Web-Scale Web Table to Knowledge Base Matching

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Dipl.-Inf. Dominique Ritze
aus Heppenheim

Mannheim, 2017

Dekan:	Dr. Bernd Lübcke, Universität Mannheim
Referent:	Professor Dr. Christian Bizer, Universität Mannheim
Korreferent:	Professor Dr. Kai Eckert, Hochschule der Medien Stuttgart

Tag der mündlichen Prüfung: 06. November 2017

Abstract

Millions of relational HTML tables are found on the World Wide Web. In contrast to unstructured text, relational web tables provide a compact representation of entities described by attributes. The data within these tables covers a broad topical range. Web table data is used for question answering, augmentation of search results, and knowledge base completion. Until a few years ago, only search engines companies like Google and Microsoft owned large web crawls from which web tables are extracted. Thus, researches outside the companies have not been able to work with web tables.

In this thesis, the first publicly available web table corpus containing millions of web tables is introduced. The corpus enables interested researchers to experiment with web tables. A profile of the corpus is created to give insights to the characteristics and topics. Further, the potential of web tables for augmenting cross-domain knowledge bases is investigated. For the use case of knowledge base augmentation, it is necessary to understand the web table content. For this reason, web tables are matched to a knowledge base. The matching comprises three matching tasks: instance, property, and class matching. Existing web table to knowledge base matching systems either focus on a subset of these matching tasks or are evaluated using gold standards which also only cover a subset of the challenges that arise when matching web tables to knowledge bases.

This thesis systematically evaluates the utility of a wide range of different features for the web table to knowledge base matching task using a single gold standard. The results of the evaluation are used afterwards to design a holistic matching method which covers all matching tasks and outperforms state-of-the-art web table to knowledge base matching systems. In order to achieve these goals, we first propose the T2K Match algorithm which addresses all three matching tasks in an integrated fashion. In addition, we introduce the T2D gold standard which covers a wide variety of challenges. By evaluating T2K Match against the T2D gold standard, we identify that only considering the table content is insufficient. Hence, we include features of three categories: features found in the table, in the table context like the page title, and features that base on external resources like a synonym dictionary. We analyze the utility of the features for each matching task. The analysis shows that certain problems cannot be overcome by matching each table in isolation to the knowledge base. In addition, relying on the features is not enough for the property matching task. Based on these findings, we extend T2K Match into T2K Match++ which exploits indirect matches to web tables about the same topic and uses knowledge derived from the knowledge base. We show that T2K Match++ outperforms all state-of-the-art web table to knowledge base matching approaches on the T2D and Limaye gold standard. Most systems show good results on one matching task but T2K Match++ is the only system that achieves F-measure scores above 0.8 for all tasks. Compared to results of the best performing system TableMiner+, the F-measure for the difficult property matching task is increased by 0.08, for the class and instance matching task by 0.05 and 0.03, respectively.

Zusammenfassung

Millionen relationaler HTML Tabellen können im World Wide Web gefunden werden. Im Gegensatz zu unstrukturiertem Text, bieten relationale Webtabellen eine kompakte Repräsentation von Entitiäten, die durch Attribute beschrieben sind. Die Tabellendaten umfassen ein breites thematisches Specktrum. Daten aus Webtabellen werden zum Beantworten von Fragen, zum Anreichern von Suchergebnissen und zum Komplementieren von Wissensbasen benutzt. Bis vor ein paar Jahren besaßen nur Suchmaschinenfirmen wie Google und Microsoft Webcrawldatensätze von denen Webtabellen extrahiert werden. Forscher außerhalb dieser Firmen konnten deshalb nicht mit Webtabellen arbeiten.
In dieser Arbeit wird der erste öffentlich verfügbare Webtabellenkorpus mit Millionen von Webtabellen vorgestellt. Der Korpus ermöglicht es Forschern mit Webtabellen zu experimentieren. Ein Profil des Korpus gibt Einblicke in die Charakteristika und Thematiken. Zusätzlich wird das Potential von Webtabellen für das Vervollständigen von domänenübergreifenden Wissensbasen untersucht. Für diesen Anwendungsfall ist es nötig, den Inhalt der Tabelle zu verstehen. Dafür werden Webtabellen mit einer Wissensbasis abgeglichen. Der Abgleich umfasst drei Aufgaben: das Abgleichen der Instanzen, Relationen und Klassen. Existierende Systeme fokussieren sich auf eine Teilmenge der Aufgaben oder werden auf Goldstandards evaluiert, die nur eine Teilmenge der Herausforderungen enthalten.
Diese Dissertation evaluiert systhematisch den Nutzen eines breiten Spektrums an Merkmalen für den Abgleich von Webtabellen und Wissensbasen anhand eines Goldstandards. Die Ergebnisse der Evaluation werden anschließend genutzt, um eine holistische Abgleichsmethode zu entwerfen, die alle drei Aufgaben abdeckt, und bestehende Methoden übertrifft. Um diese Ziele zu erreichen führen wir erst den T2K Match Algorithmus ein, der alle drei Aufgaben integriert angeht. Zudem stellen wir den T2D Goldstandard vor, der ein breites Spektrum an Herausforderungen abdeckt. Beim Evaluieren von T2K Match anhand T2D stellen wir fest, dass es nicht ausreicht den Tabelleninhalt zu betrachten. Deswegen fügen wir weitere Merkmale aus drei Kategorien hinzu: Merkmale, die in den Tabellen gefunden werden, die aus dem Kontext stammen und Merkmale aus externen Ressourcen. Wir analysieren den Nutzen der Merkmale für jede Aufgabe und zeigen, dass gewisse Probleme nicht überwunden werden können, wenn der Abgleich jeder Tabelle mit der Wissensbasis einzeln stattfindet. Außerdem reichen die Merkmale nicht zum Abgleich der Verknüpfungen aus. Basierend auf diesen Erkenntnissen erweitern wir T2K Match zu T2K Match++, der indirekte Abgleiche zu anderen Webtabellen über dasselbse Thema ausnutzt und Wissen verwendet, das aus der Wissensbasis abgeleitet wird. Wir zeigen, dass T2K Match++ auf dem T2D und Limaye Goldstandard alle bestehenden Abgleichsysteme übertrifft. Die meisten Systeme zeigen gute Ergebnisse für eine der Aufgaben aber nur T2K Match++ erreicht F-measure Werte von über 0.8 für alle Aufgaben. Verglichen mit den Ergebnissen des besten System TableMiner+, kann der F-Measure Wert des schweren Relationsabgleichs um 0.08, der des Klassen- und Instanzabgleichs um 0.05 bzw. 0.03 erhöht werden.

