Result Selection and Summarization for Web Table Search

Nguyen Thanh Tam[*], Nguyen Quoc Viet Hung[*], Matthias Weidlich[†], Karl Aberer[*]

[*]*École Polytechnique Fédérale de Lausanne,* [†]*Imperial College London*

***Abstract*—The amount of information available on the Web has been growing dramatically, raising the importance of techniques for searching the Web. Recently, Web Tables emerged as a model, which enables users to search for information in a structured way. However, effective presentation of results for Web Table search requires (1) selecting a ranking of tables that acknowledges the diversity within the search result; and (2) summarizing the information content of the selected tables concisely but meaningful. In this paper, we formalize these requirements as the *diversified table selection* problem and the *structured table summarization* problem. We show that both problems are computationally intractable and, thus, present heuristic algorithms to solve them. For these algorithms, we prove salient performance guarantees, such as near-optimality, stability, and fairness. Our experiments with real-world collections of thousands of Web Tables highlight the scalability of our techniques. We achieve improvements up to 50% in diversity and 10% in relevance over baselines for Web Table selection, and reduce the information loss induced by table summarization by up to 50%. In a user study, we observed that our techniques are preferred over alternative solutions.**

I. Introduction

Handling the information explosion driven by the Web requires means to search data. Against this background, Web Tables [5] that are extracted from HTML tables or directly uploaded by users became a common structure for data on the Web. Their popularity is explained by the advantages of a tabular structure for collecting, maintaining, and making sense of data. Web Tables allow for a structured and condensed representation of data compared to free-format text, thereby enabling efficient storage and effective presentation of data. In addition, Web Tables support complex search scenarios, in which users are able to formulate structured queries that go beyond keyword-based search. Because of their benefits, Web Tables have been put forward by several systems, among them Google Webtables [5], YAGO [32], TableFinder [38], Factual [36], and Socrata [37].

The availability of corpora of billions of Web Tables [5, 29] calls for means for efficient search. To this end, techniques to find and rank Web Tables for a given user search query have been developed [5, 38]. However, the result of these techniques is typically a list of Web Tables ranked by their relevance, which is not appropriate for effectively browsing and exploring the data. On the one hand, data on the Web are often reused by copying and adapting existing sources [10, 24]. Hence, search results contain several tables that are highly relevant, but very similar in structure and content [30]. By including very similar Web Tables at the top of the result list, users actually lose information on the completeness and diversity of the result. On the other hand, Web Tables are large in size, up to hundreds or thousands of data tuples. Hence, effective exploration of the data requires support for quickly assessing the content of a table, ideally by selecting a few tuples that represent the original data. However, arbitrary selection of tuples to provide a table summary is not meaningful since it hides regularities in the data and, again, poses a problem with respect to completeness. That is, the diversity of data in a Web Table cannot be assessed effectively.

In this paper, we formalize the requirements for effective browsing and exploration of results for Web Table search as two problems, *diversified table selection* and *structured table summarization*, and present solutions for them.

We approach the problem of diversified table selection with a measure of the goodness of a selection. It combines the relevance scores obtained from Web Table search with measures for similarity of the schema and the data tuples of Web Tables, thereby accounting for diversity in the presented result. Diversified table selection is then defined as an optimization problem using the goodness measure. We show that this problem is NP-complete and, therefore, propose a greedy algorithm with several salient performance guarantees such as near-optimality, stability, and fairness. In particular, stability ensures that the table selection can be extended in a consistent manner to support incremental data exploration.

Our technique for table summarization selects a set of representative tuples of a table that induce little information loss with respect to non-selected tuples and preserve regularities of underlying data. Our technique exploits the similarity of data tuples as the basis to define an optimization problem over measures for information loss and data regularity. Since the problem turns out to be NP-complete, we present a heuristic approach that derives clusters of similar tuples and returns a representative sample of them. Again, stability is a particular feature of our approach: a table summary can be extended in size to support a user in drilling down into a search result.

In sum, this paper makes the following contributions.

- Section II: We first describe the background for our work in terms of a user interaction scheme for Web Table search and requirements for effective result presentation. Further, we present a formal model for our approach.
- Section III: We introduce the problem of diversified table selection based on a goodness function that unifies both, the relevance and the diversity of a ranked list of Web Tables. We prove that the problem is NP-complete, propose a heuristic algorithm, and prove that our solution is near-optimal, stable, fair, and efficient.
- Section IV: We introduce structured table summarization as a problem that refers to the identification of a subset of

data tuples with minimal information loss regarding non-selected tuples and maximal preservation of regularities in the data. We show that the problem is NP-complete and approximate its solution with a heuristic based on hierarchical clustering. We prove that our approximation is efficient and stable.
- Section V: We evaluate our techniques with real-world collections of thousands of Web Tables, showing that our techniques scale for large datasets. For table selection, we achieve improvements up to 50% in diversity and 10% in relevance over baselines. For table summarization, we reduce the information loss induced by a table summary by up to 50%. We further present a user study indicating that our techniques are preferred over alternative solutions, which confirms the effectiveness of our approach.

In the remaining sections, we review related work (Section VI), before we present conclusions (Section VII).

II. Search of Web Tables

To motivate our work, we first discuss a user interaction scheme for search over Web Tables and elaborate on requirements for table selection and table summarization. Then, we present a comprehensive formal model for Web Table search.

A. Motivation

User interaction scheme. Most systems for retrieving information from a corpus of Web Tables implement a user interaction scheme that relies on keyword-based search. For a keyword query, a system returns a list of tables that is typically presented as a list of hyperlinks. These pointers are sometimes enriched with information on the schema of the respective table, e.g., by listing the attributes of a table. Given the search results, a user then browses and explores the data to identify which subsets of the data sources best meet their information needs. A user iteratively follows the pointers to the Web Tables, scrolls over the data tuples to get an overview of the provided content, and returns to the list of search results to explore another source.

Requirements for table selection. The result quality of Web Table search clearly depends on the ability to select tables of high relevance to a user query and much work has been devoted to the computation of relevance scores, cf., [25]. However, selecting tables for presentation to the user purely based on relevance is not effective. The list of top retrieval results will be polluted with redundant data sources. Since Web data is often a copied and slightly modified, many tables in the result list will be similar in structure and content [10, 24].

Against this background, table selection should not only consider the relevance scores of tables, but also the diversity of the result list in order to support effective exploration of the data [8]. Diversification of a set of Web Tables is challenging since there is no a well-established objective function that defines what an optimal search result should be for a given query. Defining such a notion of optimality, on the one hand, requires consideration of many dimensions and types of diversity. On the other hand, the inherent trade-off between relevance and diversity needs to be made.

To achieve a selection of tables that balances relevance and diversity, we argue that a good selection result should satisfy the following requirements.

(R1) *Schema and instance diversity.* Web Tables from different sources show various types of heterogeneity. They may have no schema in common, assume different attribute semantics, or use noisy and ambiguous representations of attribute values. Hence, diversification of Web Tables should embrace both, the schematic structure as well as the data tuples.
(R2) *Consideration of table popularity.* Large groups of similar Web Tables in the search results hint at popular data that was frequently copied and modified. This implies a high chance to satisfy user information needs and, thus, should be considered when selecting tables.
(R3) *Avoidance of table redundancy.* Selection of very similar tables leads to high redundancy in the search result. Therefore, selection of a table should be done relative to other tables that have been selected.

We will later elaborate on a user study, which suggests that a technique satisfying these requirements is indeed preferred over a technique that does not.

Requirements for table summarization. Most systems for Web Table search present hyperlinks to data sources along with table headers. To effectively support a user in the exploration of the data and quickly skip irrelevant results, however, a concise representation of the table content is needed. Notably, Google Webtables [5] provide such a table summary by selecting a few data tuples. Nevertheless, such a summary is rarely meaningful. Tuples are selected arbitrarily or randomly, which prevents a user from assessing diversity and regularity of the data.

To effectively support the exploration of data stored in Web Tables, we argue that meaningful table summarization should satisfy the following requirements.

(R4) *Avoidance of tuple redundancy.* Selection of very similar tuples leads to high redundancy in a table summary. Thus, the decision about the selection of a tuple should be based on the set of already selected tuples.
(R5) *Highlighting of data regularities.* Presenting the tuples of a table summary in random or arbitrary order hides data regularities. Hence, table summarization should provide a user with clues about the characteristics of the data by ordering them in a structured way.
(R6) *Multi-resolution support.* Table summaries should be flexible in size to support data exploration by drilling down into a search result without turning to the original data source. Here, it is crucial to create summaries that can be extended consistently, i.e., by selecting additional data tuples without altering the already presented tuples.

Again, the results of our user study indicate these requirements indeed give rise to technique that is preferred by users.

Example. To illustrate the requirements for table selection and table summarization, Fig. 1 depicts the result for a search for Web Tables on standard of living indices. There are three relevant data sources, each given as a table with different attributes. Here, the relevance ranking corresponds to the table indices, i.e., T_1 is most relevant and T_3 is third-most relevant.

