys can have very different appearances depending on the domain: Movie titles look very different from year numbers, which again look very different from ISBNs or the combination of title, author, and venue. Consecutively increasing values can be valid natural keys (season numbers of a TV show, as in Figure 1) or generic row numbers that have no semantic meaning. Thus, it is difficult to solve the problem of natural key discovery with a manually established ruleset, which is why we propose a supervised machine learning approach. We model the problem of natural key discovery as a binary classification problem for sets of columns in a table. We engineer features from the data and layout of the table, the stability of columns over time (temporal dimension) and the appearance of values in other tables (spatial dimension). This allows us to outperform related approaches for the discovery of subject columns, which fail to recognize many natural keys as identifiers, due to the limitations previously discussed. In Section 4 we empirically demonstrate that our approach recognizes significantly more entity identifiers than state of the art techniques, outperforming them by over 20% of F-measure.

Overall, the main contributions of this paper are:

- The first formal definition of the natural key using the novel notions of temporal stability and spatial identifiability.
- A supervised learning approach for the discovery of natural keys using a variety of specially designed features.
- An experimental evaluation of our model.
- A publicly available corpus of Wikipedia table versions with manually annotated natural keys.
- The dataset of all tables in Wikipedia (including their past versions) with programmatically annotated natural keys.[2]

The remainder of the paper is organized as follows: Section 2 reviews related work. Subsequently, Section 3 describes how we model the discovery of the natural key as a classification problem and presents our set of features. Section 4 presents the experimental evaluation of our model. Finally, Section 5 summarizes our insights and concludes the paper.

2 RELATED WORK

Since the initial WebTables project [6], webtables have received considerable attention for different purposes, such as the detection of relational webtables [3, 22], schema extraction [1], the augmentation of tables for information retrieval [8, 23], answering queries with webtables [17], or information integration with knowledge bases [11]. A concise overview of many more applications and problems is provided in the retrospective by Cafarella et al. [5].

[2]All data and annotations are available at www.IANVS.org .

While the problem of finding join candidates for a webtable has been addressed [21, 26], the discovery of keys in raw webtables has not been actively studied yet. Recently, Lehmberg et al. introduced an approach that synthesizes large tables from individual webtables [13]. For these combined tables, it is able to determine the key by discovering approximate functional dependencies. While this approach works well for the synthesized tables, the authors argue that it is not well suited to individual webtables with few rows, because too many coincidental functional dependencies would be found. In raw webtables, prior work has frequently used the subject column to identify entities [3, 14, 25]. However, current approaches for the detection of subject columns are inadequate to discover many natural keys, and thus miss many valid entity identifiers. We show this by empirically comparing against the approaches by Lehmberg et al. [14] and Zhang [25] in Section 4.

The discovery of keys in traditional database systems has received much attention, including approaches to discover composite keys [19] and foreign key relationships [7, 9, 15, 24]. In a paper about schema normalization, Papenbrock et al. introduce a scoring system to determine the best candidate for the primary key [16], while Jiang et al. have published a method to jointly detect primary and foreign keys [12]. However, both of these methods were designed for the discovery of surrogate keys, which are system-generated and lack meaning. In contrast, natural keys consist of meaningful entity attributes, especially in webtables, making approaches for traditional database systems ill-suited to discover them.

Various researchers have also studied the problem of discovering keys or key-like attribute sets in RDF data [2, 20]. However, the problem of key discovery in knowledge bases is substantially different from the discovery of keys in actual tables: the data entries in knowledge bases are frequently created in a decentralized way and thus do not necessarily follow a strict schema.

3 A SUPERVISED LEARNING APPROACH FOR NATURAL KEY DISCOVERY

We model the problem of natural key discovery as a classification task, which we solve by extracting hand-crafted features from the tables. As discussed in Section 1, global and local natural keys must both meet three criteria: keyness, temporal stability, and spatial identifiability. To ensure this, we consider data directly from the webtable, and make use of two additional inputs that correspond to the temporal and spatial dimensions: the webtable's version history and a large corpus of other, independently created webtables.