Contents

Chapter 1

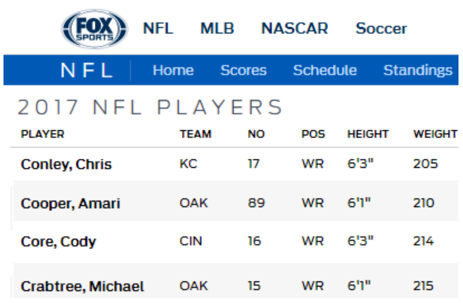
Introduction

With its massive amount of information, the World Wide Web presents the most extensive collection of knowledge that has ever been accessible to humans. “Never before has so much information from such a wide variety of sources and in so many formats been available to the public” [Hartman and Ackermann, 2010]. One reason is the availability of large encyclopedias like Wikipedia providing enormous compilations of information. Another reason lies in the decentralized nature of the Web allowing anyone to contribute websites. Since the information are mainly designed for human readers, the automatic extraction of high quality structured knowledge from the Web is challenging. The main reasons are the composition of the information in natural language and the usage of arbitrary data structures.

To overcome those shortcomings, the vision of the *Semantic Web* as described by [Berners-Lee et al., 2001] is to give information a well-defined meaning and to enable computers and humans to work in cooperation. Thus, the data can be automatically processed without the need to handle the ambiguity of language. To create a Web of data, which is one part of the semantic web vision, *Linked (Open) Data* (LOD) constitutes a paradigm of publishing and interlinking datasets on the Web [Bizer et al., 2009a]. These datasets contain well-defined information that on the one hand are prepared for human readers and on the other hand can be automatically understood by machines. Among the datasets, cross-domain knowledge bases like DBpedia or the Google Knowledge Graph can be found. They include millions of facts describing things like persons, works or places. Compared to the overall number of websites and the variety of existing topics, the amount of information in LOD datasets is limited.

Web Tables are a source type of web content that provides large amounts of data with a broad topical coverage. At the same time, web tables use a fixed structure to present their content [Cafarella et al., 2008a, Crestan and Pantel, 2011, Hassanzadeh et al., 2015, Ritze and Bizer, 2017]. Web tables provide a compact representation of entities described by attributes whereby the structure reflects logical relations. Compared to unstructured data, the effort to extract and interpret the data is reduced. For this reason, web tables have gained increasing attention by the research community. A number of use cases emerged that benefit from this rich resource of structured data. The set of use cases includes question answering [Balakrishnan et al., 2015] and knowledge base augmentation [Dong et al., 2014].

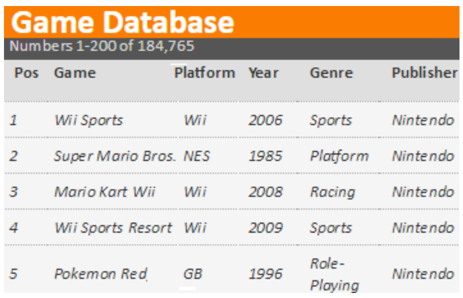
Figure 1.1, 1.2, and 1.3 depict web tables about different topics: NFL players, videogames, and countries. They all share the same structure and provide a compact representation of the domain. Each row describes one entity, e.g., NFL player, videogame, or country, using attributes like the team, the publisher, or the capital.