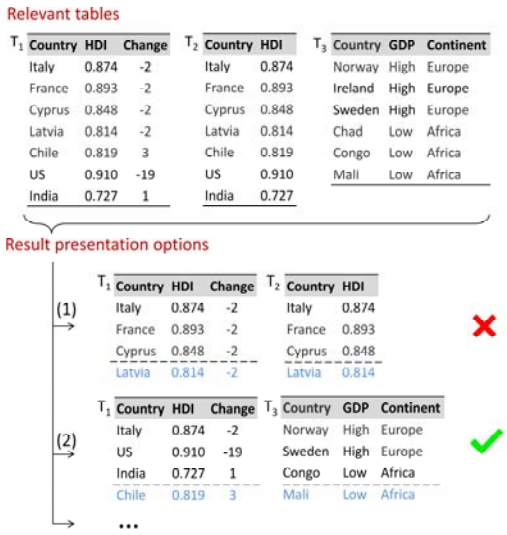

Fig. 1: Example scenario for Web Table search

Suppose that a user should be presented with two tables as a search result (e.g., on the first result page) and each of them is to be summarized by three tuples. Fig. 1 depicts two of the various selection and summarization options.

Option (1) selects tables T_1 and T_2 and provides summaries that are built of the first three rows of each table. This solution is problematic for two reasons. First, the two selected tables are very similar in structure and content, so that a user gets an incomplete view on the results on living indices. Second, the summaries hide a lot of the data of non-selected tuples (e.g., only value '-2' is considered for attribute Change). This issue persist even if the summary is extended by another tuple.

Option (2), in turn, provides a more diversified table selection, covering not only information on the Human Development Index (HDI), but also on the Gross Domestic Product (GDP) as different measures for the standard of living. In addition, the table summaries hide less information on non-selected tuples (e.g., three different values are considered for attribute Change) and hint at regularities of the data (e.g., regarding attributes GDP and Continent). Also, changing the resolution of the table summary and adding a data tuple to the summaries indeed yields a more complete picture of the underlying data (e.g., including another value for attribute Change). As such, option (2) illustrates how table selection and table summarization that meet the aforementioned requirements provide a much more effective presentation of the search results.

B. Model

We based our techniques on a compact model for search of Web Tables. We denote a universal corpus of Web Tables by $\mathcal{U}$. Each table $T \in \mathcal{U}$ is defined as a tuple $T = \langle S, D \rangle$ with S being a set of attributes $S = \{a_1, \ldots, a_n\}$ and $D = \{d_1, \ldots, d_m\}$ being a set of data tuples (i.e., rows of the table). The set of attributes S represents meta-data and defines the table *schema*. A *data tuple* (or *data instance*) $d \in D$ is a function $d : S \to \mathcal{V}$ from the attributes S to some domain $\mathcal{V}$ of attribute values.

Most techniques for search of Web Tables assign a relevance score to each table in order to rank the search results. We capture these relevance scores as a function $r : \mathcal{U} \to [0, 1]$.

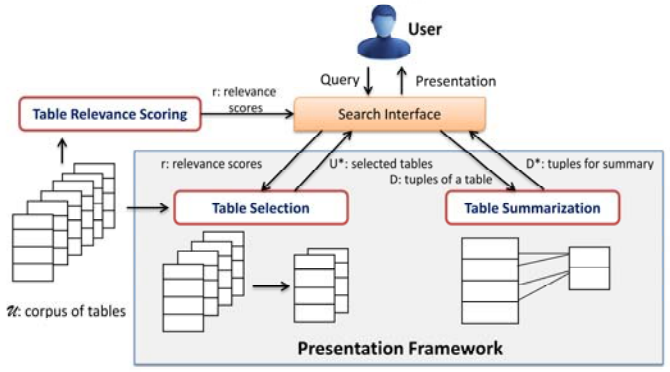

Fig. 2: Overview of the approach

We denote a sequence of n tables by $U^* = \langle T_1, \ldots, T_n \rangle$, $T_i \in \mathcal{U}$ for $1 \leq i \leq n$. For a table $T = \langle S, D \rangle$, a sequence of n tuples is denoted by $D^* = \langle d_1, \ldots, d_n \rangle$, $d_i \in D$ for $1 \leq i \leq n$. For both types of sequences, we use . as a concatenation operator. We overload set notation for sequences, meaning that set operators applied to a sequence are evaluated based on the set of sequence elements. For instance, $T \in U^*$ means that T is part of the set of elements of sequence U^* and $|U^*|$ is the number of unique elements (i.e., the length of sequence $|U^*|$ if all its elements are distinct).

Using this model, Fig. 2 gives an overview of the approach taken in this work. Given a user query, in a first step, the relevance scores (r) for a corpus of Web Tables ($\mathcal{U}$) are computed, e.g., using the techniques presented by Cafarella et al. [5] or Venetis et al. [38]. Based on the derived relevance scores, we define table selection and table summarization as the two steps conducted to support a user in effective exploration and browsing of the search results:

(1) Table selection: Select a sequence of distinct tables $U^* = \langle T_1, \ldots, T_k \rangle$, $T_i \in \mathcal{U}$ for $1 \leq i \leq k$ that is significantly smaller than the corpus, $k \ll |\mathcal{U}|$. Here, selection of the top-k results is of particular practical relevance for information retrieval, cf., [25]. An appropriate value for k depends on the user and the application context.

(2) Table summarization: For a table $T = \langle S, D \rangle$, we select a sequence of distinct data tuples $D^* = \langle d_1, \ldots, d_k \rangle$, $d_i \in D$ for $1 \leq i \leq k$ that is significantly smaller than the set of all tuples, $k \ll |D|$. Selection of the top-k results is particularly relevant and well-investigated for answering database queries [3]. The value of k is also context specific.

Note that a single search involves one execution of table selection, but many executions of table summarization. On the one hand, a summary is derived for each of the selected tables. On the other hand, users may drill down into a search result by adapting the size of the table summary.

III. Diversified Table Selection

This section proposes a technique for diversified table selection that satisfies the requirements outlined in Section II-A. To achieve diversification that embraces both schema and instances (requirement R1), we first present similarity measures of Web Tables that relate to the schematic structure as well as the data tuples (Section III-A). Then, Section III-B defines a measure of goodness for a search result that takes into

consideration the table popularity (R2) and table redundancy (R3). Using this measure, we define diversified table selection as an optimization problem and prove its NP-completeness (Section III-C). Finally, we provide a heuristic algorithm to approximate the optimal solution that comes with several salient performance guarantees (Section III-D).

A. Table Similarity

As outlined earlier, heterogeneity of Web Tables that stem from different sources is observed along several dimensions. Examples include the absence of a common schema, different semantics of attributes, or noisy and ambiguous representations of attribute values. To cater for these dimensions of heterogeneity, we first define measures for *schema similarity* and *data similarity*, before deriving a combined measure.

Schema Similarity. We compute the schema similarity of two tables by matching their sets of attributes. To this end, we exploit algorithms for schema matching [2] that construct a set of weighted attribute correspondences. Technically, for two schemas of tables, S_1 and S_2, schema matching first constructs an $|S_1| \times |S_2|$ similarity matrix over $S_1 \times S_2$, denoted by $m(S_1, S_2)$, where $m_{i,j}(S_1, S_2) \in [0, 1]$ represents a degree of similarity between the attributes i and j of S_1 and S_2, respectively. A variety of specific measures for attribute similarity has been proposed, including string-edit distances, vector-based metrics and language models, cf., [2]. Schema matching then uses the similarity matrix to extract a set of attribute correspondences $C_{1,2} \subseteq (S_1 \times S_2)$, e.g., by solving the maximum weighted bipartite sub-graph problem [26].

Following this line, we define schema similarity of two Web Tables $T_1 = \langle S_1, D_1 \rangle$ and $T_2 = \langle S_2, D_2 \rangle$ as an aggregation over the similarity of attribute correspondences in the matching:

$$ssim(T_1, T_2) = \frac{\sum_{(i,j) \in C_{1,2}} m_{i,j}(S_1, S_2)}{\max(|S_1|, |S_2|)} \quad (1)$$

Data Similarity. We evaluate similarity of data instances of Web Tables by matching attribute values. Similarity measures for attributes values have also been investigated in the field of schema matching. However, most of these techniques exploit data type similarities [2]. For Web Tables, such type information is rarely available and we have to rely on measures for textual similarity as they are the basis for searching Web documents [4]. To reflect the tabular structure of Web Tables, our approach measures textual similarity by column, i.e., considers all values related to a certain attribute.

To implement this idea for an attribute $a \in S$ of some Web Table $T = \langle S, D \rangle$, we first extract all attribute values, given as $\{d(a) | d \in D\}$. We treat all these values as strings and apply basic techniques for textual pre-processing (splitting by delimiters, stop word removal, case normalization) yielding a set of terms $K(a, D)$ representing the instances of attribute a in table T. By applying this approach to all attributes, we obtain a term corpus for table T, $K(S, D) = \bigcup_{a \in S} K(a, D)$. For two tables $T_1 = \langle S_1, D_1 \rangle$ and $T_2 = \langle S_2, D_2 \rangle$, the union of these corpora is defined as $K(1, 2) = K(S_1, D_1) \cup K(S_2, D_2)$.

