Table Expansion: Populating Relational Web Tables Based on Examples

by

Zharkyn Kassenov

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The Web contains an enormous amount of structured data in the form of web tables, and there is a great value in retrieving this data and harnessing it for decision making and gain more insights. Finding the right data on the Web and integrating it with the existing data within an organization can be a very time-consuming task. To address this problem, this thesis studies the problem of table expansion where given a query table and a corpus of tables, the goal is to expand the query table with additional rows that are likely to belong to the same table. Given the challenges of querying web tables, our approach relies only on instances in the given query table and not on the schema which may not be present or known for tables in the corpus. It uses projections to split tables into entity-attribute binary relations (sets of key-value pairs) and then leverages co-occurrence statistics to retrieve candidate key-value pairs that are then combined into candidate rows. Our experiments show that constraints required by alternative approaches, such as relying on column labels and contextual information of a web page containing the table, can negatively affect the results and, in some cases, makes them not suitable for the task.

To my Family

For supporting me in everything.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A web table is an important structural unit that is used to represent information in a logically structured way, allowing users to grasp the information with ease compared to plain text.

The World Wide Web contains an enormous amount of structured data in the form of web tables. Most of these tables are used for layout purposes, however, a fraction of them, referred to as relational tables [9, 10], entity-attribute tables [56] and 2-dimensional tables [54], contain structured data describing a set of entities.

Compared to unstructured text of a web page, web tables have several favorable features. The structure of a web table informs its semantics by reflecting logical relations between the data [24], which makes them less ambiguous than free text and easier to process algorithmically [34]. Another advantage is that they represent multiple similar entities with the same attributes, which helps reduce extraction efforts. Because of these features, web tables have gained an increasing attention by the research community.

In the last decade a significant amount of work has been developed around exploiting web tables in many different settings such as concept expansion [48], keyword-based searches [2, 38, 58] or table-as-a-query searches [16, 44], knowledge extraction [37], table extension [9, 31, 32, 53, 57], filling missing cell values [1], and entity linking [4].

In this thesis, we focus on the problem of table expansion, where the task is to populate a seed table by discovering other rows that may semantically

belong to the table, based on a corpus of web tables. The problem of table expansion is similar to the problem of table extension in the sense that new elements are added to the seed table, however, in table expansion we add rows to the table, while in table extension we add columns [52].

A seed table is defined by a set of entities described by attributes, for example a set of countries with attributes *capital city* and *language*. We refer to a seed table as a query table and the rows (entities with their attributes) as instances. The result of the query should then consist of other countries with their *capital city* and *language* that have been automatically extracted from the web table corpus.

The problem of table expansion is also related to Query-by-Example (QBE), a language supported by multiple database management systems, such as Microsoft Access and Oracle. QBE is a graphical query language, where a user uses visual tables to specify commands, example elements and conditions. The QBE input is then converted into statements expressed in a database manipulation language, such as Structured Query Language (SQL). The crucial difference of QBE from the problem of table expansion is that it requires table schema information and web tables do not contain such information.

1.1 Motivation

Data-driven decision making has become an essential component of a growing number of organizations [7]. Many such organizations aim to generate value from all available data, not only from internal sources, but also from a growing number of external sources. If traditionally the list of external sources mainly included sensor measurements and application logs, today it also includes publicly available web data.

The Web is considered to be a virtual gold mine of data for business [27] and the web tables are the big part of it. The Web Data Commons project [52], for example, extracts 233 million relational web tables that were contained in the overall set of 11 billion HTML tables found in the repository of the

Common Crawl data [1]; Yakout *et al.* [53] reports 573 million relational web tables extracted from a crawl of the Microsoft Bing search engine; Gatterbauer *et al.* [24] identifies "visual" web tables that only appear as 2-dimensional grid when a web page is rendered and the number of such tables is unestimated.

Being a rich and growing source of knowledge, web tables are used in various applications such as data search [2, 38], knowledge-base construction [18, 46] and table extension [31, 53, 57]. However, one of the overlooked developments is table expansion.

Imagine a company in the game industry that wants to expand its market. The company may want to learn about possible locations to host their tournaments. To achieve that they need to get and analyze various publicly available information about held tournaments around the world, such as locations, language, number of teams that participated, and team sizes.

Information about tournaments is posted in numerous web resources in the form of web tables. To collect this information analysts of the company have to perform three tasks: locate the web resources, retrieve web tables about tournaments, and then convert into a common table schema for further analysis. Those are laborious tasks that take many person-hours since they go beyond capabilities of web search engines.

What if the company needs to perform the analysis every year for reassessment purposes? In a year, hundreds of new web tables in the domain might have been added and some of already collected tables might have been updated. The analysts have to start from scratch and may likely spend even more time to process the grown dataset of the web tables.

Now imagine that the company had a table expansion system which takes a query table as an input and expands it with additional rows "similar" to the rows in the query table. In this case, the analysts only would have to find several example entities with attributes they are interested in and create a query table. The system would then return an expanded table ready for further analysis.

Such system would be helpful not only for data-driven businesses but also

[1] `http://commoncrawl.org/`

for data enthusiasts [36] like journalists who are looking to enrich their stories and illustrations with relevant web data.

Developing such a system supporting table expansion queries, can provide more efficient retrieval of relevant data and help businesses to use the vast amount of relational data "locked" in the web, is the goal of this thesis.

1.2 Thesis Contribution

The contributions of this thesis can be summarized as follows:

- We formalize the problem of table expansion. Unlike the problem of set expansion, the questions around a table expansion are not well studied in the literature.
- We propose a relatedness score to select an attribute value for a candidate key that relies only on tuples of a query table. Related work addresses attribute value prediction with a scoring that leverages schema level information, sometimes along with instance level information.
- We conduct experiments to evaluate our proposed system and compare it with an alternative solution. As ground truth we use both curated, high-quality web tables from Wikipedia, as well as tables that are randomly selected from a table corpus extracted from the Common Crawl.

1.3 Thesis Outline

The rest of this thesis is organized as follows.

Chapter 2 introduces relational web tables and their characteristics. First, we study the notion of a web table, look at different types of web tables, and define *relational* web tables. Then we describe different characteristics associated with relational web tables and make some simplifying assumptions.

Within Chapter 3, we review the literature closely related to ours. We give an overview of the usage of web tables as data sources, look at the topic of table relatedness, detail approaches to web table extension, and discuss entity set expansion methods.

Chapter 4 presents our approach to the problem of table expansion. First, we present some preliminaries on schema-matching and a formal data model. Next we show a naive approach that tries to apply an entity set expansion method to the problem, before describing the proposed approach that overcomes the limitations of the naive approach.

Having introduced the proposed approach, Chapter 5 presents its experimental evaluation. During the experimentation phase, the goal was to evaluate the proposed system by comparing its performance to alternative solutions.

Chapter 6 summarizes this thesis and discusses directions for future work.

Chapter 2

Background on Web Tables

Tables provide a two-dimensional structure which enables a compact visualization of data [55]. They are frequently found in both, printed documents and digital resources, such as web pages. Tables also represent an important concept in relational databases and spreadsheets. In this thesis, we focus on tables that are found on web pages known as *web tables*.

In this chapter, we introduce *relational* web tables and their characteristics. First, we study the notion of a web table, types of web tables and define relational web tables. Then we describe different characteristics associated with relational web tables and make some simplifying assumptions.

2.1 Web tables and their place in the Web

A web table is an important structural unit of a web page that represents information in a logically structured format, allowing users to grasp the information with ease, compared to plain text. Lautert *et al.* [28] defined web tables as follows:

Definition 2.1.1. A web table is a two-dimensional tabular structure found on a web page that is composed of an ordered set of x rows and y columns.

Not all web tables contain a valuable content. A web table cell can contain any number of other HTML elements, like images or other, nested web tables. Additionally, a cell can span over several columns or rows, similar to the merged cells functionality of a spreadsheet application. This flexibility has

made the table tag popular for positioning other elements on a web page. Tables used only for positioning other elements on a web page are called "layout" tables.

The web tables that actually represent tabular data are sometimes referred to as "genuine" web tables [41]. The value of a genuine web table cell is an atomic value and does not contain complex structures such as other web tables. Only a small amount of web tables on the Web are genuine web tables. According to [10], about 99% of web tables define layout of web-pages. The remaining 1% are genuine tables where rows and columns are syntactically and semantically coherent [41]. We disregard layout web tables in this work.

Contextual Information. A web page containing a web table element may include various contextual information related to that table such as web page title, page URL, table caption and surrounding paragraphs.

Figure 2.1 illustrates the contextual information of a web table on a Wikipedia web page for 2008 Summer Olympics medal table. We have highlighted some of them with reference numbers from below:

1. Page title - a required HTML tag that has a document title.
2. Page URL - hyperlink to the web page that has the table.
3. Section header - a header of a section from which the table was extracted.
4. Keywords - unique words extracted from text segments before table.
5. Table caption - value of an optional <caption> tag of the table.

Figure 2.1: Web table contextual information.

2.2 Types of Web Tables

Genuine web tables are used to represent multiple forms of tabular data and are classified into the following groups [1]: relational tables, entity tables and matrix tables.

Relational tables contain a set of similar entities described by one or more attributes [8]. Here the key column contains names of the entities. For example, a table with Olympic Games results will contain a list of countries that participated in the games and attributes such as *gold*, *silver* and *bronze* medal counts. Another example is a table with a list of clubs as shown in

Table 2.1 where the key column is the *club name* and each club is described by its *home city*, *home stadium*, *capacity* of the home stadium, and the *head* of the club. Relational tables are also referred to as entity-attribute tables [53], 2-dimensional tables [54], and horizontal tables [28].

	Club	**City**	**Stadium**	**Capacity**	**Head**
1	Buffalo Bills	Orchard Park, New York	New Era Field	71608	Sean McDermott
2	Miami Dolphins	Miami Gardens, Florida	Hard Rock Stadium	64767	Vacant
3	New England Patriots	Foxborough, Massachusetts	Gillette Stadium	65878	Bill Belichick
4	New York Jets	East Rutherford, New Jersey	MetLife Stadium	82500	Vacant
5	Baltimore Ravens	Baltimore, Maryland	M&T Bank Stadium	71008	John Harbaugh
6	Cincinnati Bengals	Cincinnati, Ohio	Paul Brown Stadium	65515	Vacant
7	Cleveland Browns	Cleveland, Ohio	FirstEnergy Stadium	67895	Gregg Williams

Table 2.1: Example of a relational table.

Entity tables describe only one entity with one or more attributes. The name of the entity is typically not in the table itself but may be found in the web page that contains the table. An example would be a football player web page that has an entity table describing the place of birth and the club information of the player, as shown in Table 2.2.

Personal	
Full name	Cristiano Ronaldo dos Santos Aveiro
Date of birth	5 February 1985 (age 33)
Place of birth	Funchal, Madeira, Portugal
Height	1.85 m (6 ft 1 in)
Playing position	Forward
Club information	
Current team	Juventus
Number	7

Table 2.2: Example of an entity table.

Matrix tables are often used to describe statistical evaluation results and the relations between entities. An example of this type is shown in Table 2.3 relating diet types to health data in a contingency table.

Diet	**Cancers**	**Fatal Heart Disease**	**Non-Fatal Heart Disease**	**Healthy**	**Total**
AHA	15 (11.02)	24 (19.03)	25 (16.53)	239 (256.42)	303
Mediterranean	7 (10.98)	14 (18.97)	8 (16.47)	273 (255.58)	302
Total	22	38	33	512	605

Table 2.3: Example of a matrix table.

In this work we target relational web tables as a data source and refer to them as web tables. Relational web tables account for about 38% of all genuine

web tables [52] (see Table 2.4), and they describe sets of entities rather than singe entities.

Type	**Number of tables**	**% of genuine tables**
Relational	90,266,223	38.37
Entity	139,687,207	59.94
Matrix	3,086,430	1.32
Sum	233,039,860	100

Table 2.4: Table types frequencies in genuine web tables.

2.3 Relational Web Tables

The term "relation", in the data modeling sense, was first introduced by E.F. Codd [13], as the central element of the relational database model, to describe the logical structure of data. Conceptually, the relational model consists of relations (tables) of tuples (objects). Each tuple contains one or more fields, each taking atomic values.

Following the Relational model definition, we can formally define a web table to be relational if the rows provide information about a set of entities and the columns represent attributes that describe them. A relational web table is an entity set, where the attributes of the table represent the entity type.

We can look at a relational web table t with n rows and m columns as a container that holds n entities described by m attributes. Each row i in the table represents an entity with a name k_i and a value $v_{i,j}$ for each attribute a_j.

Figure 2.2 illustrates a relational web table, along with the terminology used in this thesis. The attribute that uniquely describe each entity is called the key (subject) column. All other columns are called attribute columns and contain data values. The key column is sometimes referred to as key attribute and attribute columns as data attributes.

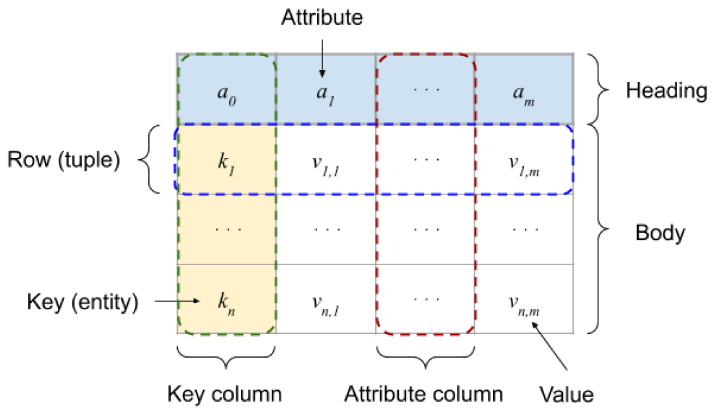

Figure 2.2: Overview of a relational web table.

In this work, we refer to relational web tables as web tables.

2.4 Characteristics of Relational Web Tables

Below we describe the main characteristics of relational tables on the Web. While most of those characteristics complicate the table expansion, some characteristics can also be beneficial.

Heterogeneous nature. It is a challenging task to use web tables as a source of information due to the way the content is created on the Web. The World Wide Web is decentralized and designed for humans to read and interact with. Also, it lacks predefined rules on how the information should be presented and only provides HTML tags to structure information, thus the content can be created in an arbitrary format. The language used to express the content of web tables is inherently versatile and ambiguous. Web data is often "dirty" because of the large number of contributors and their diversified knowledge of the standard [26].

Lack of formal table schema. Web tables visualize tabular data for human readers and are not designed for automatic processing, therefore they do not have strict constraints on their structure. In contrast to database tables, tables on the Web do not provide a formal schema. Web tables do not contain any information on what constitutes the key of a table, whether it is simple, i.e., consisting of a single column, or composite, i.e., consisting

of multiple columns. The headers might or might not be present and the number of columns varies. Columns can contain any type of information, such as number, date or text, and often several types at once, and multiple cells can be merged into one. The lack of common schemas also leads to other kinds of heterogeneities, such as different naming conventions for entities and attributes including abbreviations and different choices in the visual design of the table.