2017 NFL PLAYERS

PLAYER	TEAM	NO	POS	HEIGHT	WEIGHT
Conley, Chris	KC	17	WR	6'3"	205
Cooper, Amari	OAK	89	WR	6'1"	210
Core, Cody	CIN	16	WR	6'3"	214
Crabtree, Michael	OAK	15	WR	6'1"	215

Figure 1.1: Example web table table about NFL players.[1]

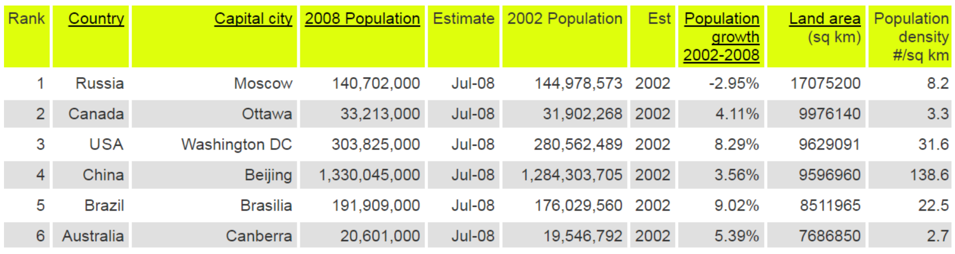


Game Database

Numbers 1-200 of 184,765

Pos	Game	Platform	Year	Genre	Publisher
1	Wii Sports	Wii	2006	Sports	Nintendo
2	Super Mario Bros.	NES	1985	Platform	Nintendo
3	Mario Kart Wii	Wii	2008	Racing	Nintendo
4	Wii Sports Resort	Wii	2009	Sports	Nintendo
5	Pokemon Red	GB	1996	Role-Playing	Nintendo

Figure 1.2: Example web table about videogames.[2]



Rank	Country	Capital city	2008 Population	Estimate	2002 Population	Est	Population growth 2002-2008	Land area (sq km)	Population density #/sq km
1	Russia	Moscow	140,702,000	Jul-08	144,978,573	2002	-2.95%	17075200	8.2
2	Canada	Ottawa	33,213,000	Jul-08	31,902,268	2002	4.11%	9976140	3.3
3	USA	Washington DC	303,825,000	Jul-08	280,562,489	2002	8.29%	9629091	31.6
4	China	Beijing	1,330,045,000	Jul-08	1,284,303,705	2002	3.56%	9596960	138.6
5	Brazil	Brasilia	191,909,000	Jul-08	176,029,560	2002	9.02%	8511965	22.5
6	Australia	Canberra	20,601,000	Jul-08	19,546,792	2002	5.39%	7686850	2.7

Figure 1.3: Example web table about countries.[3]

For humans, understanding the web table content is often an easy task, as they are made for human consumption. For machines, web tables present a two-dimensional grid including characters and numbers without any meaning. As indicated in Figure 1.4, the cell with the character sequence "USA" can refer to real-world objects like a city in Japan, a music album, or a country.

To make use of the table data considering the scale of the Web, there is a need for an automated table understanding. Traditional information extraction methods which have been applied for other data sources like unstructured text are not well

[1]Source: `http://www.foxsports.com/nfl/players`
[2]Source: `http://www.vgchartz.com/gamedb/`
[3]Source: `http://data.mongabay.com/igapo/world_statistics_by_area.htm`

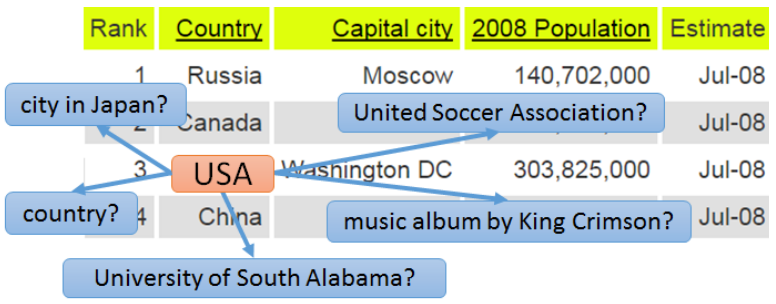


Figure 1.4: Real-world objects that are referred by the character sequence “USA” that is given in the example web table.

suited for web tables. These methods are not aware that the structure reflects the semantics of the content [Limaye et al., 2010]. Table understanding covers the overall process starting with the extraction of a table from web pages to its semantic interpretation [Hurst, 2001]. The goal of the table understanding process is to recover the semantics of the table. This includes the identification of the table topic, of the attributes meaning and of the real-world objects that are referred by the entities. Concerning the example in Figure 1.3, it should be determined that the table is about countries, the attribute “Capital city” states the capital of a country and the entity with the name “USA” “United States of America”.

One way to recover the semantics is to match the web tables to a knowledge base. A knowledge base is a combination of an ontology and instances of the classes in the ontology [Staab and Studer, 2009]. Further, properties are used to describe instances. Similarly, we find the same elements in web tables: the table itself, the attributes represented in columns, and the entities represented in rows. Accordingly, during the matching of web tables of a knowledge base, three matching tasks need to be performed: matching the tables to knowledge base classes, the attributes to knowledge base properties and table entities to knowledge base instances. As matching result of each task, correspondences between the table elements and the knowledge base elements are generated. The correspondences indicate which table elements fit to which knowledge base elements. Since knowledge bases follow the linked data principles, they provide a well-defined meaning of the data which is transferred to the corresponding web table elements. Figure 1.5 shows all correspondences between a web table and a knowledge base. First, the web table and the class *country* describe the same domain. Second, the attribute “Capital city” corresponds to the property “Capital” in the knowledge base. Third, three correspondences are found on the instance level. Hence, we know that the web table entity “USA” does not refer the city in Japan but the country.