The actual similarity assessment is grounded on binary feature vectors $L(a, k)$ of length $|K(1, 2)|$, which indicate for each term $k \in K(1, 2)$ whether the instances of attribute $a \in S_1 \cup S_2$ contain a term that is similar to k ($L(a, k) = 1$) or not ($L(a, k) = 0$). To decide whether there exists such a similar term, we again rely on the large set of textual similarity measures proposed in the literature, cf., [2] and apply a similarity threshold.

Given two attributes a_1 and a_2 of tables T_1 and T_2, respectively, the similarity of their data instances is measured as the normalized product of their feature vectors:

$$s(a_1, a_2) = \frac{\sum_{k=1}^{|K(1,2)|} L(a_1, k) L(a_2, k)}{\sqrt{\sum_{k=1}^{|K(1,2)|} (L(a_1, k))^2} \sqrt{\sum_{k=1}^{|K(1,2)|} (L(a_2, k))^2}} \quad (2)$$

Based on the similarity of data instances per attribute, we derive the data similarity of two Web Tables. To account for the high number of possible combinations between attributes of two tables, we aggregate over the maximum similarity scores per attribute of either table. Then, data similarity of Web Tables $T_1 = \langle S_1, D_1 \rangle$ and $T_2 = \langle S_2, D_2 \rangle$ is defined as follows:

$$dsim(T_1, T_2) = \frac{1}{2} \left(\sum_{a_1 \in S_1} \max_{a_2 \in S_2} s(a_1, a_2) + \sum_{a_2 \in S_2} \max_{a_1 \in S_1} s(a_1, a_2) \right) \quad (3)$$

Combined Similarity. To achieve a comprehensive similarity assessment of Web Tables, we rely on a weighted average of schema similarity and data similarity, denoted by M:

$$M(T_1, T_2) = \alpha \cdot ssim(T_1, T_2) + \beta \cdot dsim(T_1, T_2) \quad (4)$$

Parameters $\alpha, \beta \geq 0$ allow for tuning the importance of schema similarity and data similarity. It may be chosen based on prior knowledge about the reliability of either type of information. Without this, we set $\alpha = \beta = 0.5$ for normalization.

B. Goodness of Table Selection

To balance relevance and diversity in a top-k selection of Web Tables, we design a goodness measure for such a selection. Clearly, selection is driven by the given relevance scores of tables. However, taking up the requirements identified in Section II-A, the goodness measure should incorporate table popularity (R2) and table redundancy (R3) in the selection. Therefore, we define goodness of a selection of tables based on the overall, weighted relevance of a selected table, which is reduced by the relevance of similar tables that have also been selected. Intuitively, this approach favors examples from groups of similar tables, but penalizes the selection of multiple relevant tables that are very similar to each other.

Let $\mathcal{U}$ be a corpus of tables, r a relevance ranking, and $U^* = \langle T_1, \ldots, T_n \rangle$, $T_i \in \mathcal{U}$, $1 \leq i \leq n$ a selection. Then, we define $q(T) = \sum_{T' \in \mathcal{U}} M(T, T') r(T')$ as the importance of table $T \in \mathcal{U}$ in the corpus given the relevance ranking. In practice, the size of the corpus typically renders exact computation of the importance impossible. Therefore, $q(T)$ will be calculated based on a subset $\mathcal{U}' \subseteq \mathcal{U}$ with $|\mathcal{U}'| \ll |\mathcal{U}|$. Existing techniques to find and rank Web Tables typically extract such a subset of tables for which the relevance score exceeds a threshold.

With $w \in R^+$ as a positive weight parameter, we define goodness as follows (overloading set notation as detailed in Section II-B):

$$g_{\mathcal{U},r}(U^*) = w \sum_{T \in U^*} q(T)r(T) - \sum_{T_1,T_2 \in U^*} M(T_1,T_2)r(T_1)r(T_2) \quad (5)$$

While our notion of goodness is motivated by the need to consider table popularity and table redundancy in the selection, it also shows several intuitive properties that are detailed below. The respective proofs can be found in the appendix.

The first property considers the influence of the relevance scores. We observe that the more relevant a table is, the higher are the chances of it to be part of the selection.

Proposition 1 (Strength of Relevance)**.** *Let $\mathcal{U}$ be a corpus of tables, r a relevance ranking, $U^* = \langle T_1, \ldots, T_n \rangle$, $T_i \in \mathcal{U}$, $1 \leq i \leq n$ a selection, and $T \in (\mathcal{U} \setminus U^*)$ a non-selected table. Let r' be a relevance score defined such that $r'(T) > r(T)$ and $r'(x) = r(x)$ for $x \in (\mathcal{U} \setminus \{T\})$. For $w \geq 2$ it holds that:*

$$g_{\mathcal{U},r'}(U^*.T) \geq g_{\mathcal{U},r}(U^*.T)$$

Our notion of goodness further shows monotonicity. That is, when adding more tables to an existing selection, the goodness of the overall selection will increase.

Proposition 2 (Monotonicity)**.** *Let $\mathcal{U}$ be a corpus of tables, r a relevance ranking, $U^* = \langle T_1, \ldots, T_n \rangle$, $T_i \in \mathcal{U}$, $1 \leq i \leq n$ a selection, and $U' \subseteq (\mathcal{U} \setminus U^*)$ a set of non-selected tables. For $w \geq 2$ it holds that:*

$$g_{\mathcal{U},r}(U^*.U') \geq g_{\mathcal{U},r}(U^*)$$

Finally, our goodness measures shows submodularity, which refers to the property that marginal gains in goodness start to diminish due to saturation of the objective. That is, the marginal benefit of adding tables to the selection decreases w.r.t. the size of the selection.

Proposition 3 (Submodularity)**.** *Let $\mathcal{U}$ be a corpus of tables, r a relevance ranking, $U^* = \langle T_1, \ldots, T_n \rangle$, $T_i \in \mathcal{U}$, $1 \leq i \leq n$ a selection, and $T, T' \in (\mathcal{U} \setminus U^*)$ non-selected tables. Then, it holds that:*

$$g_{\mathcal{U},r}(U^*.T) + g_{\mathcal{U},r}(U^*.T') \geq g_{\mathcal{U},r}(U^*.T.T') + g_{\mathcal{U},r}(U^*)$$

C. Diversified Table Selection Problem

Using the notion of goodness, diversified table selection is defined as an optimization problem. That is, we are interested in finding a selection of top-k tables with maximal goodness.

Problem 1 (Diversified Table Selection)**.** *Let $\mathcal{U}$ be a corpus of tables, r a relevance ranking, and k a threshold for the number of tables. Then, the* diversified table selection *problem is defined to be:*

$$\underset{U^* = \langle T_1, \ldots, T_k \rangle, T_i \in \mathcal{U}, 1 \leq i \leq k}{\operatorname{argmax}} g_{\mathcal{U},r}(U^*) \quad (6)$$

Diversified table selection defines a notion of optimality for table selection. Yet, the problem turns out to be intractable.

Theorem 1. *Diversified table selection is NP-complete.*

Proof (Sketch): We prove the theorem by reduction from the Densest k-Subgraph problem, which is known to be NP-complete [16, 19]. ■

Algorithm 1: Heuristic for diversified table selection.

input : A corpus of tables $\mathcal{U}$ with relevance function r,
a measure for table similarity M,
a weight factor $w \geq 2$, and a threshold for the number of tables k.
output: A selection of tables $U^* = \langle T_1, \ldots, T_k \rangle, T_i \in \mathcal{U}, 1 \leq i \leq k$.

1 $U^* \leftarrow \langle\rangle$;
// Compute ranking score for each table in the corpus
2 Let $s : \mathcal{U} \to R$, $s(T) \mapsto w \cdot r(T) \cdot \sum_{T' \in \mathcal{U}} M(T,T')r(T')$;
3 **while** $|U^*| < k$ **do**
4 $T_m \leftarrow \operatorname{argmax}_{T \in \mathcal{U}, T \notin U^*} s(T)$;
5 $U^* \leftarrow U^*.T_m$;
// Update ranking score for the remaining tables
6 $s' \leftarrow s$;
7 Let $s : \mathcal{U} \to R$, $s(T) \mapsto s'(T) - 2 \cdot r(T_m) \cdot M(T,T_m) \cdot r(T)$;
8 **return** U^*

D. Heuristic Diversified Table Selection

Given that diversified table selection is intractable, below, we present a heuristic algorithm to approximate its solution, for which we prove various performance guarantees.

Heuristic Algorithm. Our algorithm exploits two of the aforementioned properties of the goodness function $g_{\mathcal{U},r}$, i.e., monotonicity and submodularity, to achieve a provably near-optimal solution. The algorithm iteratively expands the selection of tables by adding the table that maximizes the objective function. Thus, solving the problem requires k iterations.