3.1 Natural key discovery: A classification task

Given training data with known natural keys, we map the task of natural key discovery in webtables to a binary classification problem: For a given set of columns C in a table T, a set of past or future versions of this table H, and a corpus of independently created tables S, return *true* if C is the natural key of T, *false* otherwise. We call (C, T, H, S) an *instance* and refer to C as the *key candidate*. Wikipedia provides its version history, which means that H can be constructed for each table by matching tables across the different revisions of its pages. To construct this matching we use a bag of words model. In our experiments, we use the set of all tables in English Wikipedia pages as of 11/2017 as our table corpus S.

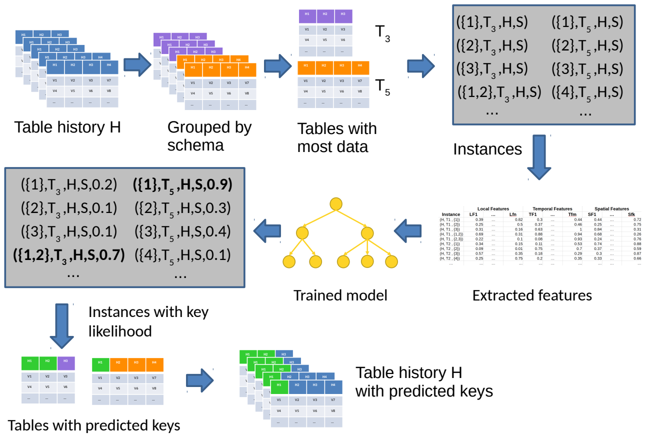


Table history H
Grouped by schema
Tables with most data
T3
T5
({1},T3,H,S) ({1},T5,H,S)
({2},T3,H,S) ({2},T5,H,S)
({3},T3,H,S) ({3},T5,H,S)
({1,2},T3,H,S) ({4},T5,H,S)
Instances
Extracted features
Trained model
({1},T3,H,S,0.2) ({1},T5,H,S,0.9)
({2},T3,H,S,0.1) ({2},T5,H,S,0.3)
({3},T3,H,S,0.1) ({3},T5,H,S,0.4)
({1,2},T3,H,S,0.7) ({4},T5,H,S,0.1)
Instances with key likelihood
Tables with predicted keys
Table history H with predicted keys

Figure 2: Predicting keys of unseen webtable histories.

Having established the problem instance, we face the challenge that the number of possible key candidates grows exponentially with the number of columns in a table. However, in the dataset that we manually annotated (see Section 4), we observed that 96.9% of all multi-column keys are either tightly clustered (all adjacent to each other) or consist of exactly two columns with exactly one non-key column between them. Ignoring all other multi-column key candidates reduces the number that needs to be considered to $O(m)$ per arity, where m is the number of columns in the table. Thus, given a set of annotated tables, we generate a reasonably sized set of training instances for each table, for which we then compute different features (see Section 3.2) that are used to train the model.

After the model is trained, the natural key of an unseen webtable is found by applying every plausible key candidate to the model and picking the candidate for which the model's posterior probability is highest. In practice, given a table history H, we do not need to predict the natural key for all $T \in H$, because many tables in H share the same schema and thus have the same natural key. Thus, we partition H into sets of tables $\{H_1, ..., H_k\}$ where each H_i consists of tables with the same schema. We then predict the natural key only for the table in each H_i that has the most data rows, and output it as the key for all tables in H_i.

We employ the same strategy when training the model on annotated table histories, so that the model does not overfit on features of large table histories with no schema changes. Figure 2 summarizes the process of predicting keys for an unseen table history.

3.2 Features for natural key discovery

Features must work for both single columns and sets of columns. Our solution is as follows: Given an instance (C, T, H, S) and a function f that calculates a feature for a single column, we sort all $c \in C$ by their position in the schema and subsequently calculate $f(c)$ for all $c \in C$. Such a function f results in $a_{\max}$ actual features in the feature space, where $a_{\max}$ is the highest key arity in the training set. If the arity of a key candidate is smaller than $a_{\max}$ we fill up the remaining feature instances of f with a reserved value – we use -1 for numeric features and "NA" for categorical features.