To generate the correspondences, each matching task performs a comparison between the web table and knowledge base elements by applying matchers. These matchers compare a pair of features using similarity measures. An exemplary in-

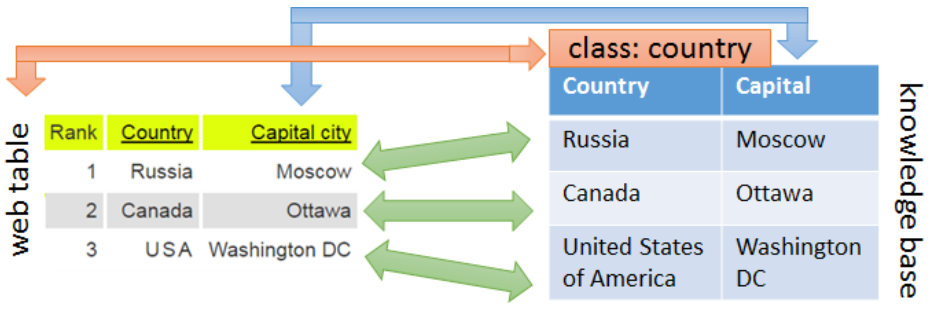


Figure 1.5: Correspondences between a web table and a knowledge base.

stance matcher takes an entity label and an instance label as feature and applies a string similarity measure. The resulting similarity provides an estimate how similar the labels and in turn the entity and the instance are. For example, the entity label "Russia" and the instance label "Russia" will have a very high similarity. In contrast, the labels "USA" and "United States of America" are not very similar, although the same real-world object is mentioned. In consequence, only comparing the labels might not be enough. This is the reason why usually a variety of matchers is considered for each task. Since each of matcher computes a similarity for a pair of features, the similarities are aggregated to get an estimation if a table element corresponds to a knowledge base element.

The main challenges of matching web tables lie in their characteristics. As stated by [Hernández and Stolfo, 1998], web data is considered as "dirty" based on the large number of contributors and the diversified knowledge of the standard. This leads to different kinds of heterogeneities: different namings of entities and attributes including abbreviations and different choices in the design of the table. In addition, there is no centralized quality control such that significant variations in the quality of the table data occur. Other challenging characteristics are the size of the tables and the amount of discriminative content [Cafarella et al., 2008b]. If a table only contains a small set of entities described by few attributes, it is difficult to generated the correct correspondences. Another challenge is posed by the large amount of web tables which make them a valuable source. On the one hand, taking as much web tables as possible into account enables to profit from the broad topical coverage. On the other hand, the missing quality control and the sizes of the web tables can lead to insufficient matching results. Hence, use cases like knowledge base augmentation benefit most from large amounts of matched web tables so that matching errors for particular tables can be compensated [Ritze et al., 2016]. As consequence, web table to knowledge base matching approaches need to handle millions of web tables [Cafarella et al., 2008b, Lehmberg et al., 2016].

The goal of this thesis is to systematically evaluate the utility of a wide range of different features for the web table to knowledge base matching task using a single gold standard. The results of the evaluation are used to design a holistic matching method which covers all three matching tasks. To get there, we first propose the T2K Match algorithm which addresses all three matching tasks in an integrated fashion. In addition, we introduce the T2D gold standard covering a wide variety of challenges. By evaluating T2K Match against the T2D gold standard, we identify that only considering the table content is insufficient due to the presented challenges: the heterogeneities, the size of the tables and the amount of discriminative content. Thus, we include features found in the table, in the table context, and features that base on external resources like a catalog with alternative names. We analyze the utility of the features for each matching task. Still, not all problems can be be overcome by matching each table in isolation to the knowledge base. Additionally, relying on the features is especially not enough for the property matching task. Based on these findings, we extend T2K Match into T2K Match++ which exploits indirect matches to web tables about the same topic and uses knowledge derived from the knowledge base.

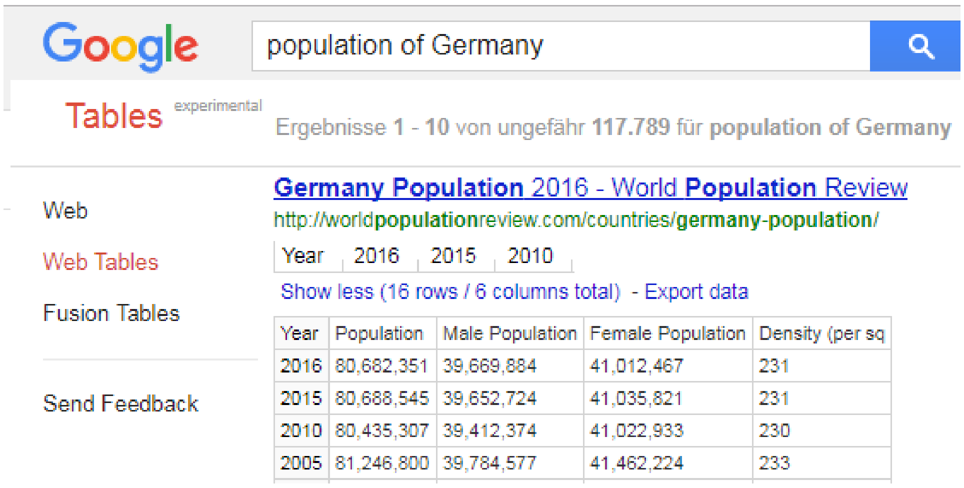
Beside the matching, we introduce the first publicly available web table corpus containing millions of web tables. Until a few years, only big search engine companies owned large web crawls from which web tables are extracted. Hence, it has not been possible for researchers outside the companies to experiment with web tables. Further, we create profile of the corpus to give insights to web table characteristics and topics. Only the availability of the corpus together the profile enabled us to generate the T2D gold standard which presents on of the basics of the evaluation.