The details of our heuristic are given in Algorithm 1. It takes a set corpus of tables $\mathcal{U}$ with relevance function r, a measure for table similarity M, a weight factor w, and a threshold for the number tables k as input and returns a selection U^* of k tables. We begin by computing a ranking score for each table $T \in \mathcal{U}$ that is based on the weight factor, the table relevance, and the table importance (line 2). In the actual greedy selection step, we select k tables. In each iteration, we add the table with the highest ranking score (lines 4 and 5), before the ranking score is updated for the remaining tables (line 7). The latter avoids re-computation of the ranking scores from scratch in each iteration. As detailed in Section III-B, in practice, the selection procedure is typically not applied over the whole corpus, but a subset $\mathcal{U}' \subseteq \mathcal{U}$ with $|\mathcal{U}'| \ll |\mathcal{U}|$ as it is extracted by existing techniques to find and rank Web Tables.

Algorithm Analysis. First, we observe that the approximation error of the proposed algorithm is bounded.

Guarantee 1 (Near-Optimality)**.** *Algorithm 1 is a $(1 - 1/e)$-approximation for diversified table selection.*

Proof: For any monotone, submodular function f with $f(\emptyset) = 0$ it is known that an iterative algorithm selecting the element e with maximal value of $f(S \cup \{e\}) - f(S)$ with S as the elements selected so far has a performance guarantee of $(1 - 1/e) \approx 0.63$ [28]. This result is applicable to Algorithm 1, since our goodness function $g_{\mathcal{U},r}$ is monotonic (Proposition 2) and submodular (Proposition 3), it holds $g_{\mathcal{U},r}(\emptyset) = 0$ (Eq. (5)), and the ranking score is defined as $s(T) = g_{\mathcal{U},r}(U^*.T) - g_{\mathcal{U},r}(U^*)$ (lines 2 and 7). ■

Next, we consider the complexity of our heuristic.

Guarantee 2 (Complexity)**.** *The time and space complexity of Algorithm 1 are $\mathcal{O}(|\mathcal{U}|^2 + k|\mathcal{U}|)$ and $\mathcal{O}(|\mathcal{U}|^2)$, respectively.*

Proof: Time complexity: The quadratic term $|\mathcal{U}|^2$ stems from the computation of the ranking score. The linear term $k|\mathcal{U}|$ is explained by k iterations, in each of which we iterate over all remaining tables, for selection of T_{max} and for updating the ranking score.

Space complexity: Storing table similarities requires $\frac{|\mathcal{U}||\mathcal{U}-1|}{2}$ space since M is symmetric and $M(T,T)$ is fixed. ■

Existing techniques for Web Table search return a subset $\mathcal{U}' \subset \mathcal{U}$ with $|\mathcal{U}'| \ll |\mathcal{U}|$ since users cannot exhaustively explore thousands of tables. Thus, time and space complexity is, respectively, $\mathcal{O}(|\mathcal{U}'|^2 + k|\mathcal{U}'|)$ and $\mathcal{O}(|\mathcal{U}'|^2)$, which is tractable in real world datasets as we will demonstrate in our evaluation.

Further, our algorithm shows stability in the selection, which is important to support incremental data exploration. If a user is first presented with the top-10 tables, but then extends the result to the top-20, the expectation is clearly that the top-10 remain unchanged.

Guarantee 3 (Stability)**.** *For U^* as returned by Algorithm 1, let $U^*_{k_1} = \langle T_1, \ldots, T_{k_1} \rangle, U^*_{k_2} = \langle T'_1, \ldots, T'_{k_2} \rangle$ be selections with $T_i \in U^*$, $1 \leq i \leq k_1$, $T'_j \in U^*$, $1 \leq j \leq k_2$, and $0 < k_1 \leq k_2$. Then, it holds that $T_i = T'_i$ for $1 \leq i \leq k_1$.*

Proof: In Algorithm 1, the construction of U^* is performed stepwise and elements are never removed from U^*. The selection also is deterministic: we always add the table with highest ranking score (line 4). Thus, a larger selection sequence comprises a smaller selection sequence as a prefix. ■

Finally, we also highlight that the selection heuristic is fair in the sense that it is genuinely driven by the relevance function.

Guarantee 4 (Fairness)**.** *Let $\mathcal{U}$ be a corpus of tables. For any set of tables $U \subset \mathcal{U}$, there exists a relevance function r, s.t. Algorithm 1 returns $U^* = \langle T_1, \ldots, T_{|U|} \rangle$, $T_i \in U, 1 \leq i \leq |U|$.*

Proof: Given U, we define r as $r(T) = 1$ if $T \in U$ and $r(T) = 0$ otherwise. Then, the ranking score $s(T)$ is positive if $T \in U$, whereas $s(T) = 0$ if $T \notin U$. Hence, the algorithm selects only elements from U. ■

IV. Table Summarization

This section takes up the requirements on table summarization outlined in Section II-A and presents a technique to derive concise and meaningful summaries of tables. To avoid tuple redundancy (requirement R4) and highlight data regularities (R5) in the result, we first define a notion of representativeness for table summaries (Section IV-A). Intuitively, the selected tuples should be similar to non-selected tuples. The presented measure of representativeness is then used to formulate structured table summarization as an optimization problem (Section IV-B). The problem turns out to be intractable, so that we also propose a clustering-based algorithm to approximate the solution (Section IV-C). We prove that the algorithm shows stability when increasing the size of the summary, thereby supporting multi-resolution summaries (R6).

A. Representativeness of Table Summaries

Representativeness of table summaries is grounded in two dimensions, *information loss* and *data regularity*, that follow directly from the aforementioned requirements. The former relates to the amount of information that is not captured by a table summary, i.e., the loss of information induced by non-selected tuples with respect to the entire data of a table. Data regularity, in turn, reflects the ability to provide a user with clues about the characteristics of the table data. As such, it serves as a reverse proxy for the tuple redundancy in the table summary. Data regularity, in turn, reflects the ability to provide a user with clues about the characteristics of the table data.

Against this background, information loss guides which tuples *to select* for a table summary and data regularities are exploited *to order* the selected tuples.

TABLE I: An exemplary Web Table

	ID	Country	GDP	Continent
∗(I)	1	United States	High	America
	2	Mali	Low	Africa
∗(II)	3	Mexico	Medium	America
∗(III)	4	Ireland	High	Europe
∗(IV)	5	Sweden	High	Europe
∗(V)	6	Congo	Low	Africa
	7	Canada	High	America
	8	Norway	High	Europe
∗(VI)	9	Chad	Low	Africa

As an example, consider Table I. Assuming that six tuples should be selected for a summary, the tuples marked with '$*$' would form a representative selection. This set has low information loss since it covers all possible values of attributes *GDP* and *Continent*. Further, ordering the tuples as indicated by the Roman numerals (I)-(VI) supports discovery of data regularities. For instance, the highlighted relation between GDP indexes and continents becomes visible once tuples with equal continent values are listed close to each other.

To quantify information loss and data regularity, we assess the similarity of tuples. In contrast to data similarity discussed in Section III-A that compares attribute values of different tables, tuple similarity compares two tuples in isolation, based on their values. Technically, tuple similarity is a function $tsim : D \times D \rightarrow [0,1]$ over the tuples of a table $T = \langle S, D \rangle$. To instantiate this function, we rely on similarity measures as proposed in the literature, e.g., the Jaccard similarity coefficient over 3-grams of the attribute values [34].

Information Loss. Using a notion of tuple similarity, information loss of a table summary is defined as follows. Let $T = \langle S, D \rangle$ be a table and $D^* = \langle d_1, \ldots, d_n \rangle$, $d_i \in D$, $1 \leq i \leq n$, a summary of T. The information loss of D^* is defined as the sum of loss scores of all non-selected tuples, where the loss score of a non-selected tuple is its dissimilarity with respect to the most similar selected tuple:

$$IL(D^*) = \sum_{d \in (D \setminus D^*)} 1 - \max_{d' \in D^*} tsim(d, d') \tag{7}$$

It holds that $IL(\emptyset) = |D|$ and $IL(D) = 0$.

Data Regularity. Let $D^* = \langle d_1, \ldots, d_n \rangle$, $d_i \in D$, $1 \leq i \leq n$ be a summary of a table $T = \langle S, D \rangle$. Then, we define data regularity for the summary as the sum of similarities of consecutive pairs:

$$DR(D^*) = \sum_{i=1}^{n-1} tsim(d_i, d_{i+1}) \tag{8}$$

The measure for data regularity ensures that tuples are presented in a meaningful order rather than randomly or arbitrarily.

B. Structured Table Summarization Problem

To avoid tuple redundancy and highlight data regularities in a table summary, we define the structured table summarization problem. It refers to the identification of a summary, such that tuples with minimal information loss are selected and their ordering maximizes data regularity. Clearly, information richness of the summary is most relevant for Web Table search, so that minimization of information loss is prioritized.