Noise. The content of the tables, in some cases, can have meaning only in the context of the website or even web page, e.g. site map and visit statistics. We consider such tables as "noise". Moreover, widely-spread content generators create millions of pages every day, and many of them containing web tables that sometimes add noise to the web data.

Lack of quality control. The absence of quality control mechanisms means that the data is often incomplete and no guarantees for the correctness can be made. This can lead to incorrect entries, missing labels, and duplicate tables, among other issues. However, many web resources provide high-quality content, e.g. Wikipedia, by having rules for content creation. The level of the noise in such resources is low and they often become the main source of data for table-as-a-query searches [16, 23], entity linking [4], and data expansion [32].

Vast quantities and redundancy. Another challenge is posed by the large and ever increasing number of web tables on the Web. Taking as many web tables as possible into account in search allows having a broad topical coverage. On the other hand, the missing quality control leads to an increased level of noise and a low quality data. Additionally, web data is often copied and slightly modified. As a result many web tables will be similar in the structure and content [15, 33].

Small tables. The majority of tables contain only a very small number of rows and columns. Web tables contain on average 14 entities, with a median of 6 [52]. The number of attributes on average is 5, with a median of 4 [52]. As a result, individual tables provide on average only little content that can be utilized to infer the semantics or match the tables to user queries or other

tables.

Less-descriptive attribute labels. The descriptiveness of attribute labels refers to how well the meaning behind an attribute (and, eventually, the complete table) can be inferred from the label text. Unfortunately, about 20% of tables on the Web miss headers entirely [42], and about 6.5% of all headers are missing some of labels [52], so-called "implicit" labels [21]. In many cases (around 80% of all headers [52]), the labels are very generic, so-called "non-informative" labels, such as *name* or *value*. These types of labels provide little to no information about the meaning of an attribute. Figure 2.3 shows that such labels are among the most frequent on the Web.

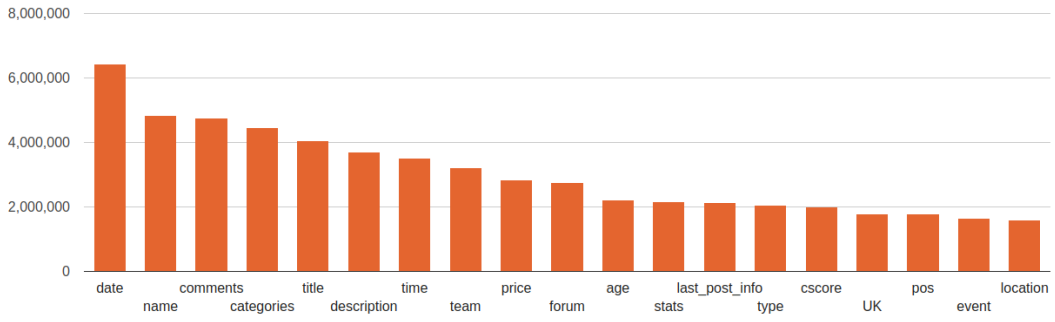

Figure 2.3: Popular Column Headers.

2.5 Simplifying Assumptions

The problem of table expansion is very challenging due to the described characteristics of web tables. To address some of the complexity and uncertainty, we make the following simplifying assumptions:

- **Consider only relational web tables.** To reduce the structural ambiguity, we only consider web tables that are relational.
- **The key column is known and simple.** We assume that key column of each table is known, simple, and is the first attribute.

These assumptions has been made in many previous works on web tables [10, 20, 32, 53]) and are satisfied by WDC Web Table Corpus [1] that we use

[1] `http://webdatacommons.org/webtables/`

for evaluation purposes.

Chapter 3

Related Work

A significant amount of research is done on issues surrounding web tables including information extraction [37], concept expansion [48], table search [2, 16, 38, 58] and table extension [9, 31, 32, 53, 57].

The research closely related to ours can be grouped as follows:

- *Table search* is the task of returning a ranked list of web tables in response to a query.
- *Table expansion* populates a given table with additional rows based on a corpus of web tables.
- *Table extension* aims at extending a given table with additional columns based on a corpus of web tables.
- *Entity set expansion* is the task of expanding a set of entities with additional entities.

In this chapter, we will first give an overview of early work on web tables and its usage as data source, then we will review the related work on the topics of web table search, web table expansion, web table extension, and entity set expansion methods, which are closely related to our work.

3.1 Early Work

Some of the early work [11, 49] on web tables is focused on extraction and structural analysis of web tables. These early approaches are focused on the

recognition and analysis of different types of tables. Cafarella *et al.* [10] is the first to extract a large scale corpus of tables from the Web and present an application based on that corpus. One of their findings is that Google's Web Crawl contained around 14.1 billion HTML tables and about 154 million of those tables contained useful relational data. They also introduce a method for differentiating relational data tables from layout tables and present several use cases for the corpus, such as table search, attribute synonym discovery, and schema auto-completion. In the following years, more research related to the use of web table corpora were published, including how to help answer specific query types. For instance, FACTO [54] uses a corpus of web tables to answer single fact keyword queries, whereas WWT [42] returns a multi-column table in response to a query consisting of keywords. Other applications of web tables include semantic understanding of web tables [34], identifying table relatedness [16], and entity expansion [14].

3.2 Table Search

Search for web tables is one of the first steps in information retrieval tasks [3, 32, 56] and in contrast to table expansion, table search does not aim at further integration of data from the relevant tables with a query table. The problem of finding related tables has been studied for web tables using queries that are keywords, attribute labels, table schemas, and table instances.

Search by keywords

Cafarella *et al.* [10] propose an approach for the web table search, called WebTables, that performs a keyword search on top of a web search engine. Particularly, given a keyword query, they use the top-ranked results returned by the web search engine, and extract the top-k tables from those result pages. Keyword-based search is also used in table extension methods. To find related tables to keyword queries, Octopus [9], for example, introduces a search operator that takes a search-style keyword query and returns a set of web tables, which are ranked based on relevance and clustered based on their similarities.

Moreover, query keywords are used to discover web tables with attributes relevant to the query. In this case, the keywords are referred to as *query attributes*. Pimplikar and Sarawagi [42] propose a table search system, called WWT, which takes a keyword query as description of table attributes and returns a multi-column table satisfying those query attributes. Table extension methods also use query attributes to find related tables. For instance, Infogather [53, 56] searches for web tables with an attribute matching a given query attribute. Later on, to speed up search for web tables related to a query attribute, table extension methods started to leverage an inverted index. REA [20] and MSJ [32], for instance, build an inverted index over column headers of tables from a corpus.

Search by table

A query for searching tables can also be a table, referred to as a *query table*. The relatedness between tables is discovered by schema matching, which is the task of identifying semantic correspondences between attributes of two schemas. The task is sometimes also referred to as *schema alignment*. Schema matching approaches can be classified into two main categories - schema-based and instance-based [43]. *Schema-based methods* primarily use labels, column data types, and other schema information to find similar attributes. In general, a schema matching method finds all the possible matches, then for each candidate table, a score function is usually defined to estimate the degree of similarity. *Instance-based methods* exploit properties of the underlying data, possibly in combination with attribute labels, to derive matches. Instances could potentially provide a more insightful view on the semantics of the data, especially in cases where the schema information is not available or is limited.

For detecting tables related in the Wikipedia table corpus, Das Sarma *et al.* [16] propose an instance-based schema matching method which discovers tables that can be either joined or unioned with the query table. The method considers column headers, entities and attribute values to determine relatedness between the tables. Another approach for establishing relations between Wikipedia tables is proposed in TableNet [23]. To identify relatedness between

two tables, in addition to column headers and attribute values, they also consider the category of the article in Wikipedia containing the table and the text of that article.

In DUMAS [5] the schema matching problem is approached by an instance-based matching algorithm. The authors look for matching rows between two tables to perform schema matching.

Ling *et al.* [35] define table stitching where identical schemas within a site are merged into a union table. The authors propose a schmea-based matching method, that relies on table headers and column headers, to determine if two tables can be unioned. Lehmberg *et al.* [29] proposes an approach that is built upon this work. First, they create union tables using the approach from [35] and afterwards, these tables are matched between each other using a hybrid matcher to determine if the tables can be stitched. The hybrid matcher combines schema-based and instance-based matching techniques, and it uses column labels, attribute values and table rows to determine relatedness between tables.

3.3 Table Expansion

To the best of our knowledge, the problem of table expansion has not been addressed in the literature. A similar definition to the problem of table expansion is given in a survey [59] that describes a problem of *row extension* which aims at extending a query table with more rows or row elements. Although the definition of the problem is similar to the definition of the problem of table expansion, the type of work they describe aims at unioning related tables [16], expanding a subject column of a query table [25, 48, 57] or at prediction of attribute values for a row of a query table that does not have the attribute values [20, 32, 53]. By combining these basic operations, one may build a table expansion operator. We compare our work to some of these baselines in Section 5.

A method for detecting tables in a web table corpus that can be either joined or unioned is proposed in [16]. The authors define two types of related-

ness between two tables, entity complement and schema complement. When two tables are entity complement, they can be unioned, and when they are schema complement, they can be joined. The work is relevant to the problem of table expansion as discovered tables can potentially be unioned (rows of one table can be added to another table), however, there are several crucial differences. First, the method discovers relatedness between "entire" tables in the corpus and not with a query table containing small number of instances. Second, it establishes entity complement relatedness between two tables only if they have similar schemas. Finally, the work only discovers relatedness between two tables and does not aggregate rows from several related tables into an expanded table.

In EntiTables, Zhang and Balog [57] propose smart assistance that helps to extend a query table with additional entity names and column labels. To discover candidate tables, the authors introduce a similarity function between tables that considers overlap between entity names, column labels, and string similarity of the table captions. To rank entities retrieved from candidate tables, the authors propose a multi-conditional probabilistic method that combines entity similarity, column labels likelihood, and the caption likelihood. The work is similar to table expansion as it aims at expanding of the subject column of the query table with additional entities, however, they do not populate attribute values. Populating entities in a subject column is also similar to the problem of entity set expansion [25, 48], which expands a given seed set of entities. We discuss entity set expansion methods in Section 3.5.

Populating attribute values for an entity, given a query table and a table corpus, is addressed in some table extension methods, and referred to as *entity augmentation* [20, 53]. Lehmberg *et al.* [32] propose an approach that populates attribute values for a given query entity set and a query attribute. It searches for tables describing query entities and merges columns with labels matching the query attribute. The resulting column is joined with the query entity set using left outer join operation. In InfoGather [53], the authors present two operations that populate attribute values for an entity name given a query attribute or a query table, namely augmentation-by-attribute

and augmentation-by-example. Instead of scoring candidate tables based on their schema matching with the query table, they propose a holistic matching framework based on topic sensitive PageRank (TSP). They build a weighted graph between all web tables in the corpus based on pairwise table similarity, which uses schema-based and instance-based schema matching, and compute personalized page rank. To predict values for a given entity name, they employ an augmentation framework that aggregates predictions from multiple matched tables based on the tables relatedness scores. The scores are calculated based on a topically related set of tables, which, in case of augmentation-by-attribute, is a set of tables in the corpus that overlap with the query table keys and a column header that matches a query attribute, and in case of augmentation-by-example, is a set of tables in the corpus that overlap with the query table keys as well as the query table tuples. The main limitation of the method is its computationally expensive prepossessing step (build graph based on pairwise table similarity and compute scores) [59]. As an alternative, Eberius *et al.* [20] present a top-k entity augmentation algorithm called REA. Instead of returning a single augmentation for a given query attribute and a query entity set, the approach returns multiple alternative augmentations. They introduce a similarity function between tables, in addition to a scoring function, which allows the construction of consistent augmentations that are retrieved from a minimal number of sources and have a maximal score. Entity augmentation is closely related to the problem table expansion as it predicts attribute values for a given entity name based on a query table, however, it does not expand the query table with additional rows.

3.4 Table Extension

A closely related type of work to ours is table extension, which is aimed at extending an input table with additional columns based on a corpus of tables. This line of work aims at collecting tables that contain the same entities but cover complementary attributes of the entities, and integrate collected tables by joining them on the same entities. For instance, a table describing countries

can be extended to have columns containing the population, language, and the capital of each country.

Table extension methods are usually implemented based on a corpus of tables extracted from the Web. Such corpus typically contains tables in the order of hundreds of millions [9, 20, 32, 56]. Those tables are indexed using an inverted index, like Apache Lucene [1], to enable search abilities based on subject (entity) names, attribute titles, and auxiliary meta-data, such as keywords surrounding the table and page title. A query, in this case, is a combination of two elements - a set of entity names (query entities), for example, a list of country names, and an attribute name (query attribute), that needs to be discovered, for instance, "population". To process a given query, all candidate tables with at least one query entity and an attribute that matches the query attribute name may be retrieved. Then, for each candidate table, various schema and instance matching methods, such as a string edit distance and a synonym databases, may be employed to calculate a mapping between query entities and entities in a candidate table, as well as between the query attribute and attributes of the candidate table. This process is repeated for every candidate table in the table corpus, and the tables with a mapping are used to compute values for the query attribute column.

One of the earliest publications related to table extension is Cafarella *et al.* [10], which is concerned with automating the search for relevant web tables. The approach does not automate table extension, but proposes a set of operators that can be used by a user to search for tables, extract their context, and to extend a table schema with additional attributes.

In InfoGather [53], the authors develop a holistic matching and augmentation framework that allows table extension. They address the problem of spuriously matched tables by developing a holistic matching framework based on topic sensitive PageRank (TSP). The authors present three operations, one operation for attribute discovery, that aims at discovering attribute names for a given set of entities, and two operation for entity augmentation (augmentation-by-attribute and augmentation-by-example) that are covered in Section 3.4.

[1] `https://lucene.apache.org/`

For the attribute discovery operation, table relatedness scores are calculated based on topic set of tables which include tables that overlap with query entities. To discover attribute names, they aggregate column labels from multiple matched tables based on the tables relatedness scores. Later, this work is extended in InfoGather+ [56] to provide a better performance with numeric and time-varying attributes. It first builds a semantic graph that annotates attributes with meta-information such as unit, scale, and timestamp. Then, the system uses this graph to compute matches between columns. A similar operation to attribute discovery is presented in EntiTables [57]. The authors propose smart assistance that helps to extend a query table with additional column labels. To discover attribute names, the authors introduced a similarity function between tables that considers overlap between entities, column labels, and string similarity of the table captions. An improved version of this approach is proposed in Table2Vec [17], where the probabilistic model for measuring relevance is replaced with neural model which derives term embeddings for row and column population tasks.

Bhagavatula *et al.* [3] introduce WikiTables, which, given a query table and a corpus of tables extracted from Wikipedia, identifies columns from the tables in the corpus that would make relevant additions to the query table. They first identify a reference column in the query table to use for joining, then find a corpus table with a column similar to the reference column, and perform a left outer join to augment the query table with an automatically selected column from the corpus table.