1.1 Motivation

With its large amounts and diverse topical coverage, web tables provide an interesting source of information. In contrast to unstructured data, web tables have several beneficial characteristics like the fixed structure that reflects logical relations. Thus, gathering information from web tables has a plethora of use cases like table search, table extension, or knowledge base augmentation that will be presented later in this section. So far, the main consumers of web tables are companies like Google [Cafarella et al., 2008b] or Microsoft [Yakout et al., 2012] since they are in possession of web crawls from which web tables can be extracted. Unfortunately, they do not give public access to their data. This is not surprising as the data is part of their business value. Thus, on the one hand, large corpora of web tables are not available to other researchers and on the other hand the topical coverage of web tables is not known, such that the benefit for certain use cases cannot be estimated.

As another result of the unavailability of public web table corpora, only a few systems matching web tables to knowledge bases have been applied on web-scale datasets. Besides the data, also the systems itself are not available such that the reported matching results are not reproducible. These circumstances support the low quality of the data extracted from web tables [Yin et al., 2011]. To be able to improve the performance, it is necessary to have the possibility to evaluate and compare the matching methods. However, before this thesis, there has been no publicly available gold standard with tables from more than one website that captures all the matching tasks and thus provide a basis for the evaluation of strategies. Since the table understanding which includes the matching presents the foundation of the following use cases, these use cases can benefit from an increased matching quality.

Table Search Search engines are focused on searching documents in the Web. The relevant documents that are retrieved need to be processed by the user to satisfy the information need. For fact seeking queries like "population of a country", the according information can often be detected in web tables in a comprised format. Searching for tables containing relevant information given a keyword query is performed by [Cafarella et al., 2009], [Venetis et al., 2011], [Yin et al., 2011] and [Pimplikar and Sarawagi, 2012]. Google uses web tables to answer queries by displaying a tabular snippet as search result [Balakrishnan et al., 2015]. An example of Google Tables, a search engine for web tables, is presented in Figure 1.6. Hence, the information relevant for the user is directly displayed without the need of considering any further document.



population of Germany

Ergebnisse **1** - **10** von ungefähr **117.789** für **population of Germany**

Germany Population 2016 - World **Population** Review
http://world**population**review.com/countries/**germany-population**/

Year	2016	2015	2010

Show less (16 rows / 6 columns total) - Export data

Year	Population	Male Population	Female Population	Density (per sq
2016	80,682,351	39,669,884	41,012,467	231
2015	80,688,545	39,652,724	41,035,821	231
2010	80,435,307	39,412,374	41,022,933	230
2005	81,246,800	39,784,577	41,462,224	233

Figure 1.6: Search for the population of the country Germany in Google Tables.[4]

Table Extension For gathering information about a set of entities, e.g., the population of all countries in the world, an individual search for each entity is nec-

[4]Source: `https://research.google.com/tables`

essary. To avoid such labor-intensive tasks, table extension aims for automatically finding the values of attributes of one or more entities given in a local table [Yakout et al., 2012, Bizer, 2014, Lehmberg et al., 2015]. For example, a local table covers the name of all countries and the task is to find all values for the attribute population. To receive the values, table extension methods search for the mentioned information in a corpus, for example in a web table corpus. Approaches for table extension have been introduced by [Cafarella et al., 2009], [Yakout et al., 2012], [Das Sarma et al., 2012], as well as by [Lehmberg et al., 2015].

Information included in tables cannot only be used to satisfy information needs but can also help to create or improve data sources like knowledge bases. Over the last years, there has been a growing trend to create, share, and use knowledge bases for a better understanding of the data [Abadi et al., 2014]. Knowledge bases are already used as background knowledge in an increasing range of applications like web search, natural language understanding, data integration, and data mining [Lehmann et al., 2015]. For example, Google's Knowledge Graph is used to improve Google's search results [Singhal, 2012]. Both - the creation and the augmentation can be facilitated using web table data.

Knowledge Base Augmentation The completeness and soundness of a knowledge base is fundamental for all application relying on the knowledge base. To increase the completness and soudness, an automated augmentation of large knowledge bases is desirable [Dong et al., 2014]. To complete a knowledge base, either missing values can be detected and filled (slot filling) or new entities and attributes can be added (entity/attribute expansion). For both augmentation types, web tables have been considered as data source [Wang et al., 2012, Sekhavat et al., 2014, Ritze et al., 2016]. Among other data found in the Web like text documents or semantic annotations, web tables have been used to generate *Google's Knowledge Vault*, a cross-domain knowledge base that is 38 times bigger than existing automatically constructed knowledge bases [Dong et al., 2014].