An overview over the entire feature set is given in Table 1. An implementation of our features and the model are available at www.IANVS.org. Corresponding to the definition, our features are divided into local, temporal, and spatial features. Some of our features are in the spirit of [16] while temporal and spatial features are new and were specifically designed for our approach.

Local features. The local features give the classifier information about the layout and content of the table or the current key candidate. These features include the number of columns, and rows; uniqueness of C; the position of $c \in C$ in the schema; the average content length; occurrences of style markups; or the presence of specific characters. Furthermore, we determine commonly used datatypes: *Integer, Decimal, Date, Year, Season (e.g., '2017/18'), Link, Ranking (consecutively increasing integers), Image Link, Wikipedia Template*. We also check whether values match a *Wikipedia Page Name*. Finally, we hash the string value of the column headers to obtain a basic numerical representation of their values.

Temporal features. The temporal features use historical information to assess the temporal stability of C, based on the other versions of T in H. To find versions of an individual column c in other tables in H, we execute a matching, once again based on a bag of words similarity measure. This matching of columns across the different table versions is used to determine the (relative) number of other occurrences of the key candidate C, as well as the number of occurrences in which C is unique. Furthermore, the number of deleted and inserted values in C over time gives an assessment of the overall volatility of $T[C]$. Finally, we can count *isolated updates* to values in C. We define an *isolated update* to a cell in a column as a change in the cell while all other cells in the associated row do not change (in that version). As keys are stable identifiers, values in the natural key should rarely receive isolated updates.

Spatial features. Given other, independently created tables (S), there are two aspects of the spatial dimension to consider: The usage of columns similar to C as a natural key; and the existence of references to the values in C. Because the former aspect requires ground truth about the keyness, it can be checked only against tables from the training set. The latter aspect can be checked against an entire corpus of webtables and thus must also scale accordingly.

To check whether a set of columns C is used as a natural key in other tables, one has to first find C in other tables. This is not a question of finding the same value set, as for example, two tables with the identical schema (ISBN,Author,Published) may contain different books, but the ISBN columns model the same concept and are clearly both natural keys. Thus, we search for similar schemata by using a distance measure between table headers. Given two tables T_1 and T_2 from different table histories with their respective headers TH_1 and TH_2 (represented as matrices of html elements), the distance between T_1 and T_2 is defined in the following way:

$$d_{TH}(T_1, T_2) = \sum_{i=1}^{n} \sum_{j=1}^{m} \delta(TH_1(i, j), TH_2(i, j)) \tag{1}$$

$$\text{where } \delta(v_1, v_2) = \begin{cases} \delta_s(v_1.text, v_2.text), & \text{if } v_1 \text{ and } v_2 \text{ exist} \\ \delta_e(v_1.text), & \text{if } v_1 \text{ exists} \\ \delta_e(v_2.text), & \text{otherwise} \end{cases}$$

and where n and m are the maximum number of rows and columns of both table headers, respectively, δ_s is a string-distance function and δ_e is the distance between a cell and a non-existent cell.

Given the string-distance functions of equality (eq), containment (ctmnt), and Levenshtein-distance (lvst), we construct four different table header distances, which are summarized in Table 2. Given such a table header distance measure (denoted as d), an instance (C, T, H, S), a threshold θ, and the training set of tables $S_T \subseteq S$, we define the following features:

- **#matches**: (relative) number of matches $|m(T, S_T)|$, where $m(T, S_T) = \{T' \in S_T \mid d(T, T') < \theta\}$
- **spatialKeyness**: (relative) number of matches, that have C as the key: $sk(T, S_T) = |\{T' \in m(T, S_T) \mid C \text{ is key in } T'\}|$
- **nnKey**: Boolean value that reflects if the nearest neighbor of T (with regards to d), also has C as the natural key
- **nnDist**: distance between T and its nearest neighbor according to d

The threshold θ needs to be set for each individual distance measure. However, given a distance measure d_{TH} and S_T, reasonable thresholds can be found automatically. The intuition is that we want to set θ so that spatial keyness is high for instances where C is in fact the key. To do so, we calculate d for each pair of table headers in S_T that originate from different table histories, and mark the result if the keys match. We then successively test different thresholds θ (with appropriate increments for each d_{TH}) and look at the set of all labeled distances smaller than θ. For these, we calculate the precision (distances marked as key matches are true positives). We then choose the highest θ that still achieves a desired precision (for example 90%) as the final threshold.