Problem 2 (Structured Table Summarization)**.** *Let $T = \langle S, D \rangle$ be a table and k a threshold for the size of the summary. The structured table summarization problem is the identification of a sequence of distinct tuples $D^* = \langle d_1, \ldots, d_k \rangle$, $d_i \in D$, $1 \leq i \leq k$ that satisfies the following conditions in decreasing priority:*

(1) Minimal information loss: there is no other selection $D' = \langle d'_1, \ldots, d'_k \rangle$, $d'_i \in D$, $1 \leq i \leq k$ s.t. $IL(D') < IL(D^*)$.
(2) Maximal data regularity: there is no permutation D' of D^* such that $DR(D') > DR(D^*)$.

Unfortunately, already the minimization of information loss requires investigating all subsets of size k of tuples of a Web Table, which is intractable. In particular, we observe that the decision of whether there is a subset with information loss less than a threshold is NP-complete.

Theorem 2. *Let $T = \langle S, D \rangle$ be a table. The problem of deciding whether there exists a sequence of distinct tuples $D^* = \langle d_1, \ldots, d_k \rangle$, $d_i \in D$, $1 \leq i \leq k$ of length k with information loss less than a positive number δ is NP-complete.*

Proof (Sketch): We prove the theorem by using the *restriction* technique [17] and show that k-medoids clustering, which is known to be NP-complete [21], is a special case of our problem. A tuple can be represented by a data point and the dissimilarity between two tuples is the distance between two data points. Hence, the information loss in Eq. (7) is equivalent to the objective function of the k-medoids clustering problem. ■

C. Heuristic Table Summarization

To cope with the inherent complexity of structured table summarization, we propose a heuristic algorithm based on clustering of similar tuples. Intuitively, we select a few tuples from each cluster, so that the selection will have low information loss since tuples that very similar to the selected ones can be expected to be part of the same cluster. Then, by placing selected tuples that are similar close to one another in the result sequence, we achieve high data regularity.

We implement this approach using agglomerative hierarchical clustering [20] – a connectivity-based clustering technique. Using agglomerative hierarchical clustering has several advantages compared to other connectivity-based clustering methods, such as k-medoids or k-means. First, k-medoids and k-means are sensitive to initialization and the choice of parameter k. Second, the cluster center obtained using k-means may not be an actual tuple of the table. Third, hierarchical clustering ensures stability of the result when increasing the solution size. In our setting, this property is important to support multi-resolution table summaries (requirement R6), so that a user can drill down into search results by consistently extending the summary.

Algorithm 2: Clustering-based Table Summarization

input : A set of tuples D,
a measure for tuple similarity $tsim$,
a threshold for the number of tuples k.
output: A sequence of tuples $D^* = \langle d_1, \ldots, d_k \rangle$, $d_i \in D$, $1 \leq i \leq k$.

```
// Step 1: Clustering -- Build a cluster tree
1  T ← agglomerativeHierarchicalClustering(D, tsim);
// Step 2: Ordering -- Reorder the tree
2  foreach pair of node siblings (n, ns) of T do
       // Linkage distance function
3      l ← distance(ns, getLeftChild(n));
4      r ← distance(ns, getRightChild(n));
       // Make children of n nearer to ns
5      if (isLeftChild(n) ∧ l < r) ∨ (isRightChild(n) ∧ l > r) then
6          swap(getLeftChild(n), getRightChild(n));
// Step 3: Selection -- Find k representative tuples
7  D* ← ⟨⟩ ;
   // Cut tree horizontally, get k max heights nodes
8  N ← cutoffKNodes(T, k) ;
   // Select leaves that gain most information
9  foreach n ∈ N in topological order do
10     if isLeaf(n) then D* ← D*.n ;
11     else
12         l* ← argmin_{l ∈ getAllLeaves(n))} IL(D*.l) ;
13         D* ← D*.l* ;
14 return D*
```

Heuristic Algorithm. The idea of our algorithm is summarized as follows: We first build a tree that provides a hierarchy of clusters that merge with each other at certain distances. The distance between two clusters is derived from the dissimilarity values of their tuples. We then flip nodes in this tree to bring similar tuples (leaves of the tree) closer to each other. Finally, we traverse the tree top-down to select k representative leaves that minimize information loss.

As outlined in Algorithm 2, our approach takes a set of tuples D, a measure for tuple similarity $tsim$, and a threshold for the number of tuples k as input and returns a selection D^* of k tuples. The algorithm involves three steps: *clustering*, *ordering*, and *selection*.

As a first step, we run hierarchical clustering on set D to obtain a cluster tree, using a linkage criterion such as single linkage, complete linkage, or average linkage [33]. Second, we move similar tuples (tree leaves) close to each other to maximize data regularity. To this end, we traverse the tree and flip nodes. More precisely, for each left node, if its left child is more similar to its sibling than its right child, we swap its children; and vice-versa (line 6). Third, we select a sequence of k leaf nodes and attempt to minimize information loss. We find a cut-off point by scanning the tree from the root to the leaves (line 8). The cut-off point is reached in that scan when the number of clusters is exactly k. For each cluster obtained, we select the leaf for which inclusion minimizes information loss (lines 9 to 13), which is prioritized.

Algorithm Analysis. The proposed algorithm shows several desired properties. First, we consider time complexity.

Guarantee 5 (Complexity)**.** *The time complexity of Algorithm 2*

TABLE II: Real-world collections of Web Tables

Domain	#Web Tables	Avg. #Rows	Avg. #Attributes
Climate	5907068	75.18	4.01
Country	6232132	172.44	4.88
Location	17983748	108.05	4.71
People	463443	48.36	4.94
Organization	6666436	123.76	4.01
Company	6451838	64.29	5.49
Phone	2435812	78.43	4.52
City	7157037	143.02	5.55
Fast Food	225752	57.45	4.10

is $\mathcal{O}(|D|^2 \log |D|)$.

Proof: Clustering takes $\mathcal{O}(|D|^2 \log |D|)$ time [27]. The ordering step visits each tree node once and potentially swaps the children, which needs $\mathcal{O}(|D|)$ time. The cut-off point for selection is determined in $\mathcal{O}(|D| \log |D|)$ time [22]. The final loop takes $\mathcal{O}(|D| - k)$ time. Thus, the overall time complexity is $\mathcal{O}(|D|^2 \log |D|)$. ■

There are several optimizations that avoid building the full cluster tree [12]. However, these techniques compromise the stability of the approach as established by the next property, so that multi-resolution table summaries would not be supported.

Guarantee 6 (Stability)**.** *For D^* as returned by Algorithm 2, let $D^*_{k_1} = \langle d_1, \ldots, d_{k_1} \rangle, D^*_{k_2} = \langle d'_1, \ldots, d'_{k_2} \rangle$ be summaries with $d_i \in D^*$, $1 \leq i \leq k_1$, $d'_j \in D^*$, $1 \leq j \leq k_2$, and $0 < k_1 \leq k_2$. Then, for all indices i, i', $1 \leq i < i' \leq k_1$ there are indices j, j', $i \leq j < j' \leq k_2$, s.t. $d_i = d'_j$ and $d_{i'} = d'_{j'}$.*

Proof: A tree obtained by hierarchical clustering shows stability w.r.t. the clusters [31]. The selection step of Algorithm 2 does not change the tree and is greedy regarding the minimization of information loss. Hence, increasing the number of clusters (lowering the cut-off point) can only extend the selection. Since the nodes are considered in topological order, the order of tuples in the sequence is preserved. ■

V. Experimental Evaluation

We evaluated our methods using a large collection of real-world Web Tables. Our experiments show that our approach leads to improvements over baselines of up to 50% in diversity and 10% in relevance for table selection, and reduces information loss induced by table summarization by up to 50%. In a user study, we further observed that participants preferred the proposed techniques over alternative solutions.

A. Experimental Setting

Datasets and Setup. We conducted our experiments using a large real-world collection of Web Tables [35] covering nine domains. The dataset is summarized in Table II. Each domain comes with subtopics (20 on average, 180 in total) that we used as keywords for Web Table search.

Unless otherwise stated, we run experiments on the 1000 tables with the highest relevance scores. This is a realistic setup since arguably, users cannot explore all tables in the search result. Results are computed by taking the average over all queries and all domains. All experiments ran on an Intel Core i7 system (2.8GHz, 4GB RAM). We used the COMA++ [1] schema matching system to derive attribute similarity matrices and constructed binary feature vectors (cf., Section III-A) using the Longest-Common-Substring measure. Tuple similarity was assessed with the Jaccard similarity coefficient over 3-grams of the string representation of the data.