1.2 Contributions

In this thesis, we provide the following contributions:

1. We offer the first publicly available web tables corpus containing 150 million relational web tables originating from millions of different data providers. Corpora of similar sizes have been generated for example by [Cafarella et al., 2008a] but they are not available to the public.
2. We profile the introduced web table corpus to derive its characteristics and give insights into its topical contents. Although web tables have been utilized in a variety of use cases, it remains unclear for which use cases web tables are most valuable. As one example, we demonstrate the potential of web tables for augmenting a knowledge base.
3. We provide the publicly available gold standard T2D which enables the evaluability of matching web tables to knowledge base approaches. In contrast to other gold standards, T2D is publicly available and covers tables from more than one data provider. It contains correspondences for all three matching tasks. Further, tables which do not overlap with the knowledge base are included to test if a matching algorithm is feasible to distinguish between overlapping and non-overlapping tables. Altogether, T2D covers a wide range of challenges and reflects the matching task more realistically.
4. We analyze the utility of features that are used in state-of-the-art web table matching systems. The set of features includes the web table context but also external resources like a surface form catalog providing alternative names. In previous works, only a subset of the tasks and a subset of the features is considered, therefore a conclusion of the usefulness of individual features cannot be drawn. We integrate all features into a single system, T2K Match, and show the utility for each task on the T2D gold standard. Since both, the system and the data is publicly available, we enable the reproducibility and transparency of the results.
5. We propose the holistic matching method T2K Match++ which solves all matcahing tasks by exploiting information gathered from web tables with similar topics. To detect similar tables, the tables are matched among each other. Existing correspondences between similar tables and the knowledge base are propagated to either detect new correspondences or to review whether correspondences should hold. Holistic matching has shown to improve the results when matching other data sources like ontologies. Until now, it has not been applied in this form in previous works to match web tables to knowledge bases. As our experiments show, the holistic framework outperforms all state-of-the-art web table to knowledge base matching systems on both, the T2D gold standard providing a wide variety of challenges and on the commonly used Limaye gold standard.

1.3 Outline

In this section, the main topic of each chapter is summarized.

Chapter 2: Data Integration Process. Having introduced and motivated the goal of the thesis, this chapter presents the foundations of data integration and matching in particular. Besides general data integration tasks and challenges that need to be addressed, we describe the matching process in which four steps are performed: process the input, compute similarities by applying different matchers, aggregate the matcher results and decide about the correspondences. Further, both matching tasks that are part of the data integration process are presented: schema and data matching.

Chapter 3: Knowledge Bases. Within this chapter, the concept of knowledge bases is described. A knowledge base is a combination of an ontology and instances of the classes in the ontology. Further, properties are defined to express relations between instances. We introduce and compare the most commonly deployed knowledge bases. For our experiments, we use the knowledge base DBpedia. Thus, a detailed overview of DBpedia about its size, the class hierarchy, and the covered topics is given.

Chapter 4: Web Tables. This chapter introduces the foundations of web tables. To create a corpus of web tables that can be used as input for the matching tasks, the web tables need to undergo an extraction process. Further, we discuss which types of web tables exist and how they are distributed among the Web.

Table Extraction Process: At first, the web tables need to be detected on a web page and their content needs to be extracted. To distinguish between different table types, i.e. to identify tables that are only used for layout purposes, different classification approaches have been introduced. A relational table is considered as a compact representation of a topic that is presented by entities which are described by attributes. To identify how the entities and attributes are named, the entity labels and the attribute headers are determined during the metadata recovery. Besides the overview of the extraction process, the methods that have been applied to create the WDC Web Table Corpora are described.

Table Corpora: The resulting WDC Web Table Corpus 2012 is the first publicly available web table corpus, comprising 150 million relational web tables from millions of data providers. The corpus is analyzed regarding different characteristics like the size of tables and it is compared to the statistics of other corpora.

Chapter 5: Profiling. The variety of use cases indicates web tables as a worthwhile data source. Since the existing web table corpora are not available to the public, it is not clear which topics are covered by web tables. In this chapter, an in-depth profiling of the publicly available WDC Web Table Corpus 2012 is provided, giving insights into its topical contents. The profiling is performed by matching the corpus of web tables to the knowledge base DBpedia. A profile is especially valuable to determine for which use cases web tables are most promising. For example, if a company likes to extend their knowledge about cars but cars are rarely mentioned in the web tables, using another data source is more promising. As one particular use case, we show the potential of web tables for augmenting DBpedia.

Chapter 6: Web Table to Knowledge Base Matching. In this chapter, the foundations of matching web tables to knowledge bases are presented. Further, the publicly available gold standard T2D is introduced together with the T2K Match web table matching framework addressing all required tasks.

Matching of Web tables: The task of matching web tables to knowledge bases poses challenges that are specific for the web tables as data source, e.g., the size of the tables or the lack of a formal schema. Further, since neither information about the schema nor about the data is available, both matching types need to be performed. Hence, three matching subtasks can be defined: instance, property, and class matching. Existing matching systems either consider only a subset of the tasks or restrict the tasks.

T2D Gold Standard: The evaluation of existing web table to knowledge base matching systems is performed on gold standards that are either not publicly available, focus on a specific topic and/or only cover tables from one data provider like Wikipedia. To overcome the limitations of previously used gold standards, the T2D gold standard is presented which covers manually generated correspondences between web tables and DBpedia for all three matching tasks.