To check for references to values in C, we look for potential join-partners for the key candidate C in S. To do so, we use LSH ensemble [26]. This index can be queried to find the set of joinable tables by searching for columns that contain the query column to a specified degree (the containment threshold). For each $c \in C$ and each containment threshold $t \in \{50, 60, 70, 80, 90, 100\}$ we obtain exactly one feature: The number of columns in the corpus that contain at least $t\%$ of the values of c as returned by the index.

4 EMPIRICAL EVALUATION

Given the set of all Wikipedia table histories, we randomly sampled 1,000 table histories and annotated all versions of each contained table. This results in a dataset containing 7,295 tables, where each table contains an average of 5.5 columns of which 2.1 are unique. Table 3 shows the basic distributions of local and global natural keys in the hand-labelled dataset, and in all Wikipedia table versions (assigned by our model as described later in this section).

We evaluate our approach against the following approaches:

- **U:** Baseline returning the first unique column as natural key.
- **UNM:** Baseline returning the first non-numeric unique column as natural key.
- **LE:** A heuristic for discovering subject columns by Lehmberg et al., which is based on finding *rdfs:label* attributes [14].
- **ZH:** A score-based approach by Zhang et al. to discover subject columns [25]. The approach considers named entity columns and scores them based on uniqueness, completeness, position in the schema, similarity to the table's context and

Table 1: Complete feature set for an instance $I = (C, T, H, S)$. Features marked with # are always calculated as both absolute and relative measures. Features marked with a * are families of features, each calculated in the same way. Features marked with a † are calculated on each $c \in C$ individually (as described in Section 3.2) and we denote c to refer to the input column. IGR_G and IGR_L denote the information gain ratio on our manually labeled dataset (see Section 4) for the tasks of global and local key discovery. The scores for feature families are averaged. Information gain ratios greater than 0.025 are in bold font.

Feature name	Description	IGR_G	IGR_L
	Local Features (see Section 3.2 for details)		
#capitalLetters†	The number of capitalized letters in all $v \in T[c]$	0.014	0.010
#chars†	Number of characters in all $v \in T[c]$	0.009	0.011
#words†	Number of words in all $v \in T[c]$	0.019	0.019
colPosition†	Number of columns to the left of of c	0.015	0.017
dataType*†	The datatype of $T[c]$	0.008	0.009
dimensions*	Number of rows and columns of the table	**0.043**	**0.043**
distanceBetweenColumns	Summed up distance between the positions of all $c \in C$	0.003	0.005
hasHeader†	True if c has a header, false otherwise	0.005	0.006
header†	Hash representation of the header of c as a numerical value	0.023	**0.026**
keyArity	Number of columns of the key candidate ($\|C\|$)	0.002	0.004
#links†	Number of links in $T[c]$	0.022	0.007
#specialChars*†	Number of occurrences of special characters (e.g., , . ; ") in $T[c]$	**0.025**	**0.029**
#styleMarkups*†	Number of characters with a specific style markup in $T[c]$	**0.030**	**0.030**
#uniqueCols	Number of unique columns in T	**0.035**	**0.037**
uniqueness	$\|\{v \in T[C]\}\|/\#rows(T)$	**0.025**	**0.033**
	Temporal Features (see Section 3.2 for details)		
candidateUniquenessOverTime	$\|H_U\|/\|H\|$, where $H_U = \{T' \in H \mid C \subseteq T' \wedge T'[C]$ is unique$\}$	**0.063**	**0.079**
individualUniquenessOverTime†	$\|H_{IU}\|/\|H\|$, where $H_{IU} = \{T' \in H \mid c \in T' \wedge T'[c]$ is unique$\}$	0.011	0.014
#colPositionChanges†	Number of versions in which c changed its position in the schema.	0.014	0.017
#deletes	Number of deletes of old values in all $\{T'[C] \mid T' \in H \wedge C \subseteq T'\}$	**0.026**	**0.029**
#inserts	Number of inserts of new values in all $\{T'[C] \mid T' \in H \wedge C \subseteq T'\}$	0.016	0.017
#otherOccurrences†	$\|\{T' \mid T' \in H \wedge T' \neq T \wedge c \in T'\}\|$	0.022	0.023
#isolatedUpdates	The number of value changes in C.	0.004	0.004
	Spatial Features (see Section 3.2 for details)		
nnDist*	Table distance to the nearest neighbor	**0.103**	**0.105**
nnKey*	Keyness of equivalent key candidate in nearest neighbor	0.006	0.002
#matches*	Number of other tables whose headers are similar	**0.028**	**0.029**
referenceCount*†	Number of references to c in other tables	0.017	0.019
spatialKeyness*	Number of other tables whose headers are similar in which C is key	**0.040**	**0.038**