Lehmberg *et al.* [32] propose an approach, called Mannheim Search Join Engine (MSJ), with the same goal as WikiTables but focused on handling tens of millions of tables from heterogeneous sources. MSJ is a table extension system that uses a corpus of web tables as a data source and automates the search and integration tasks. The system accepts a set of query entities and an optional query attribute as an input table. It searches for tables describing query entities (*search task*), and then selects relevant columns from the top-k candidate tables to merge (*integration task*). The candidate tables are joined using a multi-join operation - a series of left outer join operations with the

query table. As a result, the engine extends the input table with additional attribute columns. The case, when a query attribute is provided, is described in Section 3.4.

A more recent work that addresses the problem of extending a query table with additional columns is presented in [45]. The authors propose to search for candidate tables by looking for table columns that overlap with query table key column, then, using the notion of functional dependency, select columns from the candidate tables suitable for the table extension. As in MSJ [32], similar columns are grouped and consolidated. The resulting columns are then ranked using a relatedness score.

3.5 Entity Set Expansion

Informally, the goal of entity set expansion (ESE) is: given a small set of entities referred to as *seed set*, find other entities that semantically belong to the same set. For instance, having $Washington$, $Beijing$ and $Moscow$ as a seed set, it can be expanded with other capital cities like $London$, $Berlin$, $Seoul$ and so on. Entity set expansion is closely related to table expansion as it can be applied to expand subject column of a query table.

A considerable amount of research regarding ESE has been done in the literature and multiple methods to perform ESE have been proposed. Set expansion methods have been applied to textual data sources such as web documents [6, 39], web search query logs [40], or lists [25] and tables [48] extracted from web pages, among others.

SEAL (Set Expander for Any Language) [30, 50, 51] uses a technique to automatically construct wrappers for retrieving "lists" of items on HTML pages and is capable of handling various languages. SEAL uses the Google search engine to retrieve web pages giving a seed set as search keywords. It finds occurrences of seed entities in pages, builds a heterogeneous graph of web pages, wrappers, seeds and candidates, and then uses the graph to define a pattern for candidates extraction from the web pages (i.e. suffix and prefix based extractors). Web pages, wrappers and candidate entities are modeled

as nodes in a graph, and random walk techniques are used to rank candidates. Chen *et al.* [12] improves this approach by leveraging a page-specific extractor built in a supervised manner. SetExpan [47] also uses patterns to extract candidate entities similarly to SEAL. It contains two steps, a context feature selection step and an entity selection step. During the context feature selection, top-ranked features are selected based on expanded entities set. Then, at the entity selection step, these representative contexts are used to retrieve subsets of entities, which are then used to construct a ranked list of entities.

STEP [22] is also based on SEAL, where the pattern generation is extended to work with tuples of n-elements. The approach builds a query string from seed set and uses a search engine to retrieve relevant web pages. Using the same approach as SEAL, the authors propose to build a heterogeneous graph of web pages, wrappers, seeds and candidates, and then use it to define a pattern for candidates extraction. The main difference from SEAL is that the pattern, in addition to suffix and prefix, also includes middle-contexts (text patterns between elements of a tuple).

He *et al.* [25] propose the SEISA system that uses query logs along with web lists. Instead of using random walk ranking, the authors design an iterative similarity aggregation function. The similarity function is graph-based. Entities and lists are viewed as nodes in a bipartite graph. The similarity between two entities is established based on the set of list nodes that they are connected to. The iterative technique of SEISA does not add a set of entities in each iteration, instead, it expands the set in each iteration based on coherence and relevance scores of each entity with the previously expanded set (or a seed set in the first iteration).

Chapter 4

Proposed Approach

This chapter presents our approach to the problem of table expansion. First, we present some preliminaries on schema-matching and a formal data model. Next we show a naive approach that tries to apply an entity set expansion (ESE) method to the problem, before describing the proposed approach that overcomes the limitations of the naive approach.

4.1 Preliminaries

A table schema of a set of entities may include a set of attributes that describe those entities. We can infer that for an entity to semantically belong to a table it should have the attributes of the table. For example, a table with medals count for an Olympic game can have a column for a country name (key column) and columns for gold, silver and bronze medals counts (attribute columns). In order for a country to be in that table, it had to participate in that game to have the relevant attributes. Thus, to expand a query table based on a table corpus, entities can be obtained from tables with the same schema as a query table. Entities that can be added to the query table are referred to as *candidate entities*.

Schema matching

Consider tables in a corpus T, where each table has some schema. To expand a query table t^q, we can retrieve candidate entities from tables $t \in T$ with a schema that contains the attributes of the query table.

However, web tables do not provide any information about their attributes as their schema is unknown. Identifying semantic correspondences between attributes of two schemas is the task of schema matching. Schema matching approaches can be classified into two main types - schema-based and instance-based [43]. Schema-based methods primarily use labels, column data types, and other schema information to find similar attributes. Instance-based methods exploit properties of the underlying data, possibly in combination with attribute labels, to derive matches.

The common approach for schema matching used in the related work (finding related tables [42], table extension [17, 57], and entity augmentation [20, 32]) is to leverage schema level information along with table contextual information related to a table. Entity augmentation [17, 57], for example, requires knowledge of column labels and table meta-information to match it with an augmentation attribute.

Although, schema level information and table meta-data are useful in many cases, using them for schema matching poses several problems. First of all, about 20% of tables on the Web miss headers entirely [42], and about 6.5% of headers are missing some of labels [52], so-called "implicit" labels [21]. Moreover, header labels are often ambiguous as short headers (excluding the empty string) are used in more than 80% of all headers [52]). Secondly, a web table and its containing page might lack contextual information related to the table, e.g. tables without a title and pages that contain only tables, such as *name* or *value*. These types of labels provide little to no information about the meaning of an attribute. Finally, schema-level approaches require additional pre-processing (like identifying column type [32], building a semantic graph [56]) and using external knowledge-bases to identify header label synonyms.

Instead of relying on schema level information, our approach exploits instance level information to match table schemas. Instance-based schema matching can be performed vertically and horizontally [5]. A vertical approach matches each attribute of a schema separately by extracting properties about the attributes. These properties include column data type, distribution of characters, average string length, etc. Two attributes are matched when they

have similar properties. A horizontal approach, on the other hand, checks for "duplicate" rows across tables, i.e., rows with the same or similar values in different tables. This approach is considered to be more precise in differentiating attributes with similar domains, as only attribute values from tuples describing the same real-world entity are compared [29]. Correspondence between attributes is derived by checking for the same or similar data values among the duplicate rows of an entity. A similar approach is proposed in SearchJoins [32], where the existence of duplicates within data sets is exploited to identify matching attributes.

Data model

We disregard the header part of tables, as our approach uses instance-based schema matching, and define a table as a set of rows, $t = \{r_i\}$. Rows r are typically modeled as n-tuples from a cross product of the attribute domains of the table, i.e., $r \in A_1 \times ... \times A_n$. This model implicitly imposes an order on the attributes. Such an attribute order is arbitrary, and in a web table collection one may have several tables representing similar entities, but with a different order of the attributes.

With our assumption that the key of a table is a single attribute, say a_1, we can represent table t in a way that is independent of an attribute order as follows: Let $a_1, ..., a_n$ be the n attributes of a table t with m rows, having n associated attribute domains $A_1, ..., A_n$. Then, the table can be represented as a set $e(t)$ [1] of triples (k_i, a_j, v_{ij}), where, for all $1 \leq i \leq m$ and all $2 \leq j \leq n$:

- k_i is the key of the i-th row, i.e., $k_i \in A_1$,
- a_j is the j-th attribute with domain A_j, and
- v_{ij} is the value of the j-th attribute in row i, i.e., $v_{ij} \in A_j$.

The i-th row (tuple) of the original table corresponds then simply to the set $r_i = \{(k_i, a_j, v_{ij}) \mid 2 \leq j \leq m\} \subseteq e(t)$. Thus, it is straightforward to

[1] We choose the letter e in $e(t)$ to indicate that this representation is closer to a formal representation of an entity set than a relational table, because of the independence of the attribute order.

reconstruct from a set $e(t)$ the original table t, as well as any equivalent table, which differs from t only by the the order of the attributes.

Note that this representation of a table can be further simplified if a table has only a key column and one attribute column. In this case, we can drop the attribute indicator and represent a table just as a set of key-value pairs. We will make use of this simplification later in this chapter.

4.2 A Naive Approach

If we consider rows of tables as entities, a table t with n rows can be viewed as a set of entities $t = \{e_1, ..., e_n\}$ and the table expansion problem becomes similar to the entity set expansion (ESE) problem, which has been studied in the literature [25, 47, 48]. The standard solution is to leverage inter-dependencies that arise naturally in lists and tables by using co-occurrences to find entities that belong to the same "concept". The same solution could be applied to the table expansion problem, if rows are considered as entities.

We will demonstrate the approach on a simple example where a query table t^q and three tables t_1, t_2 and t_3 from table corpus T are provided, as shown in Figure 4.1.

Query table t^q has two attributes - "Country" and "City", table t_1 - "Country" and "Capital", table t_2 - "Country", "Capital" and "Language", and table t_3 - "Country", "Language" and "Currency". The first column of all tables is the key column, and the other columns are attribute columns.

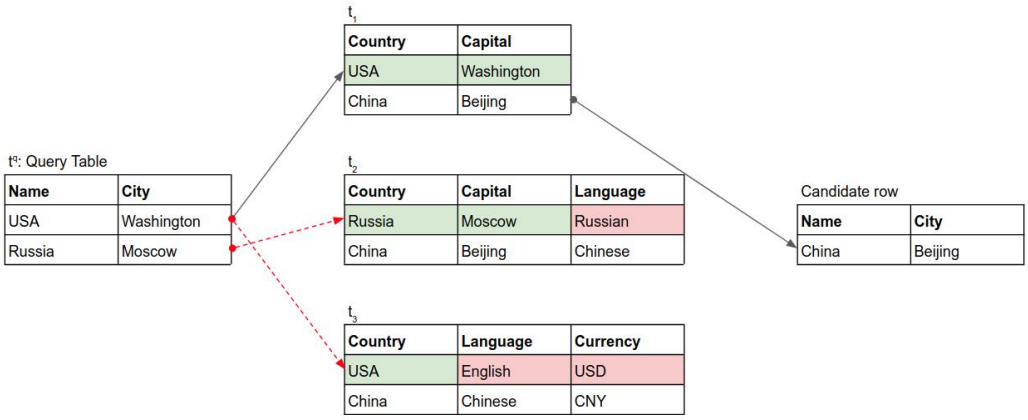

Figure 4.1: Example 1 of the naive approach.

In this example, the row $\langle USA, Washington \rangle$ from the query table appears only in table t_1 and the row $\langle Russia, Moscow \rangle$ does not appear in any table. It is notable that the row $\langle Russia, Moscow, Russian \rangle$ in table t_2 partially matches the query record $\langle Russia, Moscow \rangle$ but has an additional column, and row $\langle USA, English, USD \rangle$ from table t_3 only matches the key value in the query record $\langle USA, Washington \rangle$. Since the row $\langle China, Beijing \rangle$ co-occurs with rows from t^q in table t_1, it is counted as evidence that the row is relevant, and the row $\langle China, Beijing \rangle$ can be a candidate entity.

Now, suppose the query table t^q has three attributes - "Name", "City" and "Language", and the other tables are the same as above, as shown in Figure 4.2.

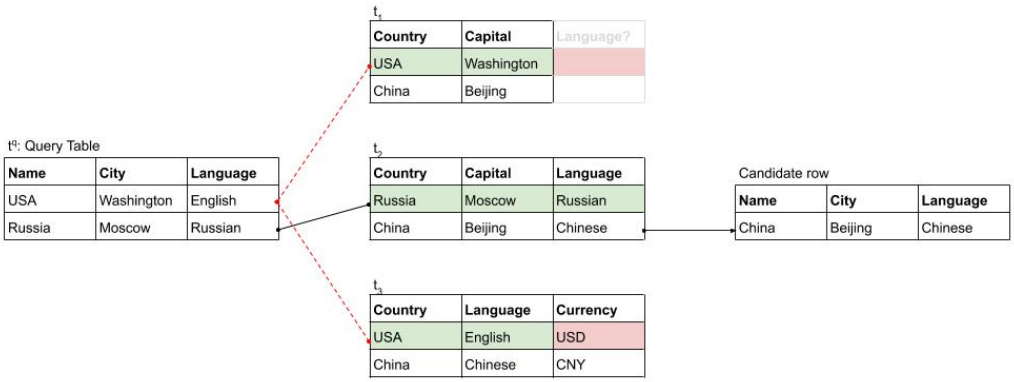

Figure 4.2: Example 2 of the naive approach.

In this case, row $\langle Russia, Moscow, Russian\rangle$ from table t^q occurs in table t_2. Row $\langle USA, Washington, English\rangle$ from the query table t^q does not appear in any table from table corpus T, however, it has partially matching rows in table t_2 (row $\langle USA, Washington\rangle$) and in table t_3 (row $\langle USA, English\rangle$). In this example, the row $\langle China, Beijing, Chinese\rangle$ can be a candidate entity as it co-occurs with row $\langle Russia, Moscow, Russian\rangle$ in table t_2.

Consider another example, where query table t^q has an attribute "Currency" instead of "Language", illustrated in Figure 4.3.

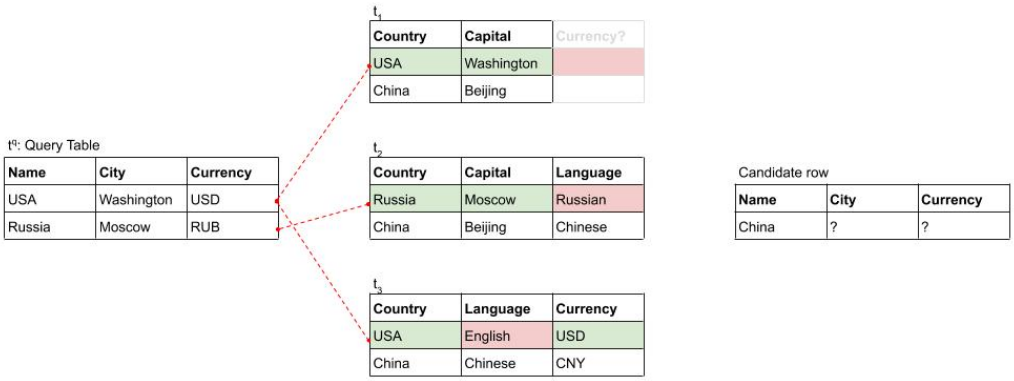

Figure 4.3: Example 3 of the naive approach.

In this example, none of the rows of the query table t^q occurs in any of the tables from table corpus T. As a result, none of the rows in tables t_1, t_2 and t_3 can be a candidate entity. Note, however, that the row $\langle USA, Washington, USD\rangle$ from query table t^q has a partially matching row $\langle USA, Washington\rangle$ in table t_1 and the row $\langle USA, English, USD\rangle$ in table t_3. These partially matching rows combined could represent a row $\langle USA$, $Washington, USD\rangle$ that can be added to the query table. However, the naive approach looks for occurrences of all values of a row and therefore cannot leverage partial information. Due to this limitation, it becomes less likely to find a matching row with every additional attribute in the query table.