Web Table Matching Algorithm T2K Match: We introduce a matching system called T2K Match that addresses the three matching tasks at the same time while the tasks mutually influencing each other. We evaluate the performance of the system by applying it on the T2D gold standard. The results indicate the difficulties of matching web tables to a knowledge base: the lack of information that is available for each table and different kinds of heterogeneities like various naming. A comparison to existing systems shows that the findings are in line with our results and similar performances are achieved within a restricted scenario.

Chapter 7: Feature Utility Analysis. During the matching process, features serve as input for the matchers. The matchers compute similarities by applying a similarity measure on these features. This chapter gives an overview of features that are proposed in literature and analyzes their utility for all three matching tasks.

Feature Review: A large number of features has been suggested in literature for the matching of web tables. While some of the features rely on external resources, e.g., a surface form catalog covering alternative names, others represent for example the context of a table. State-of-the-art systems only implement a limited set of features and the various systems are evaluated on different gold standards. Thus, it is difficult to decide how important a certain feature is and how much influence it has on the overall performance.

Feature Utility Study: A feature utility study is provided in which each feature is analyzed. All features are included in T2K Match. Taking a set of features into account requires an aggregation strategy that can properly combine the generated similarities. The aggregation strategy is extended such that it adapts itself for each table depending on the distinctiveness of a feature. Different combinations of features are evaluated on the T2D gold standard to identify the essential features for each matching task. Further, an analysis is given to show how much performance gain is possible by exploiting the features.

Chapter 8: The T2K Match++ Method. This chapter starts with highlighting the limitations of the previously described matching methods. None of the existing systems tackles the matching of web tables to a knowledge base in a way such that similarities of similar tables to the knowledge base are taken into account. Thus, the holistic matching framework T2K Match++ that simultaneously matches web tables to a knowledge base by exploiting that web tables can share the same topics is presented. Therefore, the functionality of the T2K Match algorithm is extended with a component that includes indirect mappings. For each table, similar tables are searched and matched among each other in an efficient way. Using both, the correspondences between the tables and the correspondences to the knowledge base, more evidence about the correctness of the correspondences can be gained. Since the additional indirection can lead to errors, the classification step is extended and takes domain knowledge into account. The chapter presents the evaluation of T2K Match++ on the T2D gold standard which results in performance increases for all three tasks, compared to T2K Match. Further, another evaluation on a second gold standard (Limaye gold standard [Limaye et al., 2010]) demonstrates the applicability of the algorithm on unknown datasets. Additionally, the results show that the proposed approach outperforms the existing state-of-the-art web table to knowledge base matching systems introduced by [Limaye et al., 2010], [Zhang, 2016], [Mulwad et al., 2013] and [Venetis et al., 2011].

Chapter 9: Conclusion. The final chapter summarizes the core contributions of this thesis, discusses limitations and directions for future work as well as presents the research impact of this work.

1.4 Published Work

Parts of the work presented in this thesis have been published previously:

- **The extraction of the WDC Web Table Corpus from the Common Crawl:**

  Oliver Lehmberg, Dominique Ritze, Robert Meusel, Christian Bizer: A Large Public Corpus of Web Tables containing Time and Context Metadata. In Proceedings of the 25th International World Wide Web Conference (WWW), 2016.

- **The profiling of the WDC Web Table Corpus:**

  Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, Christian Bizer: Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In Proceedings of the 25th International World Wide Web Conference (WWW), 2016.

- **T2K web table matching:**

  Dominique Ritze, Oliver Lehmberg, Christian Bizer: Matching HTML Tables to DBpedia. In Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics (WIMS), 2015.

- **The feature utility study:**

  Dominique Ritze & Christian Bizer: Matching Web Tables To DBpedia - A Feature Utility Study. In Proceedings of the 20th Extended Database Technology Conference (EDBT), 2017.

- **The T2K Match++ method:**

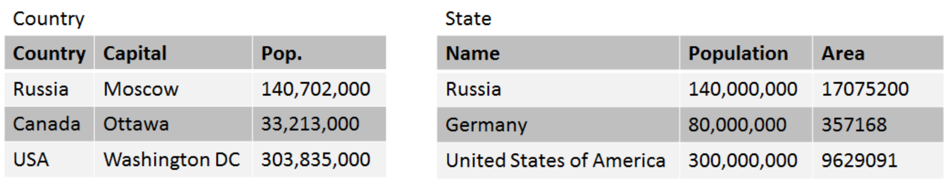
  Dominique Ritze & Christian Bizer: T2K Match++: Restricting Holistic Matching with Domain Knowledge. Paper submitted to the 11th ACM International Conference on Web Search and Data Mining (WSDM), 2018.