Table 2: Table-header distance measures

$\delta_s(s_1, s_2)$	$\delta_e(s)$	Description
$eq(s1, s2)$	2	Exact header matches
$ctmnt(s1, s2)$	2	Header containment
$lvst(s_1, s_2)$	$lvst(s_1, \text{“”}) * 8$	Similar headers, high missing cell penalty
$lvst(s_1, s_2)$	$lvst(s_1, \text{“”}) * 0.25$	Similar headers, low missing cell penalty

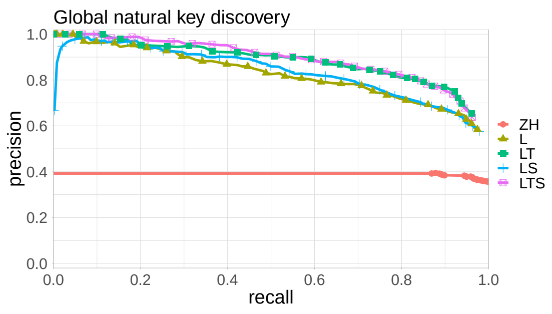
Table 3: Distribution of natural keys in the ground truth (hand-labelled and predicted) and the entire Wikipedia.

Arity	Ground Truth		Predicted		Entire Wikipedia	
	Global	Local	Global	Local	Global	Local
1	42.66%	48.89%	46.23%	51.11%	49.23%	48.05%
2	6.02%	2.00%	2.41%	0.25%	2.67%	0.05%
>2	0.10%	0.33%	–	–	–	–

a web-search score that counts occurrences of the named entities in pages returned by a search engine.

We compare these approaches against a random forest, trained using the following four subsets of our features: local features (**L**), local and temporal features (**LT**), local and spatial features (**LS**), and full feature set (**LTS**). We use 10-fold cross validation to evaluate the models. Given a table history H, we count each distinct schema in H as one example (as described in Section 3.1) which results in 1,618 tables on which we evaluate.

The F-measures of all approaches are reported in Table 4. The results show that related works *ZH* and *LE* do not perform significantly better and sometimes even worse than the baselines for either type of natural key. This confirms our intuition that existing



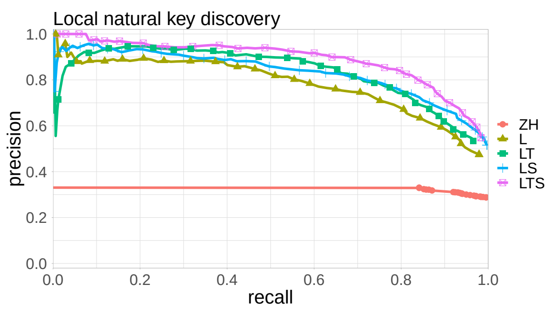


Figure 3: Precision-recall curves for global and local natural key discovery

approaches for subject column detection are too narrowly focused on finding specific types of identifiers and thus miss many natural keys that our approach is able to find.