4.3 The T-REx System

The naive approach looks for entire rows of a query table in other tables, and to find a matching row, a table has to have the same schema including the order of attributes. Since tables on the web can be created in an arbitrary way, most web tables will not have the same schema as the query table. To overcome the limitation of the naive approach, we propose a solution where each table is split into a set of sub-tables consisting of two columns each, a key column and an attribute column. These tables referred to as entity-attribute binary (EAB) relations [53].

For example, in Fig.4.3, we can split the query table into two binary relation tables - one table with "Name" and "City" attributes, and another table with "Name" and "Currency" attributes. We can also split other tables using the same strategy. As a result, it would be possible to match rows of sub-tables of the query table with sub-tables of tables in the table corpus. See Fig. 4.4.

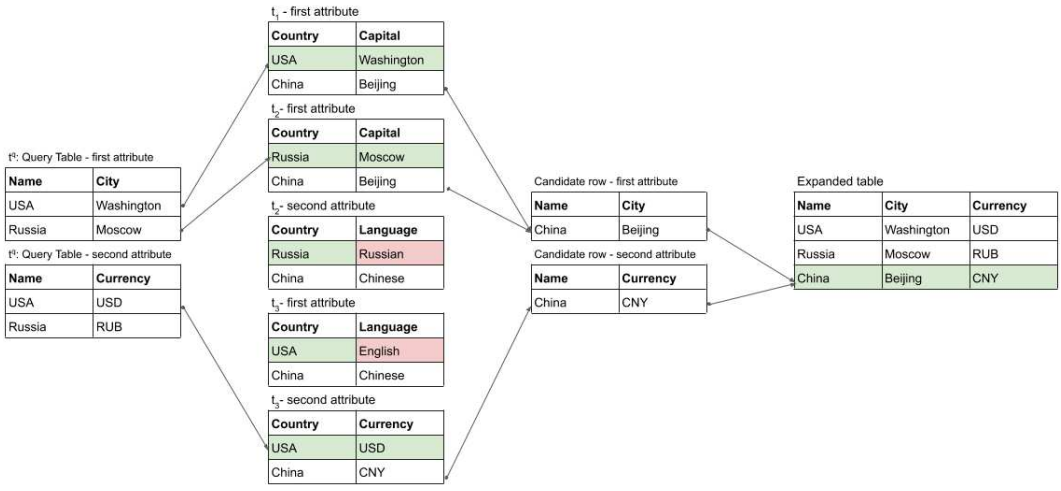

Figure 4.4: Using binary relations.

We will first introduce the approach with the assumption that all web tables contain only one attribute. Afterwards, we will remove this simplification and describe the approach for tables with any number of attributes.

Figure 4.5: Candidate tables.

4.3.1 Table expansion for tables with one attribute

Let's look at the problem of table expansion when every table has only two columns - a key column and an attribute column. Each table can be represented as a set of *key-value pairs*, i.e. $t = \{(k_1, v_1), ..., (k_n, v_n)\}$ and schema-matching is significantly simplified as only one attribute has to be matched.

If a table t has values in its key column that also occur in the query table t^q, this could indicate that the tables t and t^q have matching schemas. We refer to such tables as *candidate tables* [56], to keys of a candidate table that are not query keys as *candidate keys*, and to tuples with a candidate key as *candidate pairs*.

We illustrate an example of candidate tables in Fig. 4.5. Tables t_1 and t_2 are candidate tables as they both have key "USA", however, t_3 is not a candidate as it does not have any keys from the query table.

A candidate pair (k, v) represents a candidate entity, where key k is a name of an entity and value v is a query attribute value of the entity. Thus, candidate entities can be collected from candidate tables and added to the query table. However, since schema matching is approximate, several candidate pairs with the same key k can have different attribute values, i.e. a candidate key k can be present in a pair (k, v_1) in one table and in a pair (k, v_2) in another.

This could happen, for instance, if the entity is described in the table corpus with multiple attributes (each of which would be represented by an individual table under our current simplifying assumption), duplicate tables with small changes (e.g. Olympic Games medal counts updates after "doping" scandals), or due to a data entry error in one of the tables of the table corpus.

To decide which value to choose for the query attribute of a particular candidate key, we propose a score that measures the relevance of a given candidate key-value pair to a query table.

Key-value pair relevance score

Let's consider a situation, where in a set T_1 of candidate tables a candidate pair (k, v_1) occurs, and in a set T_2 of candidate tables a candidate pair (k, v_2) occurs. Then, key k can be associated either with attribute value v_1 or attribute value v_2. To resolve this conflict, we can compare the number of tables that contain candidate pair (k, v_1) ($|T_1|$) with the number of tables that contain candidate pair (k, v_2) ($|T_2|$). However, the number of tables does not include any information about the similarity between tables in the table corpus and the query table. Since, an instance-based matching determines the correspondence between schemas based on the size of overlap in content, we propose to count co-occurrence of a given candidate pair with tuples of the query table in all tables in a corpus.

We refer to the size of intersection between two tables as the number of matches and define it as follows:

Definition 4.3.1. Matches (M^+). Let t^q be a query table and t be a table, then the number of matches in table t with tuples from t^q is defined as:

$$M^+(t^q, t) = |\{(k, v) \mid (k, v) \in t^q \text{ and } (k, v) \in t\}|. \tag{4.1}$$

In Fig. 4.6, both query key-value pairs $<$*USA, Washington*$>$ and $<$*Russia, Moscow*$>$ are found in a candidate table t_1, therefore the number of matches between tables will be equal 2.

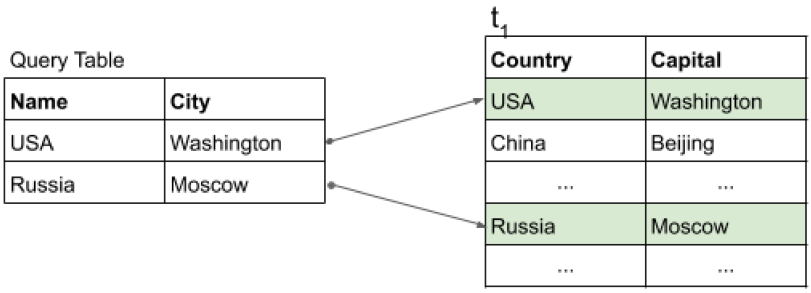

Figure 4.6: Matches of query tuples.

Following the definition of Matches M^+, a pair $(k_2, v_2) \in t_1$ is considered to be a duplicate of $(k_1, v_1) \in t^q$ if keys and values match between the pairs, i.e. $k_2 = k_1$ and $v_2 = v_1$, and it is an indication that attributes of table t_1 and t^q are similar. Another case, that needs to be considered, is when keys match between two pairs but values do not, i.e. $k_2 = k_1$ and $v_2 \neq v_1$. Despite having the same keys as query table in key column, the attributes are not the same.

The example illustrated in Fig. 4.7 demonstrates a case when keys of the query tuples match with keys in table t_1, however, the values do not match.

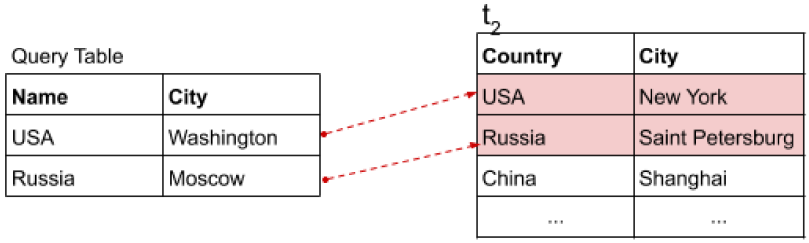

Figure 4.7: Mismatches with query tuples.

Next, we will define the number of mismatches.

Definition 4.3.2. Mismatches (M^-). Let t_q be a query table and t be a table. The number of mismatches in table t with tuples from query table t^q is defined as:

$$M^- (t^q, t) = |\{k \mid (k, v) \in t^q \text{ and } (k, v') \in t \text{ and } v \neq v'\}| . \tag{4.2}$$

We want to avoid counting matches in a table if there is more evidence that its attribute is more dissimilar than similar to the query attribute. For

instance, Fig.4.8 illustrates a case when tuples of table t_1 match all query tuples, which indicates that its attribute has a high similarity with the query table attribute. Table t_2 has two matches and one mismatch, due to an error or outdated information, nevertheless, the table t_2 has more matches than mismatches, which indicates some similarity of its attribute to the attribute of the query table. Table t_3 has 2 mismatches and only 1 match with the query table, which suggests that the attributes are more dissimilar and hence the attribute in t_3 represents a different concept.

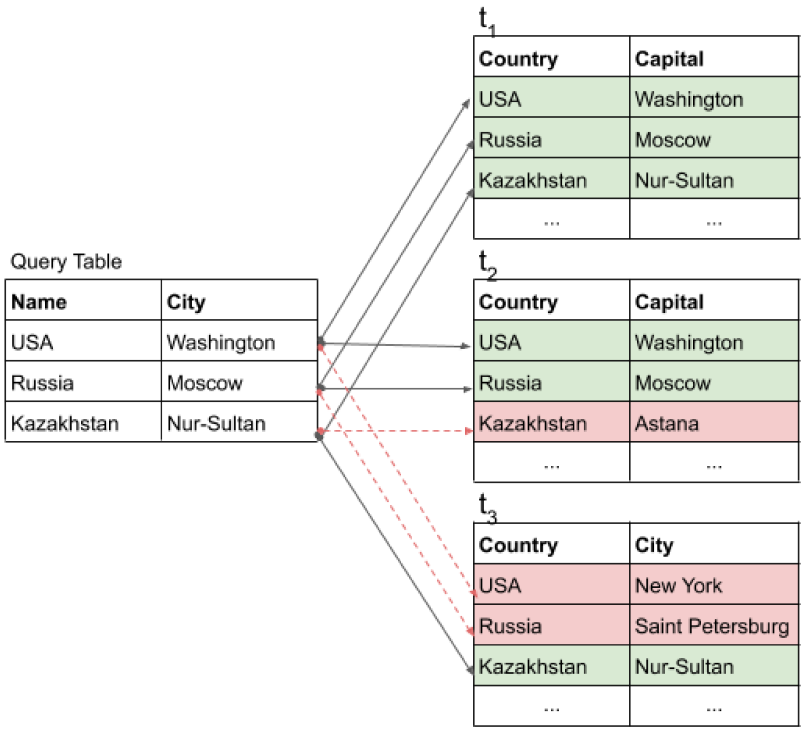

Figure 4.8: Match score example.

Next, we define the overall match score between a table and a query table by subtracting the number of mismatches from the number of matches between the two tables.

Definition 4.3.3. Match Score (M). Let t^q be a query table and t be a table, then the match score of table t is defined as:

$$M(t^q, t) = \begin{cases} M^+(t^q, t) - M^-(t^q, t), & \text{if } M^+(t^q, t) - M^-(t^q, t) \geq \alpha \\ 0, & \text{otherwise} \end{cases}, \quad (4.3)$$

where α ($\alpha > 0$) is a threshold on the number of matches over the number of mismatches in a table. A high threshold will only allow non-zero match scores for tables with high attribute similarity, but will reduce the tolerance to possible data entry errors. In our experiments, we use a threshold of $\alpha = 1$, which means that any table with more matches than mismatches will receive a non-zero match score. Match scores are used to calculate a key-value pair relevance score for each candidate pair, as defined next.

Definition 4.3.4. Key-Value Pair Relevance Score ($Score_{pair}$). Let T be a table corpus, $\boldsymbol{K}$ be the universe of candidate keys, $\boldsymbol{V}$ be the universe of values, (k, v) be a tuple where $k \in \boldsymbol{K}$ and $v \in \boldsymbol{V}$, and t^q be a query table. The relevance score of a candidate pair (k, v) to query table t^q is defined as:

$$Score_{pair}((k, v), t^q) = \sum_{\{t \,|\, t \in T \text{ and } (k,v) \in t\}} M(t^q, t) \tag{4.4}$$

The relevance score of a candidate key-value pair sums match scores of tables that contain the pair. A table that has more matches will proportionally contribute more to the score of the pair than a table that has fewer matches, as it has a higher chance representing the same concept as the query table attribute.

Let's consider an example illustrated on Fig. 4.9. Notice that table t_1 contains candidate tuple $\langle Brazil, Brasilia \rangle$ and tables t_2 and t_3 contain candidate pair $\langle Brazil, Sao\ Paulo \rangle$. The key of both candidate pairs is "$Brazil$" but associates query attribute value can be either "$Brasilia$" or "$Sao\ Paulo$".

Tuples of the table t_1 match all tuples from the query table and therefore its match score, M, is equal to 3. Attribute "Capital" of the table t_1 has high similarity to attribute "City" of the query table. Since table t_1 is the only table that contains the pair $\langle Brazil, Brasilia \rangle$, the relevancy score of the pair will be equal to 3.

The pair $\langle Brazil, Sao\ Paulo \rangle$ appears in the tables t_2 and t_3. Table t_2 has two matches and one mismatch, thus, its match score with the query table is equal to 1. Table t_2 contains all keys from the query table, meaning that the table describes entities of the same concept, however, it has one mismatch with

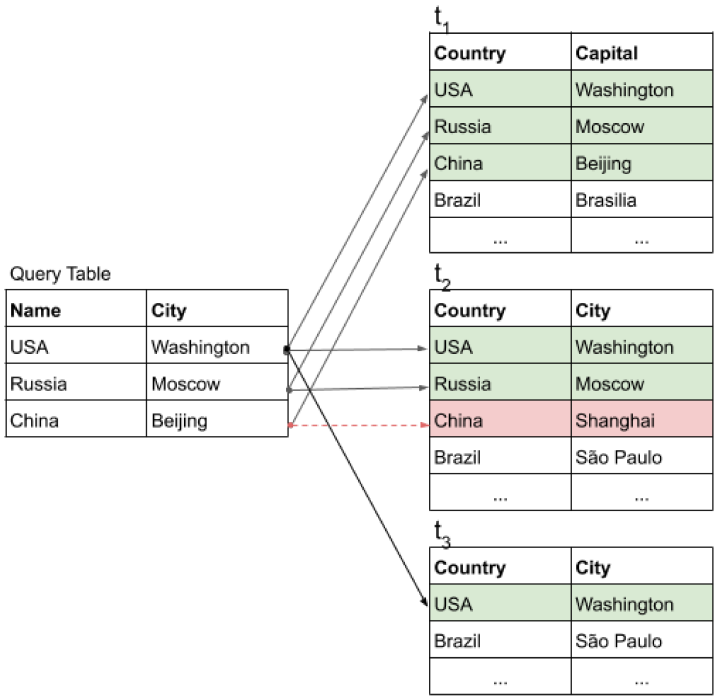

Figure 4.9: Key-value pair relevance score example.

the query table and it is an indication that its attribute might be different from the query attribute. The match score of the table t_2 is equal to 1. Table t_3 has only one match, which shows some similarity to the query table and its match score will be equal to 1. The relevancy score of the pair $\langle Brazil, Sao\ Paulo \rangle$ to the query table will be the sum of match scores of tables t_2 and t_3, and equal to 2.