Evaluation Measures. We use the following measures:

Subtopic recall (S-Recall). A popular metric to evaluate diversity of search results is subtopic recall [41]. A table may cover many subtopics, so that a set of tables is diverse if it contains many subtopics. For a query q, the metric measures the proportion of unique subtopics retrieved in the result U^*:

$$S\text{-}Recall(q, U^*) = \frac{|\cup_{T \in U^*} subtopics(T)|}{subtopics(q)}$$

Normalized relevance. This metric measures the relevance of the diversified result set w.r.t. the top-k relevant set returned by search engines [41], i.e, it indicates how well a diversification algorithm preserves relevance when diversifying the result. Formally, normalized relevance ($\in [0, 1]$) of a top-k table selection U^* from corpus $\mathcal{U}$ is defined as the sum of selected relevance scores over the sum of the k highest relevance scores:

$$nR(\mathcal{U}, r, U^*) = \frac{\sum_{T \in U^*} r(T)}{\max_{U = \langle T_1, \ldots, T_{|U^*|} \rangle, T_i \in \mathcal{U}, 1 \leq i \leq |U^*|} \sum_{T \in U} r(U)}$$

Here, a higher score indicates higher relevance and $nR(\mathcal{U}, r, U^*) = 1$ means that U^* is exactly the selection of tables with the highest relevance scores.

Normalized Information Loss. To evaluate the quality of table summarization, one cannot use the information loss defined in Eq. (7) since the measure is not scaled and changing the problem parameters (table size, parameter k) leads to incomparable values. Hence, we define normalized information loss as a comparison of a top-k table summary D^* with a baseline D_0 that is obtained by random sampling of k tuples:

$$nIL(D^*, D_0) = \frac{IL(D_0)}{IL(D^*)}$$

Here, a value of larger than one means that D^* suffers less from information loss than D_0, and vice-versa.

Normalized Data Regularity. Following the same reasoning, we also define normalized data regularity to evaluate table summarization. For the top-k table summary D^* and D_0 as the baseline obtained by random sampling, we have:

$$nDR(D^*, D_0) = \frac{DR(D^*)}{DR(D_0)}$$

Again, a value larger than one means that D^* performs better in terms of data regularity than D_0, and vice-versa.

B. Evaluation of Table Selection

Efficiency. To study the effects of the top-k value on the computation time required by our heuristic algorithm for diversified table selection, Fig. 3 shows the computation time (in ms) relative to the result size. We observe that a solution is obtained quickly, in less than 140ms for k = 50, which can be seen as the maximum number of search results a user can

handle. In fact, we observe a linear trend of computation time despite the super quadratic time complexity of our algorithm. This highlights that our approach is efficient for real datasets.

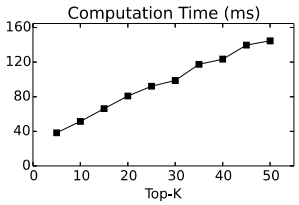

Fig. 3: Computation Time of Diversified Table Selection

Effectiveness (Top-K). Next, we study the effects of varying the top-k value on the diversity and relevance of the result. We use S-Recall and Normalized Relevance to measure diversity and relevance of the table selection returned by our approach, respectively. We randomly set the tunning parameter α (trade-off between schema similarity and data similarity) and w (trade-off between relevance and diversity) according to uniform distributions $\mathscr{U}(0,1)$ and $\mathscr{U}(1,2)$, respectively. The final result is computed as the average of 100 runs.

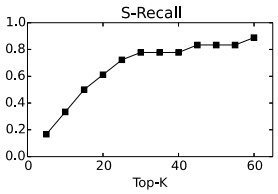

Fig. 4: Top-k vs. diversity

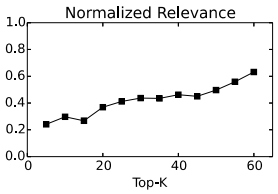

Fig. 5: Top-k vs. relevance

The results are depicted in Fig. 4 and 5. When increasing the top-k value, both S-Recall and Normalized Relevance increase as well. This is expected because of two reasons. First, when the size of the result set increases, more dissimilar tables are included as a result of the output objective of Algorithm 1, leading to higher S-Recall. Second, if we consider larger results, more relevant tables are chosen by our algorithm since the chance that they are dissimilar is higher, leading to higher Normalized Relevance. We conclude that our algorithm is stable and (except for a very few outliers) non-decreasing with the number of representative tables presented to user.

Comparison with k-medoids clustering. Next, we compare the effectiveness of our diversified table selection approach over another diversification algorithm – k-medoids clustering. As proposed in [13], k-medoids clustering can be used to diversify search results by generating k clusters of relevant results and picking a representative from each cluster. For the comparison, we measure relative improvement as $\Delta = (X - X_0)/X_0$, where X is the diversity (S-Recall) or relevance (Normalized Relevance) of the selection of our approach and X_0 is the respective value when using k-medoids clustering. We randomly set the tuning parameters as discussed above and take the average result over 100 runs.

The result for relative improvement in diversity (S-recall) is illustrated in Fig. 6 for an increasing top-k value. Interestingly but not surprisingly, the diversity is not continuously improved. At the beginning, diversity improvement increases sharply with the top-k value and reaches a maximum when $k \approx 30$. Then, the diversity improvement gradually falls down and reaches zero when $k = 60$. We conclude that when more relevant

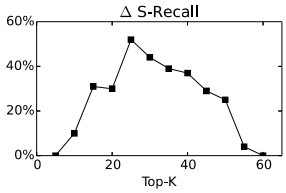

Fig. 6: Diversity improvement

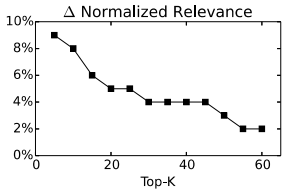

Fig. 7: Relevance improvement

results are included, dissimilar tables are shifted forward to the front of the selection, so that diversity of the top relevant results is higher.

The relative improvement in normalized relevance is depicted in Fig. 7. In general, the normalized relevance of our algorithm is always better than for the method based on k-medoids clustering. This result confirms that our algorithm can improve diversity without loss of relevance. Another key observation is that the relative improvement decreases when increasing the top-k value. This is expected since, when selecting more tables, the number of k-medoids clusters increases which increases chances to select relevant tables.

C. Evaluation of Table Summarization

Efficiency. To investigate the efficiency of our heuristic for structured table summarization, we measure computation time (in ms) when varying table sizes (tuples per table) from 50 to 100 and top-k values from 5 to 20. Three different linkage strategies are used for hierarchical clustering: *HierSingle* (single linkage), *HierComplete* (complete linkage), and *HierAverage* (average linkage).

TABLE III: Computation Time (ms) of Table Summarization

Table Size × K	HierSingle	HierComplete	HierAverage
50 × 5	60.269	65.488	49.684
50 × 10	65.452	60.937	51.79
50 × 15	69.948	85.551	64.648
50 × 20	86.897	121.625	71.372
100 × 5	115.487	194.91	104.547
100 × 10	187.299	478.561	179.917
100 × 15	194.504	658.073	254.477
100 × 20	271.09	846.499	227.732

Table III lists the computation times observed for the different configurations. Computation time increases if table size or top-k value increase. However, for all configurations, table summaries are created in less than one second. In most cases, *HierAverage* performs best.

Effectiveness (Information Loss). Next, we study the effectiveness of our approach in terms of information loss of the obtained table summaries. We vary table size from 50 to 400 and the top-k value from 5 to 45. The three aforementioned hierarchical clustering strategies are compared to the baseline method, which randomly selects a list of k tuples over the whole table. Note that a larger normalized information loss indicates that a technique suffers less from information loss.

For different table sizes (and a fixed top-k value of 10), Fig. 8 shows the normalized information loss of our approach (with three hierarchical clustering strategies) and the baseline (*Random*). We observe significant improvements for the medium ranges of table sizes. For the boundary cases of small (size < 50) and large (size > 350), there are no improvements over the baseline. This is explained by two reasons. For small

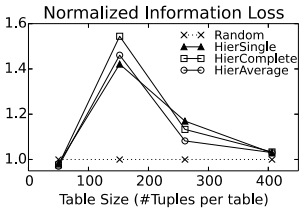

Fig. 8: Effect of Table Size

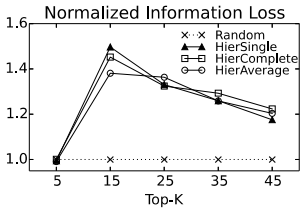

Fig. 9: Effect of Top K

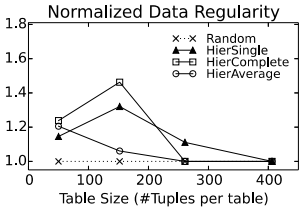

Fig. 10: Effect of Table Size

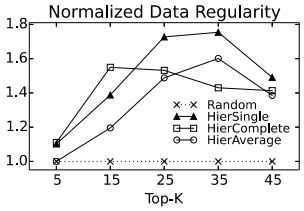

Fig. 11: Effect of Top K

tables, random selection of tuples works well since virtually no information loss can be implied by non-selected tuples. For large tables, a fixed summary size of $k = 10$ cannot capture sufficient information about non-selected tuples in the first place, so that structured table summarization has little effect.

For different top-k values (and a fixed table size of 150), the results in terms of the normalized information loss are shown in Fig. 9. We observe significant improvements for all cases except the smallest result size ($k = 5$). After reaching the maximum the ratios drop since more tuples are presented to user and, thus, less information is lost in comparison to random selection.

In sum, the results emphasize that our improves the quality of table summaries in terms of avoided information loss by up to 50% compared to the baseline method.