Table 4 shows that our solution performs much better than all baselines and related work approaches. It is apparent that just the local features already help the classifier to perform quite well (>70% F-measure). The additional temporal and spatial features significantly improve this score. Using the full feature set as opposed to using just local features increases the F-measure by 7.4% for global natural keys and by 8.8% for local natural keys. When equipped with local, temporal and spatial features, our approach beats all the existing work by at least 20% F-measure for both natural key types.

Table 4: F-measures for all compared methods.

	Baselines		Related Work		Our Approach			
	U	**UNM**	**LE**	**ZH**	**L**	**LT**	**LS**	**LTS**
Global	0.480	0.560	0.470	0.550	0.735	**0.811**	0.752	**0.810**
Local	0.590	0.520	0.580	0.470	0.706	0.748	0.761	**0.793**

Figure 3 shows precision-recall curves for all approaches that have a configurable threshold. Our approach is able to balance precision and recall well, because even for relatively high recall values, the precision remains acceptable. The reported F-measures for all our methods were achieved by using a tuned classification threshold, which optimized F-measure. This resulted in balanced precision and recall for all models. For example *LTS* achieved 80.2% precision and 82.7% recall for global natural keys as well as 78.5% precision and 80.8% recall for local natural keys. That being said, it is also possible to tune the models for a higher precision at the expense of recall, should that be desired. The curves show that *LTS* can achieve 90.2% precision at 54.2% recall for global natural keys and 90.7% precision at 62.8% recall for local natural keys. Our investigation of classification errors revealed two common error types: The models struggle in cases where there is little useful temporal or spatial data available or natural keys use specific Wikipedia syntax like *{{CHN}}* or *{{flagathlete|[[Ellen van Dijk]]|NED}}* .

We have used our model to discover natural keys in the entire corpus of Wikipedia table histories. For efficiency reasons, we did not employ spatial features, thus using *LT* as our model. The corpus contains 40 million tables in total. On a server with two Intel Xeon E5-2650 2.00 GHz CPUs and 128 GB RAM, we were able to discover natural keys for the entire set of table histories in Wikipedia in less than five days, using five threads in parallel. When both models are already in memory, we are able to process a single table in less than 50 milliseconds. As there are on average fewer than 12,000 updates to tables in Wikipedia every day, our method would be able to process all daily updates in less than ten minutes.

5 CONCLUSION AND FUTURE WORK

We studied the problem of natural key discovery in Wikipedia tables. Natural keys are a particularly important piece of metadata that extends the traditional notion of the subject column, allowing additional types of identifiers to be recognized. We established the first formal definition of the natural key, incorporating context and distinguishing local and global natural keys. In our formulation, natural keys need to satisfy all requirements of a surrogate key, remain stable over time, and be identifiable across space.

Next, we presented a novel approach to discover such natural keys. We model the problem as a binary classification task and are able to predict single- and multi-column natural keys with a single model. In accordance with our definition, we extract features from the table itself, from its version history (time) and from independently created tables (space). Our experiments show that our approach finds significantly more entity identifiers than related approaches, clearly outperforming them by at least 20% F-measure for both local and global natural keys. We demonstrate that the usage of temporal and spatial features raises F-measure by at least 7% for both types of natural keys. When executing our discovery approach on the entire set of Wikipedia table versions (40 million tables) we are able to process a single table in 50 milliseconds, allowing us to process 15 years of Wikipedia tables in less than 5 days.

While our approach has currently been applied only to English Wikipedia tables, we are confident that it can also be applied to other language-versions and furthermore arbitrary webtables as well with a reasonable amount of additional effort: only two of our features are tailored to Wikipedia tables and none are language-specific. Furthermore, as the proposed method allows us to maintain the current state of Wikipedia tables and process incoming updates in an online and parallelized way, future work could design a system that maintains the state of all Wikipedia tables, updates them on a daily basis and discovers new natural keys in an online fashion.