Even though, the candidate pair $\langle Brazil, Sao\ Paulo \rangle$ occurs in two candidate tables, it has lower relevancy score as tables t_2 and t_3 have low match scores compared to table t_1 that contains candidate pair $\langle Brazil, Brasilia \rangle$.

4.3.2 Table expansion for tables with multiple attributes

For a table with multiple attribute columns, we can look at each attribute independently by splitting the table into a set of sub-tables of two columns (EAB relations), a key column and an attribute column. We use projections to split tables into sub-tables.

Definition 4.3.5. Projection (Π). Let t be a table with schema $A =$

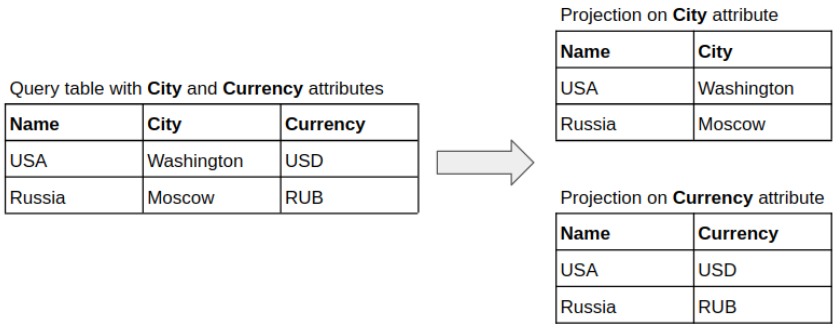

Figure 4.10: Projections of a query table.

$\{a_1, ..., a_m\}$ and a_1 be the key attribute. The projection of the table t on attribute a_1 and a_j $(j = 2, ..., m)$ is written as $\Pi_j(t)$ and returns a set of tuples of table t restricted to key attribute a_1 and attribute a_j.

Note that there can be $m - 1$ sub-tables of a table t with m columns, each for an attribute column of the table.

In Fig. 4.10, we illustrated how a query table with two attributes can be split into two sub-tables using projections. The initial query table on the left side contain attributes "City" and "Currency" and on the right side projections, one on attribute "City" and another on attribute "Currency".

Projections turn a table with multiple attributes into a set of sub-tables with a single attribute. This will allow us to apply attribute value selection using a similar approach as described in the previous section.

To account for the fact that tables can have multiple columns which are split into several projections, we have to adapt the calculation of the key-value pair relevance score to make sure that from a given table, at most one attribute (the most relevant attribute) contributes a non-zero match score to a particular key value pair.

Key-Value Pair Relevance Score

The key-value pair relevance score, $Score_{pair}$, uses the match score M, which computes a match score of a table with a single attribute. Let t^q be a query table with an attribute a and t be a candidate table with multiple attributes $b_1, ..., b_m$. A projection on an attribute a of query table t^q can, in principle,

match with projections on several attributes of candidate table t, resulting in several different, non-zero matching scores. That means that multiple columns of the same table could contribute to a specific key-value pair relevance score, when using the previous approach that would consider all projections independently. Therefore, we make sure that for a given query attribute only the "best matching" attribute of a multi-attribute table contributes to $Score_{pair}$ calculations for this query attribute. To select a single, best matching, attribute of a table with multiple attributes for the calculation of its match score with the query table, we propose to use the highest match score M between all projections of the candidate table.

Definition 4.3.6. Best Match Score (M^*). Let t_j^q be a projection on an attribute j of a query table t^q, and let t be a candidate table with m attribute columns, then the best match score of a t_j^q and the table t is defined as:

$$M^*(t_j^q, (k,v), t) = \max_{\{b \,|\, b \in attr(t) \,:\, (k,v) \in \Pi_b(t)\}} \{M(t_j^q, \Pi_b(t))\} \ . \tag{4.5}$$

For tables with multiple attributes, the key-value pair relevance score then uses this best match score when calculating a key-value relevance score, as defined below:

Definition 4.3.7. Key-Value Pair Relevance Score ($Score_{pair}$). Let T be a table corpus, $\boldsymbol{K}$ be the universe of candidate keys, $\boldsymbol{V}$ be the universe of values, (k,v) be a tuple where $k \in \boldsymbol{K}$ and $v \in \boldsymbol{V}$, and t_j^q be a projection on an attribute j of a query table t^q. The relevance score of a candidate pair (k,v) to sub-table t_j^q of query table t^q is defined as:

$$Score_{pair}((k,v), t_j^q) = \sum_{\{t \,|\, t \in T \text{ and } \exists b \in attr(t) \,:\, (k,v) \in \Pi_b(t)\}} M^*(t_j^q, (k,v), t) \tag{4.6}$$

The following sub-section will detail the pseudo-code of the framework.

4.3.3 Pseudo-code

In order to solve the table expansion, the proposed solution performs three main tasks:

1. Find a set of candidate tables T^c in a table corpus T for the expansion of a query table t^q.

2. Retrieve candidate key-value pairs for each attribute of query table t^q.

3. Form a set of candidate rows R to expand query table t^q.

The pseudo-code of the system is shown in Algorithm 1. The input of the algorithm is a set of relational web tables T and a query table t^q.

$keys(t)$ returns the set of keys of table t, and on line 1, K^q is initialized to be the set of keys of query table t^q. T^c is initialized on line 2 and includes all tables that contain at least one key k from the set of keys K^q of the query table. K is initialized on line 3 and includes all keys that belong to a candidate table t but are not among the keys K^q of the query table t^q.

A candidate row r consists of a set of 3-tuples of the form (k, j, v) where k is a candidate key, j is an attribute index of the query table t^q, v is an associated attribute value. The call $attrs(t^q)$ returns indexes of attributes of table t^q. The code from line 5 to line 23 retrieves the tuples to form candidate rows and collects them into a set Q, which is initialized on line 4.

The tuples (k, j, v) are constructed for each attribute j of the query table t^q in two parts. First, candidate key-value pairs (k, v) are retrieved from projections of candidate tables T^c and stored in a set P on lines 7-14. Since each key-attribute-value tuple of a candidate row must be associated with an attribute j of the query table t^q, the projection $\Pi_j(t^q)$ is used to find a projection with the highest number of matches in each candidate table $t \in T^c$. The tuples (k, v) from those projections are collected in one set of tuples P.

In the second part, for each candidate key k, a pair (k, v) is selected from all pairs in P associated with k, based on maximum $Score_{pair}$, and a tuple (k, j, v) is added to the set Q on lines 15-22.

As a result, after line 23, the set Q will have tuples (k, j, v), where value v belongs to a candidate table and has maximum $Score_{pair}$ with the key k to the projection $\Pi(t^q, j)$.

On lines 25-28 tuples from Q are grouped into a set of candidate rows R based on keys from K. On line 29, the set of candidate rows R is returned.

Algorithm 1: T-REx System

Data: T - set of relational web tables, t^q - a query table
Result: Set of tuples

1 $K^q \leftarrow keys(t^q)$
2 $T^c \leftarrow \{t \mid t \in T \text{ and } \exists k \in keys(t) : k \in K^q\}$
3 $K \leftarrow \{k \mid t \in T^c \text{ and } k \in keys(t) \text{ and } k \notin K^q\}$
4 $Q \leftarrow \emptyset$
5 **foreach** $j \in attrs(t^q)$ **do**
6 $P \leftarrow \emptyset$
7 **foreach** $t \in T^c$ **do**
8 **foreach** $b \in attrs(t)$ **do**
9 **if** $M(\Pi_j(t^q), \Pi_b(t)) > \alpha$ *and* $M(\Pi_j(t^q), \Pi_b(t)) = M^*(\Pi_j(t^q), t)$ **then**
10 $P \leftarrow \{(k, v) \in \Pi_b(t) \mid k \notin K^q\}$
11 **break**
12 **end**
13 **end**
14 **end**
15 **foreach** $k \in K$ **do**
16 **foreach** $(k, v) \in P$ **do**
17 **if** $(k, v) \in \underset{(k,v^*)\in P}{\text{argmax}}\ Score_{pair}((k, v^*), \Pi_j(t^q))$ **then**
18 $Q \leftarrow Q \cup \{(k, j, v)\}$
19 **break**
20 **end**
21 **end**
22 **end**
23 **end**
24 $R \leftarrow \emptyset$
25 **foreach** $k \in K$ **do**
26 $r \leftarrow \{(k, j, v) \mid (k, j, v) \in Q \text{ and } j \in attrs(t^q)\}$
27 $R \leftarrow R \cup r$
28 **end**
29 **return** R

4.3.4 Example

We will demonstrate the approach with an example, where a table corpus (see Figure 4.11) contains five tables: table t_1 - a table of capital cities, t_2 - a table of places where matches of an online championship took place including the

language of the broadcasting, $t3$ - a table of languages and currencies, $t4$ - a table of languages and $t5$ - a table of currencies.

t_1: Capital Cities

Country	Capital
China	Beijing
USA	Washington
Russia	Moscow
Spain	Madrid

t_2: Championship Matches

Country	City	Language
USA	Washington	English
Russia	Moscow	Russian
Spain	Madrid	English

t_3: Languages and Currencies

Country	Language	Currency
China	Chinese	CNY
USA	English	USD
Russia	Russian	RUB
Spain	Spanish	EUR

t_4: Languages

Capital	Language
Beijing	Chinese
Moscow	Russian
London	English

t_5: Currencies

Country	Currency
France	EUR
Australia	AUD
Canada	CAD

Figure 4.11: Table Corpus T.

Assume, a user provided query table t^q in Figure 4.12, which has three columns and two rows. We can infer from the table that the user wants to retrieve a list of countries with their capital cities and languages.

t^q

Country	Capital	Language
China	Beijing	Chinese
USA	Washington	English

Figure 4.12: Query Table t^q.

Task 1. Find a set of candidate tables.

Keys from the query table t^q are used to filter tables from the table corpus T to obtain tables that contain query keys in their key column. Tables t_1, t_2 and

t_3 have keys that match query table keys, and tables t_4 and t_5 do not have a key that match any query table keys (see Figure 4.13). Therefore, only tables t_1, t_2 and t_3 are selected as candidate tables.

t_1: Capital Cities

Country	Capital
China	Beijing
USA	Washington
Russia	Moscow
Spain	Madrid

t_2: Championship Matches

Country	City	Language
USA	Washington	English
Russia	Moscow	Russian
Spain	Madrid	English

t_3: Languages and Currencies

Country	Language	Currency
China	Chinese	CNY
USA	English	USD
Russia	Russian	RUB
Spain	Spanish	EUR

t_4: Languages

Capital	Language
Beijing	Chinese
Moscow	Russian
Washington	English

t_5: Currencies

Country	Currency
France	EUR
Australia	AUD
Canada	CAD

Figure 4.13: Tables filtered by appearance of keys of the query table. Matches with query table keys are indicated with blue background, and excluded tables are greyed out.

Task 2. Retrieve candidate key-value pairs.

After candidate tables have been found, the next step is to retrieve candidate keys and associated values for each attribute of the query table t^q. The query table t^q has two attributes, "Capital" and "Language", and the following procedure is performed for each of them to retrieve candidate key-value pairs.

Query attribute "Capital"

First, a projection of the left-most attribute "Capital" is created (See Figure 4.14).

$\Pi_1(t^q)$

Country	Capital
China	Beijing
USA	Washington

Figure 4.14: Projection of the first attribute of the query table t^q.

To find matches between candidate tables and the projection $\Pi_1(t^q)$, each candidate table is also considered as a set of projections. For instance, table t_1 has only one attribute column and therefore has only one projection. The projection has matches with both key-value pairs $<$*China, Beijing*$>$ and $<$*USA, Washington*$>$ from $\Pi_1(t^q)$. Since the column has two matches and is the column with maximum number of matches, these candidate keys and associated values ($<$*Russia, Moscow*$>$ and $<$*Spain, Madrid*$>$) will be selected as candidate pairs, giving an initial value of the relevance score $Score_{pair}$ equal to 2 (number of matches in t_1) for each pair. The matches are highlighted in the Figure 4.15.

t_1: Capital Cities

Country	Capital
China	Beijing
USA	Washington
Russia	Moscow
Spain	Madrid

t_2: Championship Matches

Country	City	Language
USA	Washington	English
Russia	Moscow	Russian
Spain	Madrid	English

t_3: Languages and Currencies

Country	Language	Currency
China	Chinese	CNY
USA	English	USD
Russia	Russian	RUB
Spain	Spanish	EUR

Figure 4.15: Table columns matching projection $\Pi_1(t^q)$. Matches with query table keys are indicated with blue background, matches with attribute values are indicated with green background, and mismatches are indicated with red background.

A summary of resulting Best Match Score M^* and candidate key-value

pairs of each candidate table is presented in Table 4.1. Both of the projections of table t_3 have negative Match Score M as they have mismatches and no matches with $\Pi_1(t^q)$. As a result, none of them is used for retrieving candidate key-value pairs.

Table	**Best Match Score** M^*	**Candidate key-value pairs**
t_1	2	{<Russia, Moscow>, <Spain, Madrid>}
t_2	1	{<Russia, Moscow>, <Spain, Madrid>}
t_3	0	

Table 4.1: Best Match Score M^* and candidate key-value pairs of candidate tables for projection of query table on attribute "Capital".

After collecting candidate pairs from all three candidate tables, two candidate pairs are found - $<$*Russia, Moscow*$>$ and $<$*Spain, Madrid*$>$. Both pairs have a score of 3, which represents several matches with pairs from $\Pi_1(t^q)$ in candidate tables t_1 and t_2 with positive Match Score M in at least one projection. See Table 4.2.

Candidate key-value pair	**Occurs in tables**	**Relevance Score** $Score_{pair}$
<Russia, Moscow>	t_1, t_2	3 (2 + 1)
<Spain, Madrid>	t_1, t_2	3 (2 + 1)

Table 4.2: Candidate key-value pairs their relevancy score for query attribute "Capital".

The keys in the candidate key-value pairs are all different from each other, which means that each key has only one value that it can be associated with. For illustration purposes, these pairs can be used to expand the query projection $\Pi_1(t^q)$ (Figure 4.16).

Expanded $\Pi_1(t^q)$

Country	**Capital**
China	Beijing
USA	Washington
Russia	Moscow
Spain	Madrid

Figure 4.16: Resulting expanded projection $\Pi_1(t^q)$.

Query attribute "Language"

After retrieving candidate key-value pairs for projection $\Pi_1(t^q)$, the system repeats the procedure for the projection of the next attribute (see Figure 4.17).

$\Pi_2(t^q)$

Country	Language
China	Chinese
USA	English

Figure 4.17: Projection of column 2 of the query table t^q.