Effectiveness (Data Regularity). We further study the results in terms of data regularity in table summaries. Again, we vary table sizes and the top-k value as outlined above.

For different table sizes (and a fixed top-k value of 10), Fig. 10 illustrates the normalized data regularity. In general, our method achieves data regularity values higher than the baseline (up to a factor of 1.5). Another key finding is that the difference ratio decreases when table size increases. This is because of the fixed top-k value: to capture more information with 10 tuples, the selected tuples need to be more dissimilar to minimize information loss. As a consequence, data regularity is reduced.

The effect of different top-k values (under a fixed table size of 150) on normalized data regularity is illustrated in Fig. 11. Again, our summarization strategies perform better than the baseline (up to a factor of nearly 1.8). Data regularity increases when increasing the top-k value until about 35 tuples (except for HierComplete). This is expected since increasing the top-k value increases the number of clusters in our table summarization algorithm. Hence, selected tuples from these clusters are more similar, increasing data regularity. However, the difference ratio drops down for larger k. This is because, with fixed table size, increasing the top-k value further increases the chances of selecting similar tuples by random sampling, leading to higher data regularity.

Overall, the results underline that our approach is able to consider data regularities in table summaries.

D. User Study

To evaluate our techniques also from a user perspective, we conducted a user study using the CrowdFlower system. We designed two surveys in which a user is assigned to a certain evaluation task, called *HIT*. In each HIT, a number of questions on the result quality had to be answered. We allowed a maximum number of 10 users for each HIT and finally count all user responses to determine a trend in the result perception.

Below, we first discuss table selection before turning to table summarization. Finally, we reflect on limitations of the study.

Benefits of Table Selection. For this experiment, we designed HITs that ask users to compare two selections of tables for a given query. For each keyword, we retrieved a collection of relevant tables from Google Web Tables. A first list (*baseline*) is built by selecting the tables according to their relevance scores. A second list (Diversified Table Selection, DTS) contains the tables selected by our technique. Then, we built a HIT for each keyword (so there are 180 HITs in total) that comprised two questions. First, we asked users to rate the diversity of the DTS list against the baseline by five choices: from (1) *highly less diverse* to (5) *highly more diverse*. In the second question, we asked users which of the lists they prefer.

To make the study independent from table summarization techniques, no sample data of the tables was presented to the participants. We further considered only cases in which the number of identical tables in the two lists is less than 70% to prevent users from being confused with close to identical list.

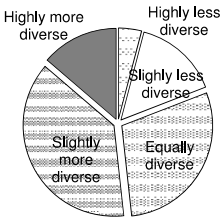

Fig. 12: Selection - Diversity

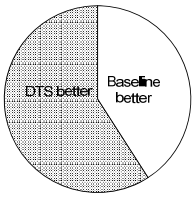

Fig. 13: Selection - Preference

For the first question on the diversity of the lists of selected tables, the percentages of user answers are shown in Fig. 12. We observe that 51.73% of the users answered that the selection derived with our technique is *highly* (13.72%) or *slightly* (38.01%) more diverse; whereas only few users considered it to be *slightly* (15.02%) or *highly* (4.08%) less diverse than the baseline. This confirms that our technique is sound and indeed increases diversity of table selection.

As illustrated in Fig. 13, 59.09% of the users prefer the DTS list over the baseline, which highlights the importance of diversification for satisfying the search intent of users and suggests that our selection technique helps to achieve it.

Benefits of Table Summarization. For this experiment, we designed HITs in which users were shown two versions of a summary of a Web Table. We randomly selected 10 of the tables retrieved for each keyword. Then, a first baseline summary shows tuples that were chosen randomly from the table. The

second version (Structured Table Summarization, STS) selected the tuples using our summarization technique. For each table, we build three separate HITs in which the number of tuples in both versions is fixed to 5, 20, and 50, respectively. The reason for separating different sizes of the tables into different HITs is to avoid dependence of user answers. This leads to 5400 (180 keywords $\times$ 10 tables $\times$ 3 sizes) HITs in total. For each HIT, we ask users which version is easier to read and which of the versions they prefer.

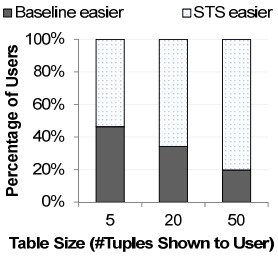

Fig. 14: Summary - Readability

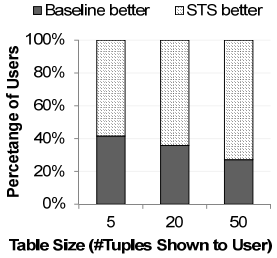

Fig. 15: Summary - Preference

Fig. 14 illustrates that for all summary sizes, users consider the STS version to be more readable than the baseline. This trend gets stronger with larger summaries. Over 80% of the users opt for the STS version for summaries of size 50.

The results shown in Fig. 15 indicate that the majority of users also prefers the STS version over the baseline. Again, the trend is stronger for larger summaries. For summaries with 50 tuples, more than 72% of the users prefer the STS version.

In sum, the results on readability and preference illustrate the effectiveness of our method for table summarization.

Limitations. Reflecting on limitations of our user study, we note that, despite the large number of HITs, the number of participants in the study is too small to allow for conclusions on the statistical significance of our observations. However, the user study consistently indicates that the proposed techniques lead to better results for table selection and table summarization than the baselines that represent the state-of-the-art.

VI. Related Work

Web Tables. Handling structured data in the Web has received a lot of attention in recent years. Early work includes Cafarella et al. [5], who report on massive extraction of structured data from the surface Web. Google Squared [9] and Fusion Table [18] are prominent applications that allow for table-based handling of Web data. Based on the model of Web Tables, the challenges of *annotating*, *searching*, *matching*, and *clustering* these tables have been addressed in the research community. *Annotations* provide semantics for a Web Table by associating the table columns and rows with types and relations [43]. *Search* aims at retrieval of Web Tables by matching table columns to keywords of a search query [29]. *Matching* enables data integration for Web sources by establishing the semantic correspondences between Web Table columns [15]. *Clustering* focuses on measuring the relatedness between Web Tables in order to group, join, or union them for simplifying the large-scale table corpus and improving table search [11]. Our work builds on the result of Web Table search and aims at supporting a user in browsing and exploring the retrieved tables. Our approach to table selection is further inspired by clustering of Web Tables, as the goodness function for a selection favors examples from groups of similar tables, but penalizes the selection of multiple tables from the same group.

Diversification Techniques. The importance of diversification in search results has been long acknowledged in information retrieval [14]. Diversification techniques are often categorized as being *threshold-based* [39], *function-based* [23], or *graph-based* [42]. *Threshold-based* algorithms define a threshold on one criterion (i.e. either relevance or diversity), and then select the results that both satisfy this threshold and optimize the other criterion. *Function-based* approaches combine both relevance and diversity in a unified function to extract a result set that maximizes this function. *Graph-based* approaches model the search results and their relations as a graph, and rank them according to the collective information inferred from the graph. In this work, we motivate the need for diversification for Web Table search and propose a function-based approach for diversified table selection.

Table Summarization. Table summarization is an active field of research and has been mainly studied for relational databases. While numerous techniques for table summarization have been proposed, the one by Chandola and Kumar [7] is closest to our work. Their approach uses clustering-based sampling to select representative tuples and leverages frequent sets to consider data regularities in relational tables. However, unlike our work, this approach is not stable if the size of the summary is increased, so that it lacks support for multi-resolution summarization. We overcome this limitation by combining hierarchical-based clustering and ordering techniques. In many cases, table summarization may also consider meta-data (e.g., table attribute taxonomies) [6]. In our setting of Web Tables, however, meta-data is either unavailable or costly to build up. Therefore, our technique for table summarization relies on the Web Table content itself. Table summarization may not only refer to a single table, but to a whole relational database, as proposed by Yang et al. [40]. They present a clustering-based method to select the tables that represent the database while maintaining the relations (e.g., foreign-keys) between the tables. Despite the fact that such relations do not exist between tables in the Web, the use case of Web Table search requires separate summarization of tables from diverse data sources. As such, instead of exploiting relations between tables, we target summaries that highlight relations between tuples of a table.

VII. Conclusions

This paper proposed techniques to support a user in browsing and exploring a result for Web Table search. Based on requirements identified for such support techniques, we formally defined two problems: *diversified table selection* and *structured table summarization*. The former relates to the selection of tables based on their relevance while ensuring diversity within the result. The latter refers to the selection of tuples to provide a concise yet meaningful summary of a table. We showed that both problems are intractable. Hence, we developed heuristic algorithms for their approximation that come with performance guarantees, such as near-optimality, stability and fairness. Our evaluation showed that our techniques outperform respective baseline methods significantly, up to 50% in diversity for table selection and up to 50% reduction of information loss induced

by table summarization. Our user study showed that participants indeed saw a clear benefit in using the proposed techniques.

In future work, we aim at adapting our approach to other Web data sources that are not tabular. Also, we intend to devise applications on top of our techniques and support data exploration beyond keyword search.