REFERENCES

[1] Marco D Adelfio and Hanan Samet. 2013. Schema extraction for tabular data on the web. *PVLDB* 6, 6 (2013), 421–432.

[2] Manuel Atencia, Jérôme David, and François Scharffe. 2012. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 144–153.

[3] Sreeram Balakrishnan, Alon Y Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying WebTables in Practice. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*.

[4] Tobias Bleifuß, Leon Bornemann, Theodore Johnson, Dmitri V. Kalashnikov, Felix Naumann, and Divesh Srivastava. 2018. Exploring Change – A New Dimension of Data Analytics. *PVLDB* 12, 2 (2018), 85–98.

[5] Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten years of webtables. *PVLDB* 11, 12 (2018), 2140–2149.

[6] Michael J Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. 2008. Uncovering the Relational Web. In *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB)*.

[7] Zhimin Chen, Vivek Narasayya, and Surajit Chaudhuri. 2014. Fast foreign-key detection in Microsoft SQL server PowerPivot for Excel. *PVLDB* 7, 13 (2014), 1417–1428.

[8] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y. Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. ACM, 817–828.

[9] Fabien De Marchi, Stéphane Lopes, and Jean-Marc Petit. 2009. Unary and n-ary inclusion dependency discovery in relational databases. *Journal of Intelligent Information Systems* 32, 1 (2009), 53–73.

[10] AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration*. Morgan Kaufmann.

[11] 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 *Proceedings of the International Semantic Web Conference (ISWC)*. Springer, 260–277.

[12] Lan Jiang and Felix Naumann. 2019. Holistic primary key and foreign key detection. *Journal of Intelligent Information Systems* (2019), 1–23.

[13] Oliver Lehmberg and Christian Bizer. 2019. Synthesizing N-ary Relations from Web Tables.. In *International Conference on Web Intelligence, Mining and Semantics*. 17:1–17:12.

[14] Oliver Lehmberg, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, and Christian Bizer. 2015. The Mannheim search join engine. *Web Semantics: Science, Services and Agents on the World Wide Web* 35 (2015), 159–166.

[15] Mozhgan Memari, Sebastian Link, and Gillian Dobbie. 2015. SQL data profiling of foreign keys. In *Proceedings of the International Conference on Conceptual Modeling (ER)*. Springer, 229–243.

[16] Thorsten Papenbrock and Felix Naumann. 2017. Data-driven Schema Normalization. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 342–353.

[17] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *PVLDB* 5, 10 (2012), 908–919.

[18] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching HTML tables to dbpedia. In *International Conference on Web Intelligence, Mining and Semantics*. 1–6.

[19] Yannis Sismanis, Paul Brown, Peter J Haas, and Berthold Reinwald. 2006. GORDIAN: efficient and scalable discovery of composite keys. In *Proceedings of the International Conference on Very Large Databases (VLDB)*. 691–702.

[20] Tommaso Soru, Edgard Marx, and Axel-Cyrille Ngonga Ngomo. 2015. ROCKER: A refinement operator for key discovery. In *Proceedings of the International World Wide Web Conference (WWW)*. International World Wide Web Conferences Steering Committee, 1025–1033.

[21] Fabian Tschirschnitz, Thorsten Papenbrock, and Felix Naumann. 2017. Detecting inclusion dependencies on very many tables. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 18.

[22] Yalin Wang and Jianying Hu. 2002. A machine learning based approach for table detection on the web. In *Proceedings of the International World Wide Web Conference (WWW)*. 242–250.

[23] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. 2012. InfoGather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 97–108.

[24] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M Procopiuc, and Divesh Srivastava. 2010. On multi-column foreign key discovery. *PVLDB* 3, 1-2 (2010), 805–814.

[25] Ziqi Zhang. 2017. Effective and Efficient Semantic Table Interpretation using TableMiner+. *Semantic Web* 8, 6 (2017), 921–957.

[26] Erkang Zhu, Fatemeh Nargesian, Ken Q Pu, and Renée J Miller. 2016. LSH Ensemble: Internet-scale domain search. *PVLDB* 9, 12 (2016), 1185–1196.