Table t_1 does not have any matches with projection $\Pi_2(t^q)$, table t_2 has one match on attribute "Language" and table t_3 has two matches on attribute "Language". The matches and mismatches are illustrated on Figure 4.18.

t_1: Capital Cities

Country	Capital
China	Beijing
USA	Washington
Russia	Moscow
Spain	Madrid

t_2: Championship Matches

Country	City	Language
USA	Washington	English
Russia	Moscow	Russian
Spain	Madrid	English

t_3: Languages and Currencies

Country	Language	Currency
China	Chinese	CNY
USA	English	USD
Russia	Russian	RUB
Spain	Spanish	EUR

Figure 4.18: Table columns matching projection $\Pi_2(t^q)$. Matches with query table keys are indicated with blue background, matches with attribute values are indicated with green background, and mismatches are indicated with red background.

A summary of resulting Best Match Score M^* and candidate key-value pairs of each candidate table is presented in Table 4.3. Attribute "Language"

of table t_2 has a Match Score M equal to one as it has one match and no mismatches, and attribute "Language" of table t_3 has two matches and no mismatches as well.

Table	**Best Match Score M^***	**Candidate key-value pairs**
t_1	0	
t_2	1	{<Russia, Russian>, <Spain, English>}
t_3	2	{<Russia, Russian>, <Spain, Spanish>}

Table 4.3: Best Match Score M^* and candidate key-value pairs of candidate tables for projection of query table on attribute "Language".

From all candidate tables, three candidate key-value pairs have been retrieved - $<$*Russia, Russian*$>$, $<$*Spain, English*$>$ and $<$*Spain, Spanish*$>$ (See Table 4.4). Key *Russia* has only one value to be associated with and it has Relevance Value Score $Score_{pair}$ equal to 3. Key *Spain*, on the other hand, has a conflict as it can be associated with two values - *English* and *Spanish*. To resolve the conflict the pair with highest Relevance Value Score $Score_{pair}$ is chosen. In this example, pair $<$*Spain, English*$>$ has a score equal to 1 and $<$*Spain, Spanish*$>$ - equal to 2. Therefore, pair $<$*Spain, Spanish*$>$ is selected.

Candidate key-value pair	**Occurs in tables**	**Relevance Score $Score_{pair}$**
<Russia, Russian>	t_2, t_3	3 (1 + 2)
<Spain, English>	t_2	1
<Spain, Spanish>	t_3	2

Table 4.4: Candidate key-value pairs their relevancy score for query attribute "Language".

For illustration purposes, these pairs can be used to expand the query projection $\Pi_2(t^q)$ (Figure 4.19).

Expanded $\Pi_2(t^q)$

Country	**Language**
China	Chinese
USA	English
Russia	Russian
Spain	Spanish

Figure 4.19: Resulting expanded projection $\Pi_2(t^q)$.

Task 3. Form a set of candidate rows.

After candidate key-value pairs are retrieved for all attributes of the query table t^q, candidate rows can be formed based on candidate keys and associated values for each attribute. For candidate key *Russia*, value *Moscow* has been retrieved for projection $\Pi_1(t^q)$ on attribute "Capital" and value *Russian* for projection $\Pi_2(t^q)$ on attribute "Language". Therefore, a candidate row $<$*Russia, Moscow, Russian* $>$ is formed. For candidate key *Spain*, values *Madrid* and *Spanish* have been retrieved for projections $\Pi_1(t^q)$ and $\Pi_2(t^q)$, respectively. As a result, a candidate row $<$*Spain, Madrid, Spanish* $>$ is formed. The candidate rows are then added to the given query table (see Figure 4.20).

Expanded Table

Country	**Capital**	**Language**
China	Beijing	Chinese
USA	Washington	English
Russia	Moscow	Russian
Spain	Madrid	Spanish

Figure 4.20: Expanded Table.

Chapter 5

Experimental Evaluation

In this chapter, we present an experimental evaluation of the proposed method. During the experimentation phase, our goal was to evaluate the T-REx system by comparing its performance to alternative solutions. We were interested in conditions at which our system outperforms the alternatives, at which it falls behind, and how the input influences the result. The goals of the evaluation are:

- To compare the entity names (subject column) expansion of the system with the result of an ESE approach.
- To compare the attribute value predictions of the system with those of EA approaches.
- To study the sensitivity of results (precision and recall) with respect to how frequent the key-value pairs of query tuples are in the table corpus.
- To study the sensitivity of results quality with respect to the number of examples in a query table.

5.1 Dataset

We used the WDC Web Table Corpus [52] for our evaluation. It contains Web tables extracted [19] from the Common Crawl [1] excluding layout HTML

[1] `http://commoncrawl.org/`

tables. WDC Web Table Corpus is part of the Web Data Commons project [2] and contains 233 million Web tables.

In this work, the English-language relational web tables were used. Each table has been accompanied by meta- and contextual information in the corpus. Meta-information includes details such as table orientation, key-column index, whether the table has a header row and its index. Contextual information covers page title, page URL, table caption and page keywords, and were described in Chapter 2.

5.2 Experimental setup

5.2.1 Ground truth tables

To evaluate the proposed system, two types of ground truth were used from the table corpus: (1) a set of curated tables, and (2) a set of random tables. During evaluations we used ground truth tables of both types.

Curated Tables. A set of 5 curated tables, as listed in Table 5.1, was obtained from Wikipedia [3], The Guardian [4] and WikiData [5], representing information from common domains. Tables generated from WikiData using SPARQL [6] describe a class of objects with multiple attributes, for example, all countries with their capital cities and currencies. In this case, to retrieve all records, the methods have to find multiple tables from the corpus.

Title	# of columns	# of rows	Source
Language, currency and region by country	4	236	WikiData
2008 Summer Olympics medal table	5	84	Wikipedia
World's top 100 footballers 2012	6	100	TheGuardian.com
Top 100 best selling books of all time	6	100	TheGuardian.com
The greatest films of all time	8	25	TheGuardian.com

Table 5.1: Curated tables.

Random Tables. Random ground truth tables, which represent arbitrary domains, are extracted from the table corpus. They are then removed from

[2] http://webdatacommons.org/webtables/
[3] https://www.wikipedia.org/
[4] https://www.theguardian.com/
[5] https://www.wikidata.org/
[6] https://query.wikidata.org/

the corpus during search to ensure that these tables are not used in the experiments. We ensure that each random ground truth table has at least 5 rows and 2 columns. Table 5.2 describes examples of random tables that were selected as ground truth.

Title	# of columns	# of rows
1994 USMS Top 10 SCM for Women 45-49	7	11
CyberSports	16	17
Citizens Advice Bureau - Counselling & Advice Services in...	3	11
New Mexico vs Sacramento State (Oct 30, 2009) - Sacramento...	9	18
iTunes - Music - Shangaan Electro - New Wave Dance Music...	7	13
PHP: ingres_connect - Manual	5	18

Table 5.2: Examples of random tables.

5.2.2 Query Tables

Query tables are generated by selecting a subset of tuples from the ground truth tables. Depending on the purpose of an experiment, the query tuples (examples) are either selected randomly or based on the frequency of their key-value pairs in the table corpus. To select tuples by frequency, all tuples of the ground truth table are sorted by the frequency of their occurrence in the corpus. Tuples at the top of this sort order are referred to as "head" tuples, at the bottom as "tail" tuples and in the middle as "mid" tuples. We differentiate tuples by frequency in order to measure sensitivity with respect to this characteristic since the key-value relevancy score in T-REx is based on co-occurrence frequency of key-value pairs.

We intentionally limit the size of the query tables to 5 as the 90 million relational tables in the Web Data Commons web tables corpus have a median number of 6 rows [30, 52], and the majority of web tables are very small [29], see Table 5.3 for table statistics of the WDC corpus.

	min.	max.	average	median
Columns Horizontal Tables (attributes)	2	18,106	5.20	4
Rows Horizontal Tables (entities)	2	17,033	14.45	6
Columns Vertical Tables (entities)	3	16,142	8.44	5
Rows Vertical Tables (attributes)	1	486	3.66	3

Table 5.3: Statistics about the columns and rows in the WDC 2015 Corpus.

5.2.3 Baseline Methods

The problem of table expansion is not addressed by any other work, however, it can be conceptually divided into two separate tasks: subject column (entity names) expansion and attribute value prediction. These tasks are illustrated in Fig. 5.1. Given a query table, first, the entity names are expanded and then values are predicated for attributes. Therefore, the problem can be addressed by combining an entity set expansion (ESE) method to expand entity names with an entity augmentation (EA) method to predict values for query attributes.

We compare the expansion of entity names (subject column) of T-REx with SEISA as an ESE method. For the evaluation of predicted values for attributes, we compare T-REx with REA and Search Joins.

SEISA. We have implemented SEISA based on the original publication [25]. As an input, it accepts a seed set of subject names of a query table and a parameter α that balances the emphasis between relevance and coherence scores.

REA. An implementation of the system proposed in [20] is provided by its authors [7] and is designed to work with numeric values. It includes a tool to build a Lucene index over a web tables corpus and it uses Redis [8] for in-memory caching functionality. The system's input is a set of entity names as well as keywords describing an attribute label. It has a limit of 100 on the number of top augmentations that are returned.

MSJ. The implementation of the Mannheim Search Joins Engine was provided by its authors [9]. The solution includes tables corpus indexing functionality that creates a Lucene index with a required structure, which was used to index the data set. The engine accepts subject names as an input, and the query can additionally be constrained to a specific column label (augmentation attribute).

An additional configuration file is provided and allows adjusting various

[7] `https://github.com/JulianEberius/REA`
[8] `https://redis.io/`
[9] `https://github.com/MannheimSearchJoinsEngine/MannheimSearchJoinsEngine`

parameters. We kept the default values of the parameters as provided by the authors, except for the maximum number of matched tables (maxMatchedTables), which we increased from 100 to 5000 to allow MSJ to process more matching tables.

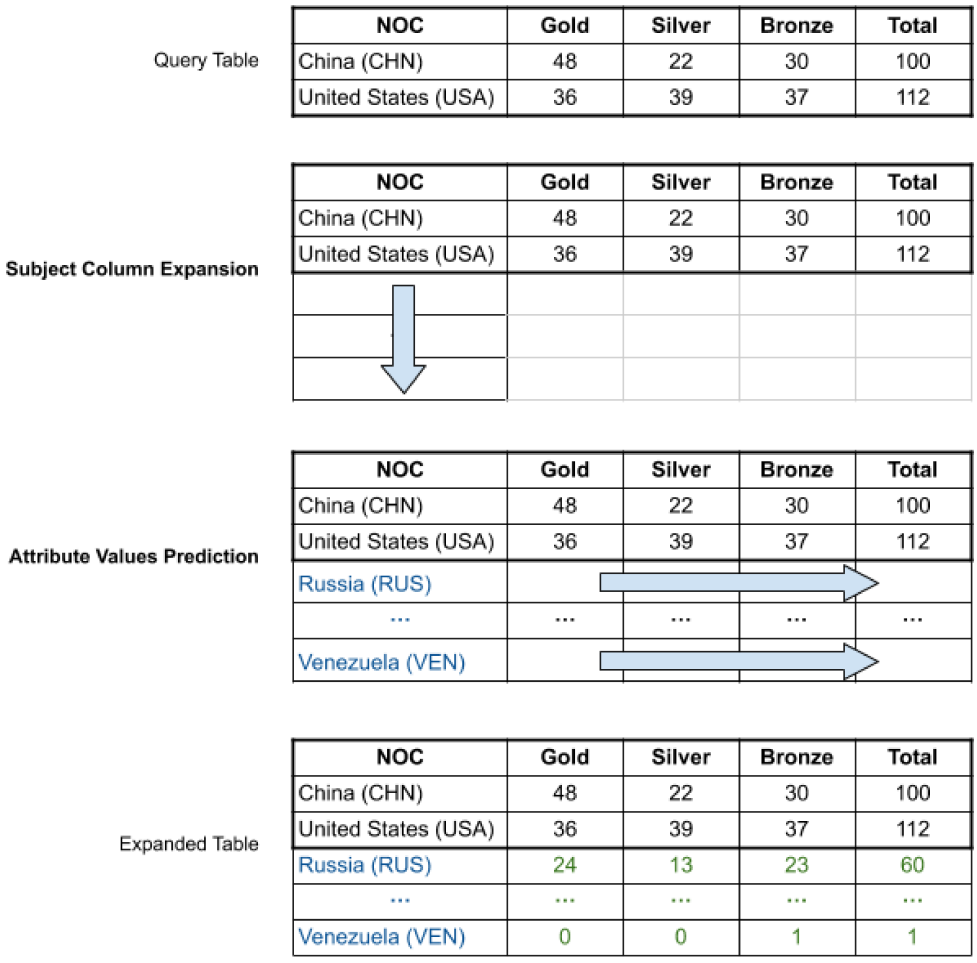

Figure 5.1: Subject column expansion and attribute value prediction.

5.2.4 Hardware and Software

The experiments were run on a laptop with Intel Core i7-7700HQ 4 Core (2.80 GHz), 32GB RAM, Samsung 860 EVO SSD, running Ubuntu 16.04 LTS. The T-REx framework was implemented in Scala (JVM), SEISA was implemented

in Scala, REA - in Java and Scala [10], and MSJ - in Java [11]. The dataset was indexed for each solution using Lucene Index [12] that allows searching for documents (web tables) based on terms.

5.3 Evaluation methods

The expanded tables are evaluated with respect to precision and recall. *Precision* (5.1) answers the question of what fraction of retrieved items are actually relevant (match ground truth). *Recall* (5.2) shows the fraction of the relevant items that are retrieved.

$$P = \frac{\#correctly\_retrieved\_items}{\#retrieved\_items} \tag{5.1}$$

$$R = \frac{\#correctly\_retrieved\_items}{\#ground\_truth\_items} \tag{5.2}$$

Additionally, we use *F-measure* (5.3), which is the weighted harmonic mean of precision and recall. The measure is helpful for evaluation as it combines the other two scores into a single one.

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} \tag{5.3}$$

where $\alpha = [0, 1]$.

For this evaluation, we use balanced F-measure, which equally weights precision and recall ($\alpha = 0.5$).

$$F_{\alpha=0.5} = \frac{2PR}{P+R} \tag{5.4}$$

The balanced F-measure does not emphasize either of the scores as we assume that for the user both are equally important.

[10] `https://github.com/JulianEberius/REA`
[11] `https://github.com/olehmberg/SearchJoin`
[12] `https://lucene.apache.org/`

5.4 Experimental Results

This section consists of two sub-sections. In the first subsection, we evaluate the subject column expansion performance of T-Rex and compare it to SEISA - showing the sensitivity of the methods with respect to the size of the query table. In the second subsection, we compare the performance of the value prediction for attributes between T-REx and EA methods (REA and Search Joins). Additionally, we evaluate the sensitivity of T-REx with respect to "head" vs "tail" query tuples.

5.4.1 Subject column expansion

To evaluate subject column expansion, we conduct multiple experiments to evaluate the sensitivity of T-REx and SEISA with respect to the number of examples in the query table and with respect to the relative frequency (head, mid, tail) of query key-value pairs in the table corpus.

Sensitivity with respect to the size of a query table.