Acknowledgment. The research has received funding from the EU-FP7 EINS project (grant number 288021) and the ScienceWise project.

REFERENCES

[1] D. Aumueller et al. "Schema and ontology matching with COMA++". In: *SIGMOD*. 2005, pp. 906–908.
[2] P. Bernstein et al. "Generic Schema Matching, Ten Years Later". In: *PVLDB*. 2011, pp. 695–701.
[3] N. Bruno et al. "Top-k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation". In: *TODS* (2002), pp. 153–187.
[4] M. J. Cafarella et al. "Data integration for the relational web". In: *PVLDB*. 2009, pp. 1090–1101.
[5] M. J. Cafarella et al. "Webtables: exploring the power of tables on the web". In: *PVLDB*. 2008, pp. 538–549.
[6] K. S. Candan et al. "Reducing metadata complexity for faster table summarization". In: *EDBT*. 2010, pp. 240–251.
[7] V. Chandola et al. "Summarization - compressing data into an informative representation". In: *ICDM*. 2005, 8 pp.–.
[8] H. Chen et al. "Less is more: probabilistic models for retrieving fewer relevant documents". In: *SIGIR*. 2006, pp. 429–436.
[9] D Crow. "Google squared: Web scale, open domain information extraction and presentation". In: *ECIR*. 2010.
[10] N. Dalvi et al. "An analysis of structured data on the web". In: *PVLDB*. 2012, pp. 680–691.
[11] A. Das Sarma et al. "Finding Related Tables". In: *SIGMOD*. 2012, pp. 817–828.
[12] D. Defays. "An efficient algorithm for a complete link method". In: *CJ* (1977), pp. 364–366.
[13] M. Drosou et al. "DisC Diversity: Result Diversification Based on Dissimilarity and Coverage". In: *PVLDB*. 2012, pp. 13–24.
[14] M. Drosou et al. "Search Result Diversification". In: *SIGMOD*. 2010, pp. 41–47.
[15] J. Fan et al. "A hybrid machine-crowdsourcing system for matching web tables". In: *ICDE*. 2014, pp. 976–987.
[16] U. Feige et al. "The dense k-subgraph problem". In: *Algorithmica* (2001), pp. 410–421.
[17] M. R. Garey et al. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.
[18] H. Gonzalez et al. "Google fusion tables: web-centered data management and collaboration". In: *SIGMOD*. 2010, pp. 1061–1066.
[19] J. He et al. "Gender: A generic diversified ranking algorithm". In: *NIPS*. 2012, pp. 1142–1150.
[20] A. K. Jain et al. "Data clustering: a review". In: *CSUR* (1999), pp. 264–323.
[21] L. Kaufman et al. *Clustering by means of medoids*. North-Holland, 1987.
[22] D. Knuth. *The Art of Computer Programming. Sorting and Searching*. Vol. 3. Addison-Wesley, Reading, 1973.
[23] O. Küçüktunç et al. "Diversified recommendation on graphs: pitfalls, measures, and algorithms". In: *WWW*. 2013, pp. 715–726.
[24] X. Li et al. "Truth finding on the deep web: is the problem solved?" In: *VLDB*. 2012, pp. 97–108.
[25] C. D. Manning et al. *Introduction to information retrieval*. Vol. 1. Cambridge university press, 2008.
[26] A. Marie et al. "On the Stable Marriage of Maximum Weight Royal Couples". In: *IIWeb*. 2007, pp. 1–6.
[27] F. Murtagh. "A survey of recent advances in hierarchical clustering algorithms". In: *CJ* (1983), pp. 354–359.
[28] G. Nemhauser et al. "An analysis of approximations for maximizing submodular set functions–I". In: *MP* (1978), pp. 265–294.
[29] R. Pimplikar et al. "Answering table queries on the web using column keywords". In: *PVLDB*. 2012, pp. 908–919.
[30] F. Radlinski et al. "Redundancy, diversity and interdependent document relevance". In: *SIGIR*. 2009, pp. 46–52.
[31] S. P. Smith et al. "Stability of a hierarchical clustering". In: *PR* (1980), pp. 177–187.
[32] F. M. Suchanek et al. "Yago: a core of semantic knowledge". In: *WWW*. 2007, pp. 697–706.
[33] G. J. Szekely et al. "Hierarchical clustering via joint between-within distances: Extending Ward's minimum variance method". In: *JOC* (2005), pp. 151–183.
[34] P.-N. Tan et al. *Introduction to data mining*. Pearson Education India, 2007.
[35] *http://lsirwww.epfl.ch/web_table/*.
[36] *http://www.factual.com/*.
[37] *http://www.socrata.com/*.
[38] P. Venetis et al. "Recovering Semantics of Tables on the Web". In: *PVLDB*. 2011, pp. 528–538.
[39] M. R. Vieira et al. "On query result diversification". In: *ICDE*. 2011, pp. 1163–1174.
[40] X. Yang et al. "Summarizing relational databases". In: *PVLDB*. 2009, pp. 634–645.
[41] C. X. Zhai et al. "Beyond independent relevance: methods and evaluation metrics for subtopic retrieval". In: *SIGIR*. 2003, pp. 10–17.
[42] B. Zhang et al. "Improving web search results using affinity graph". In: *SIGIR*. 2005, pp. 504–511.
[43] M. Zhang et al. "InfoGather+: Semantic Matching and Annotation of Numeric and Time-varying Attributes in Web Tables". In: *SIGMOD*. 2013, pp. 145–156.

APPENDIX: PROOFS OF GOODNESS PROPERTIES

Monotonicity follows by the following transformation ($w \geq 2$):

$$
\begin{aligned}
&g_{\mathcal{U},r}(U^*.U') - g_{\mathcal{U},r}(U^*) = w \sum_{x \in U'} q(x) r(x) \\
&- \Big(\sum_{x \in U', x' \in U^*} r(x) M(x,x') r(x') + \sum_{x \in U^*, x' \in U'} r(x) M(x,x') r(x') \\
&+ \sum_{x,x' \in U'} r(x) M(x,x') r(x') \Big) = w \sum_{x \in U'} r(x) \sum_{x' \in \mathcal{U}} M(x,x') r(x') \\
&- \Big(2 \sum_{x \in U^*, x' \in U'} r(x) M(x,x') r(x') + \sum_{x,x' \in U'} r(x) M(x,x') r(x') \Big) \\
&\geq 2 \sum_{x \in U'} r(x) \sum_{x' \in \mathcal{U}} M(x,x') r(x') - \Big(2 \sum_{x \in U^*, x' \in U'} r(x) M(x,x') r(x') \\
&+ \sum_{x,x' \in U'} r(x) M(x,x') r(x') \Big) = 2 \sum_{x \in U'} \Big(\sum_{x' \in \mathcal{U}} M(x,x') r(x') \\
&- \sum_{x' \in U^*.U'} M(x,x') r(x') \Big) = 2 \sum_{x \in U'} \sum_{x' \notin U^*.U'} M(x,x') r(x') \geq 0
\end{aligned}
$$

Submodularity follows by the following transformation:

$$
\begin{aligned}
&g_{\mathcal{U},r}(U^*.T) + g_{\mathcal{U},r}(U^*.T') \geq g_{\mathcal{U},r}(U^*.T.T') + g_{\mathcal{U},r}(U^*) \\
&\Leftrightarrow g_{\mathcal{U},r}(U^*.T') - g_{\mathcal{U},r}(U^*) \geq g_{\mathcal{U},r}(U^*.T.T') - g_{\mathcal{U},r}(U^*.T) \\
&\Leftrightarrow w q(T') r(T') - 2 r(T') \sum_{x \in U^*} r(x) M(x,T') \geq w q(T') r(T') - \\
&\qquad 2 r(T') \sum_{x \in U^*.T} r(x) M(x,T') \Leftrightarrow 2 r(T) r(T') M(T,T') \geq 0
\end{aligned}
$$

Strength of Relevance follows by this transformation ($w \geq 2$):

$$
\begin{aligned}
&g_{\mathcal{U},r'}(U^*.T) - g_{\mathcal{U},r}(U^*.T) = g_{\mathcal{U},r'}(U^*) + w q(T) r'(T) \\
&- 2 r'(T) \sum_{x \in U^*} M(x,T) r'(x) - g_{\mathcal{U},r}(U^*) - w q(T) r(T) + 2 r(T) \sum_{x \in U^*} M(x,T) r(x) \\
&= w q(T) [r'(T) - r(T)] - 2 \sum_{x \in U^*} M(x,T) r(x) [r'(T) - r(T)] \\
&= w \sum_{x \in \mathcal{U}} M(x,T) r(x) [r'(T) - r(T)] - 2 \sum_{x \in U^*} M(x,T) r(x) [r'(T) - r(T)] \\
&= (w-2) \sum_{x \in U^*} M(x,u) r(x) (r'(T) - r(T)) \\
&\quad + w \sum_{x \in \mathcal{U} \setminus U^*} M(x,u) r(x) [r'(T) - r(T)] \geq 0
\end{aligned}
$$