We look at the precision, recall and balanced F-measure obtained by the methods when increasing the size of the query tables. We provide the set of keys (subject names) from the query tables as seeds for SEISA as it is an ESE method, and we use query tables with all attribute columns for T-REx.

First we will look at the balanced F-measure shown in Fig. 5.2. We see that T-REx has overall better results than SEISA on this score. The performance measure is decreasing with increasing number of examples in the query tables for both methods. The balanced F-measure combines precision and recall, we will look at these measures below.

Figure 5.3 reports Recall of the methods when increasing the size of the query tables. Both SEISA and T-REx, have a Recall value of over 0.86 and the performance measure increases with increasing seed size, reaching over 0.95 with 5 examples. Each query tuple might co-occur with tuples from the ground truth table, therefore, every additional example in the query table is likely to bring additional relevant tuples. However, we can see that the increase slows

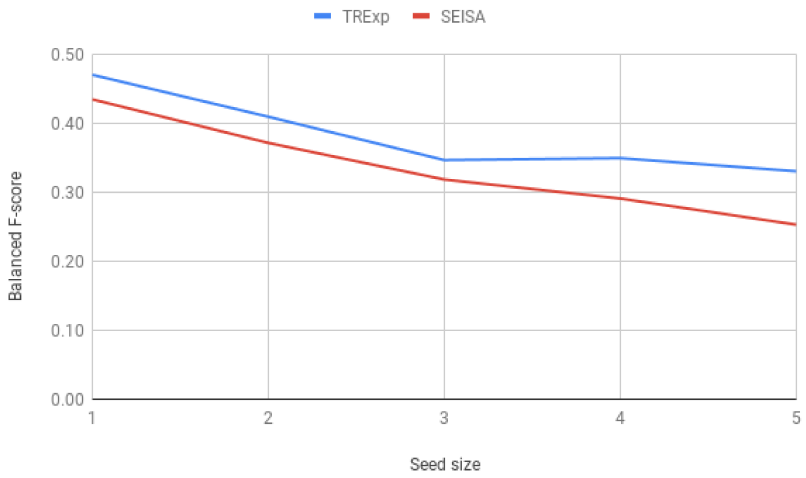

Figure 5.2: Sensitivity of the $F_{\alpha=0.5}$-score to the number of examples.

down as each additional example results in a smaller number of "new" relevant keys.

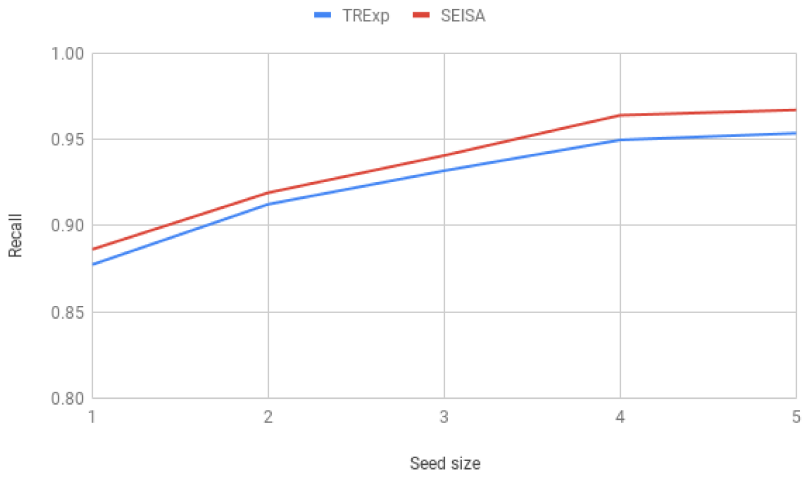

Figure 5.3: Sensitivity of the recall to the number of examples.

Overall, SEISA outperforms T-REx in terms of Recall, in average, by 1.3%. The difference comes from the fact that SEISA returns all entity names that co-occur with the keys, whereas T-REx also considers attribute values of each row in the query table, which leads to a stronger filter on the possible result. The benefit of this stronger filter becomes evident when we look at the Precision results reported in Fig. 5.4. The Precision of SEISA is lower than the Precision

of T-REx, in average, by 13% as SEISA returns more non-relevant keys than T-REx. The Precision of both methods drops with increasing size of the query table as the portion of non-relevant keys in the result is increasing. Since the ground truth tables were randomly picked and removed from the table corpus, it is possible that there is no other table in the corpus that contains as complete a set of entities belonging conceptually to the same set. Candidate keys will then be selected from tables that overlap with the query table but are likely irrelevant to the ground truth table. The larger the query table, the more likely this scenario is.

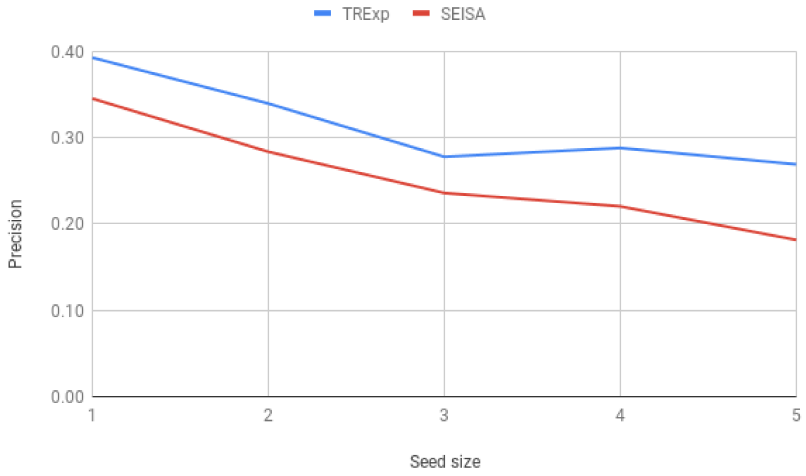

Figure 5.4: Sensitivity of the precision to the number of examples.

Sensitivity with respect to the frequency of query key-value pairs.

Next, we examine the sensitivity of the methods with respect to the frequency of query key-value pairs in the corpus. The balanced F-measure is shown in Fig. 5.5. For both methods, the measure is lower for head examples compared to tail examples. We will look at recall and precision of the methods below.

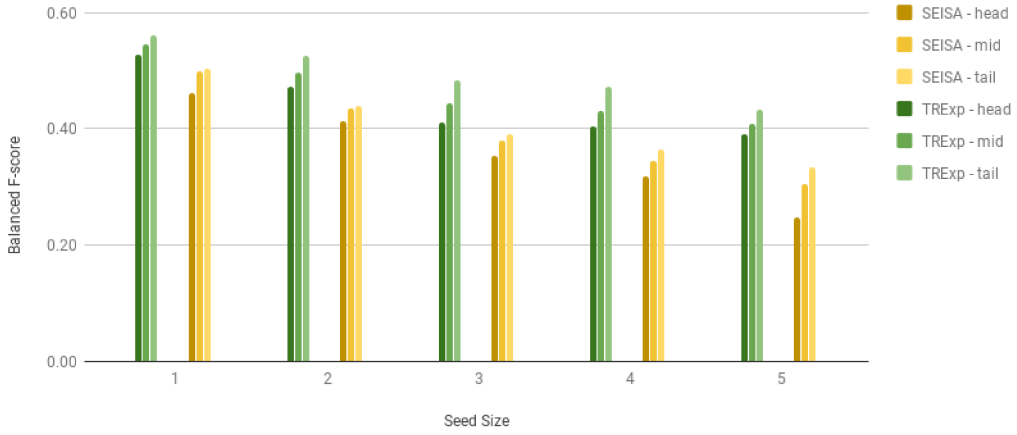

Figure 5.5: Sensitivity of the $F_{\alpha=0.5}$-score to the number of examples.

Recall is reported in Fig. 5.6. More frequent (head) examples increase recall for both methods, which is especially noticeable when seed size is equal to 1. The difference decreases when more examples are given. In case of head tuples, the more frequent key-value pairs are more likely to co-occur with many keys from the ground truth and each additional head tuple adds fewer "new" tuples as most co-occur with the previous head tuples. In case of tail tuples, the first tuple is likely to co-occur with a smaller number of ground truth tuples, and each following, more frequent, tuple is likely to bring more "new" tuples.

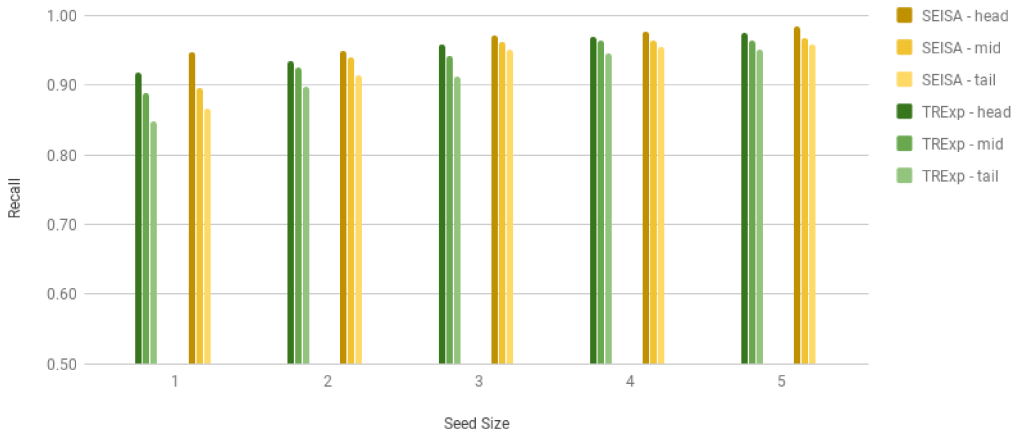

Figure 5.6: Sensitivity of the Recall to the number of examples.

The opposite is true for Precision as shown in Fig. 5.7. The more frequent key-value pairs are, the more likely they co-occur with non-relevant keys.

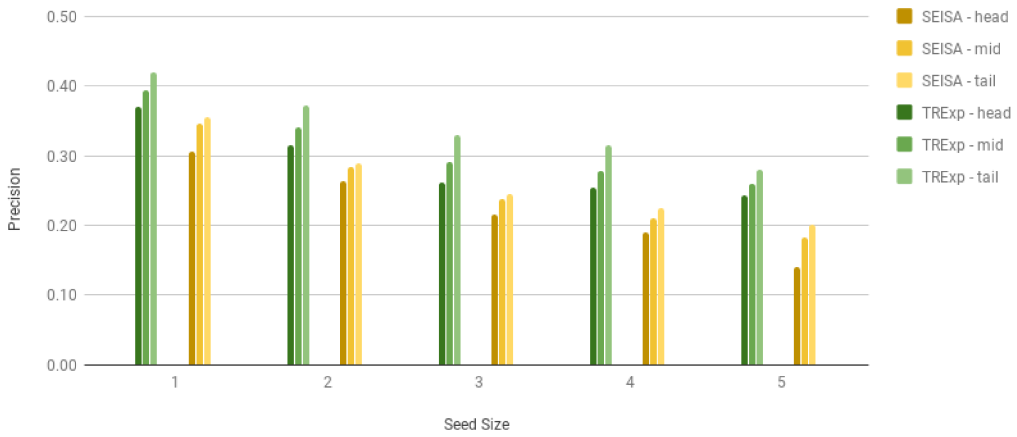

Figure 5.7: Sensitivity of the Precision to the number of examples.

In summary, for the task of the subject column expansion, T-Rex shows better performance results. Even though SEISA demonstrates slightly higher recall than T-REx, its precision is significantly lower resulting in a clear performance advantage of T-REx with respect to the balanced F-measure. As discussed above, T-Rex has a higher precision and lower recall as it filters candidate tables based on query key-value pairs, whereas SEISA only filters based on query keys.

5.4.2 Predicting attribute values

We evaluate attribute values prediction by comparing T-REx with REA and MSJ. The EA methods take a set of entity names and an attribute name (augmentation attribute) as input, whereas T-REx takes a query table.

For both REA and MSJ, we provided values from the subject column of a ground truth table as entity names, and we used column header labels as an augmentation attributes. For example, for the Gold medals count column of the 2008 Summer Olympic Games, we specify “Gold” as the augmentation attribute. In case the header label is empty or does not exist, the EA methods return empty result.

One problem we experienced with EA methods is their sensitivity to the provided augmentation attribute names. Web tables and context meta-information vary greatly in quality, completeness and consistency in the data set. Especially column header labels prove to be a major challenge as they are often "too generic" (this characteristic of web tables was described in Chapter 2). In multiple cases, we were not able to get any results from REA or MSJ due the their sensitivity to the provided attribute names. In comparison, our approach does not require attribute labels or any other context information.

As an input for T-REx, we provided a query table with a single example (tuple). Adding more examples would have been a significant advantage over EA methods that take only an attribute name to predict an attribute value.

Since REA works only with numeric columns and MSJ can work with any column data type, we first compare T-Rex with MSJ for all columns and then with REA for numeric columns.

Comparing T-REx with MSJ.

The results are presented in Table 5.4. MSJ has much lower recall, precision and F-score than T-REx mainly due to a smaller number of non-empty results compared to T-REx. MSJ returned results for 41 columns out of 151 possible ($\sim$ 27%) and T-REx - 129 columns out of 151 possible ($\sim$ 85%).

	T-REx	**MSJ**
Recall	0.63	0.19
Precision	0.66	0.21
F-Score	0.64	0.20

Table 5.4: Comparison of Precision, Recall and F-Score between T-REx and MSJ.

Comparing T-REx with REA.

The results are presented in Table 5.5. Similarly to MSJ, REA has lower recall, precision and F-score than T-REx mainly due to a smaller number of non-empty results in T-REx. REA returned results for 17 columns out of 35 numeric columns ($\sim$ 48%), T-REx - 27 columns out of 35 numeric columns ($\sim$ 77%).

	T-REx	REA
Recall	0.65	0.40
Precision	0.66	0.43
F-Score	0.65	0.42

Table 5.5: Comparison of Precision, Recall and F-Score between T-REx and REA.

In summary, for the task of predicting attribute values, as part of the problem of tables expansion, T-Rex demonstrates better performance than EA methods. The sensitivity of MSJ and REA to the quality and existence of a provided augmentation attribute significantly reduces their performance.

Sensitivity with respect to the frequency of query key-value pairs.

We also evaluate the sensitivity of T-REx with respect to the frequency of key-value pairs and the size of the query table. Fig. 5.8 reports Recall. Head tuples return more relevant tuples than mid and tail tuples. Increasing the number of examples in the query tables also increases the Recall performance of the method.

Figure 5.8: Sensitivity of the Recall to the number of examples.

Precision is illustrated in Fig. 5.9. Its behaviour similar to Recall.

Figure 5.9: Sensitivity of the Precision to the number of examples.

We also report balanced F-measure in Fig. 5.10, which is consequently also similar to precision and recall.

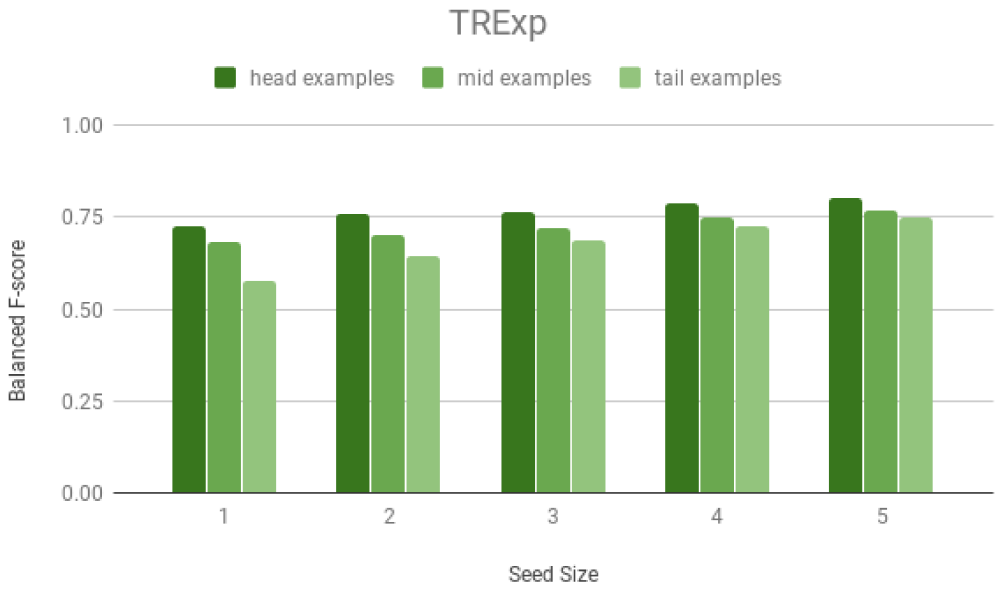

Figure 5.10: Sensitivity of the $F_{\alpha=0.5}$-score to the number of examples.

In summary, both, the number of examples in the query table as well

as frequency of the query key-value pairs, improve the Recall performance of T-Rex for subject column expansion and attribute value prediction tasks. However, the number of examples in the query table also results in lower precision performance of T-Rex for the task of subject column expansion.

Chapter 6

Conclusion and Future Work

In this chapter, we present some concluding remarks about our work and discuss a few possible future directions.

6.1 Conclusion

This thesis focuses on the problem of table expansion, where given a corpus of web tables, the task is to populate a query table by discovering other rows that may semantically belong to the table. The main challenge of using web tables as a data source is the lack of schema level information, which leads to various problems during schema matching. To mitigate this challenge, we have introduced an algorithm that does not consider schema-level information and relies only on instance-level information. It uses projections to split tables into entity-attribute binary relations - sets of key-value pairs, and then leverages co-occurrence to retrieve candidate key-value pairs that are then combined into candidate rows.

In an experimental evaluation, we compared the approach with an alternative solution to the problem of table expansion - an entity set expansion (ESE) method for subject column expansion and entity augmentation (EA) methods for attribute value prediction. We show that an alternative solution is not suitable for the task, especially for cases when a query table is from an arbitrary domain. The correctness of EA methods greatly depends on the quality of the web tables in the corpus and the accuracy of schema-level information.

6.2 Future Work

We have identified several directions for future work. One direction is enhancing schema matching of the current approach by utilizing various table pre-processing methods such as identifying attribute data types, unit conversions, and holistic matching methods [56]. The system can also be extended to support other scoring functions.

Another promising direction is related to broadening the application of the system, such as support for complex keys, multiple attribute values, filling missing values in a table and returning a ranked list of rows. Additionally, the system can be modified to return partially filled candidate records instead of predicting an attribute value for each query attribute of a candidate key. Moreover, the system can return multiple possible attribute values for a given query attribute of a candidate key with a confidence score assigned to each attribute value.

References

[1] A. Ahmadov, M. Thiele, J. Eberius, W. Lehner, and R. Wrembel, "Towards a hybrid imputation approach using web tables," in *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, IEEE, 2015, pp. 21–30. 1, 8

[2] S. Balakrishnan, A. Halevy, B. Harb, H. Lee, J. Madhavan, A. Rostamizadeh, W. Shen, K. Wilder, F. Wu, and C. Yu, "Applying webtables in practice," 2015. 1, 3, 15

[3] C. S. Bhagavatula, T. Noraset, and D. Downey, "Methods for exploring and mining tables on wikipedia," in *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, 2013, pp. 18–26. 16, 22

[4] ——, "Tabel: Entity linking in web tables," in *The Semantic Web - ISWC 2015*, M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, K. Thirunarayan, and S. Staab, Eds., Cham: Springer International Publishing, 2015, pp. 425–441, ISBN: 978-3-319-25007-6. 1, 12

[5] A. Bilke and F. Naumann, "Schema matching using duplicates," in *21st International Conference on Data Engineering (ICDE'05)*, IEEE, 2005, pp. 69–80. 18, 26

[6] L. Bing, W. Lam, and T.-L. Wong, "Wikipedia entity expansion and attribute extraction from the web using semi-supervised learning," in *Proceedings of the sixth ACM international conference on Web search and data mining*, ACM, 2013, pp. 567–576. 23

[7] E. Brynjolfsson, L. M. Hitt, and H. H. Kim, "Strength in numbers: How does data-driven decisionmaking affect firm performance?" *Available at SSRN 1819486*, 2011. 2

[8] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu, "Uncovering the relational web.," in *WebDB*, 2008. 8

[9] M. J. Cafarella, A. Halevy, and N. Khoussainova, "Data integration for the relational web," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 1090–1101, 2009. 1, 15, 16, 21

[10] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: Exploring the power of tables on the web," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 538–549, Aug. 2008, ISSN: 2150-8097. DOI: 10.14778/1453856.1453916. [Online]. Available: http://dx.doi.org/10.14778/1453856.1453916. 1, 7, 13, 16, 21

[11] H.-H. Chen, S.-C. Tsai, and J.-H. Tsai, "Mining tables from large scale html texts," in *Proceedings of the 18th conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, 2000, pp. 166–172. 15

[12] Z. Chen, M. Cafarella, and H. Jagadish, "Long-tail vocabulary dictionary extraction from the web," in *Proceedings of the Ninth ACM international conference on Web search and data mining*, 2016, pp. 625–634. 24

[13] E. Codd, "A relational submodel for large shared data banks," *Commun. Ass. Comput. Mach., June 1970*, 1970. 10

[14] B. B. Dalvi, W. W. Cohen, and J. Callan, "Websets: Extracting sets of entities from the web using unsupervised information extraction," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 243–252. 16

[15] N. Dalvi, A. Machanavajjhala, and B. Pang, "An analysis of structured data on the web," *arXiv preprint arXiv:1203.6406*, 2012. 12

[16] A. Das Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu, "Finding related tables," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12, Scottsdale, Arizona, USA: ACM, 2012, pp. 817–828, ISBN: 978-1-4503-1247-9. DOI: 10.1145/2213836.2213962. [Online]. Available: http://doi.acm.org/10.1145/2213836.2213962. 1, 12, 15–18

[17] L. Deng, S. Zhang, and K. Balog, "Table2vec: Neural word and entity embeddings for table population and retrieval," *arXiv preprint arXiv:1906.00041*, 2019. 22, 26

[18] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 601–610. 3

[19] J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, and W. Lehner, "Building the dresden web table corpus: A classification approach," in *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, IEEE, 2015, pp. 41–50. 49

[20] J. Eberius, M. Thiele, K. Braunschweig, and W. Lehner, "Top-k entity augmentation using consistent set covering," in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, ACM, 2015, p. 8. 13, 17–21, 26, 52

[21] T. M. Embley and W. Martin, "Eukaryotic evolution, changes and challenges," *Nature*, vol. 440, no. 7084, pp. 623–630, 2006. 13, 26

[22] N. A. S. Er, T. Abdessalem, and S. Bressan, "Set of t-uples expansion by example," in *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, ACM, 2016, pp. 221–230. 24

[23] B. Fetahu, A. Anand, and M. Koutraki, "Tablenet: An approach for determining fine-grained relations for wikipedia tables," in *The World Wide Web Conference*, 2019, pp. 2736–2742. 12, 17

[24] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from web tables," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 71–80. 1, 3

[25] Y. He and D. Xin, "Seisa: Set expansion by iterative similarity aggregation," in *Proceedings of the 20th international conference on World wide web*, ACM, 2011, pp. 427–436. 18, 19, 23, 24, 28, 52

[26] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data mining and knowledge discovery*, vol. 2, no. 1, pp. 9–37, 1998. 11

[27] K. Jayamalini and M. Ponnavaikko, "Deep data mining: An approach to convert semistructure data to structured data using improved tree matching," *Journal of Engineering and Applied Sciences*, vol. 13, no. 9, pp. 2722–2731, 2018. 2

[28] L. R. Lautert, M. M. Scheidt, and C. F. Dorneles, "Web table taxonomy and formalization," *ACM SIGMOD Record*, vol. 42, no. 3, pp. 28–33, 2013. 6, 9

[29] O. Lehmberg and C. Bizer, "Stitching web tables for improving matching quality," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1502–1513, 2017. 18, 27, 51

[30] O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer, "A large public corpus of web tables containing time and context metadata," in *Proceedings of the 25th International Conference Companion on World Wide Web*, International World Wide Web Conferences Steering Committee, 2016, pp. 75–76. 23, 51

[31] O. Lehmberg, D. Ritze, P. Ristoski, K. Eckert, H. Paulheim, and C. Bizer, "Extending tables with data from over a million websites," *Semantic Web Challenge 2014*, 2014. 1, 3, 15

[32] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer, "The mannheim search join engine," *Web Semant.*, vol. 35, no. P3, pp. 159–166, Dec. 2015, ISSN: 1570-8268. DOI: 10.1016/j.websem.2015.05.001. [Online]. Available: http://dx.doi.org/10.1016/j.websem.2015.05.001. 1, 12, 13, 15–19, 21–23,

[33] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava, "Truth finding on the deep web: Is the problem solved?" *arXiv preprint arXiv:1503.00303*, 2015. 12

[34] G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and searching web tables using entities, types and relationships," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1338–1347, 2010. 1, 16

[35] X. Ling, A. Y. Halevy, F. Wu, and C. Yu, "Synthesizing union tables from the web," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013. 18

[36] K. Morton, M. Balazinska, D. Grossman, and J. Mackinlay, "Support the data enthusiast: Challenges for next-generation data-analysis systems," *Proceedings of the VLDB Endowment*, vol. 7, no. 6, pp. 453–456, 2014. 4

[37] E. Muñoz, A. Hogan, and A. Mileo, "Using linked data to mine rdf from wikipedia's tables," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, ser. WSDM '14, New York, New York, USA: ACM, 2014, pp. 533–542, ISBN: 978-1-4503-2351-2. DOI: 10.1145/2556195.2556266. [Online]. Available: http://doi.acm.org/10.1145/2556195.2556266. 1, 15

[38] T. Nguyen, Q. Viet Hung Nguyen, M. Weidlich, and K. Aberer, "Result selection and summarization for web table search," *Proceedings - International Conference on Data Engineering*, vol. 2015, pp. 231–242, May 2015. DOI: 10.1109/ICDE.2015.7113287. 1, 3, 15

[39] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas, "Web-scale distributional similarity and entity set expansion," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, Association for Computational Linguistics, 2009, pp. 938–947. 23

[40] M. Paşca, "Weakly-supervised discovery of named entities using web search queries," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, ACM, 2007, pp. 683–690. 23

[41] G. Penn, J. Hu, H. Luo, and R. T. McDonald, "Flexible web document analysis for delivery to narrow-bandwidth devices," in *ICDAR*, 2001. 7

[42] R. Pimplikar and S. Sarawagi, "Answering table queries on the web using column keywords," *arXiv preprint arXiv:1207.0132*, 2012. 13, 16, 17, 26

[43] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *the VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001. 17, 26

[44] D. Ritze, O. Lehmberg, and C. Bizer, "Matching html tables to dbpedia," in *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, 2015, pp. 1–6. 1

[45] S. Sarabchi, "Extending Tables using a Web Table Corpus," Master's thesis, University of Alberta, Canada, 2020. 23

[46] Y. A. Sekhavat, F. Di Paolo, D. Barbosa, and P. Merialdo, "Knowledge base augmentation using tabular data.," in *LDOW*, 2014. 3

[47] J. Shen, Z. Wu, D. Lei, J. Shang, X. Ren, and J. Han, "Setexpan: Corpus-based set expansion via context feature selection and rank ensemble," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2017, pp. 288–304. 24, 28

[48] C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein, "Concept expansion using web tables," in *Proceedings of the 24th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2015, pp. 1198–1208. 1, 15, 18, 19, 23, 28

[49] H.-L. Wang, S.-H. Wu, I. Wang, C.-L. Sung, W.-L. Hsu, and W.-K. Shih, "Semantic search on internet tabular information extraction for answering queries," in *Proceedings of the ninth international conference on Information and knowledge management*, 2000, pp. 243–249. 15

[50] R. C. Wang and W. W. Cohen, "Language-independent set expansion of named entities using the web," in *Seventh IEEE international conference on data mining (ICDM 2007)*, IEEE, 2007, pp. 342–350. 23

[51] ——, "Character-level analysis of semi-structured documents for set expansion," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, Association for Computational Linguistics, 2009, pp. 1503–1512. 23

[52] WDC, *WDC Web Table Corpus 2015*, `http://webdatacommons.org/webtables/`, [Online; accessed 10-January-2019], 2015. 2, 10, 12, 13, 26, 49, 51

[53] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: Entity augmentation and attribute discovery by holistic matching with web tables," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12, Scottsdale, Arizona, USA: ACM, 2012, pp. 97–108, ISBN: 978-1-4503-1247-9. DOI: `10.1145/2213836.2213848`. [Online]. Available: `http://doi.acm.org/10.1145/2213836.2213848`. 1, 3, 9, 13, 15, 17–19, 21

[54] X. Yin, W. Tan, and C. Liu, "Facto: A fact lookup engine based on web tables," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 507–516. 1, 9, 16

[55] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition," *Document Analysis and Recognition*, vol. 7, no. 1, pp. 1–16, 2004. 6

[56] M. Zhang and K. Chakrabarti, "Infogather+: Semantic matching and annotation of numeric and time-varying attributes in web tables," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ACM, 2013, pp. 145–156. 1, 16, 17, 21, 22, 26, 32,

[57] S. Zhang and K. Balog, "Entitables: Smart assistance for entity-focused tables," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17, Shinjuku, Tokyo, Japan: ACM, 2017, pp. 255–264, ISBN: 978-1-4503-5022-8. DOI: 10.1145/3077136.3080796. [Online]. Available: http://doi.acm.org/10.1145/3077136.3080796. 1, 3, 15, 18, 19, 22, 26

[58] ——, "Ad hoc table retrieval using semantic similarity," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18, Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 1553–1562, ISBN: 978-1-4503-5639-8. DOI: 10.1145/3178876.3186067. [Online]. Available: https://doi.org/10.1145/3178876.3186067. 1, 15

[59] ——, "Web table extraction, retrieval, and augmentation: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 2, pp. 1–35, 2020. 18, 20