Chapter 2

The Data Integration Process

The increasing amount of data that is generated every day enables a variety of applications which require an efficient and accurate data integration. The goal of data integration is to provide a unified access to data residing in multiple, autonomous data sources [Dong and Srivastava, 2015]. Data integration is crucial for many use cases, including application areas like business, science, government, or the Web. Already in 2005, enterprise information integration has been estimated to yield about half a billion revenue [Halevy et al., 2005]. For a business, information integration is required in a variety of sectors, including customer relation management, supply chain management, business intelligence, or when companies are merged. In science, especially within the biology domain, a large variety of databases exist, i.e., about 180 databases in 2016 [Rigden et al., 2016]. Integrating data from various databases enables the researcher to collect all information that are for example necessary to perform experiments. Beside business and science, applications that require data integration can also be found in the public sector, e.g., to identify suspects by combining information about calls, online profiles, and credit card transactions [Aggrawal, 2015]. Lastly, based on the plethora of available data source, the Web itself presents a large application area. For example, data integration is required for the search in the World Wide Web [Halevy et al., 2006] or to build portals like *Shopping.com* where products with their prices from different retailers are prepared for the user.

Example 2.1 Figure 2.1 pictures two data sources, i.e., databases, describing countries. Both sources contain information about overlapping countries for which partially different aspects are described. Although the sources are useful in isolation, the value is considerably enhanced when the sources are integrated. For example, if someone likes to get the information about the capitals as well as the area, only the combination of these two sources satisfies the information need. Combining the information from both sources requires to come up with a variety of findings. Among others, it needs to be detected that *USA* and *United States of America* refer to the same country and *Population* and *Pop.* both state the population of a country.



Country

Country	Capital	Pop.
Russia	Moscow	140,702,000
Canada	Ottawa	33,213,000
USA	Washington DC	303,835,000

State

Name	Population	Area
Russia	140,000,000	17075200
Germany	80,000,000	357168
United States of America	300,000,000	9629091

Figure 2.1: Example of two sources describing countries.

In this chapter, we give on overview of the data integration principles which lays the foundations for the integration of web tables. We will start with an overview of the data integration tasks schema matching, data matching, and fusion (Section 2.1) together with a description of the challenges (Section 2.2). As the thesis focuses on the matching of web tables, we subsequently explain the general matching process as well as methods for both, schema and data matching (Section 2.3).

2.1 Tasks

Automatically integrating data introduces three tasks: schema matching, data matching and fusion [Christen, 2012]. Only if all tasks are performed, a full integration of the data across multiple autonomous sources can be achieved. In this section, we give definitions of the tasks and show how the results of each task look like.

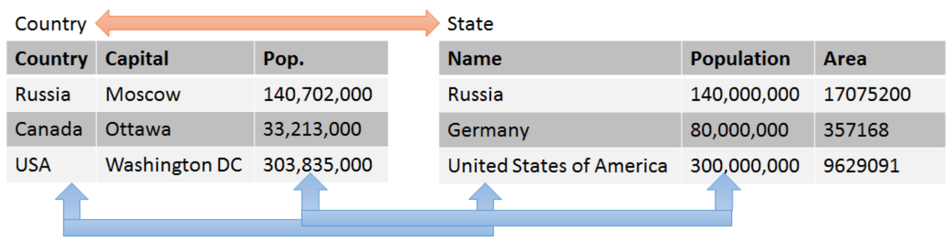
Schema Matching According to [Rahm and Bernstein, 2001], we define schema matching as follows:

Definition 2.1 *(Schema Matching) Given two schemata s and t, schema matching produces a set of correspondences, called mapping, indicating which elements semantically correspond to each other.*

Example 2.2 Figure 2.2 depicts the corresponding schema elements in the country sources. Both sources named *Country* and *State* describe the same concepts, indicated by the orange arrow. Further, *Country* and *Name* state the names of countries such that they can be considered as equivalent. In addition, *Pop.* and *Population* provide the same meaning. Thus, the schema elements are said to be equal, recognizable by the blue arrow. Neither the schema element *Capital* nor *Area* correspond to any other schema elements.

As can be seen from the example, equivalent schema elements do not necessarily have the same name. Thus, one of the challenge is to deal with different naming conventions as we discuss in the next section. Further, we will introduce common schema matching strategies in Section 2.3.3.

Data Matching According to [Köpcke and Rahm, 2010], we define data matching, also called record linkage, deduplication, instance matching, as follows:



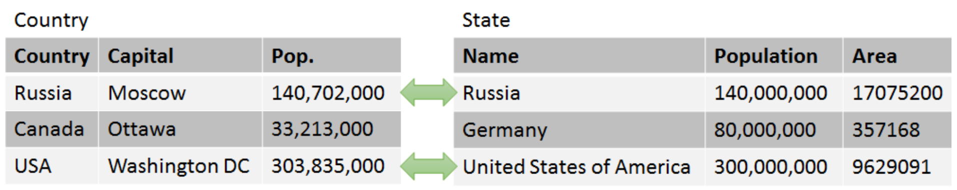
Country	Capital	Pop.
Russia	Moscow	140,702,000
Canada	Ottawa	33,213,000
USA	Washington DC	303,835,000

Name	Population	Area
Russia	140,000,000	17075200
Germany	80,000,000	357168
United States of America	300,000,000	9629091

Figure 2.2: Schema matching results of the country sources.

Definition 2.2 *(Data Matching) Given two sets of entities $A \in S_A$ and $B \in S_B$ of a particular semantic entity type from data sources S_A and S_B, data matching finds out which entities in $A \times B$ represent the same real-world objects.*

The data sources S_A and S_B do not need to be distinct. Especially the term deduplication refers to the task of finding the same entities within one source.



Country	Capital	Pop.
Russia	Moscow	140,702,000
Canada	Ottawa	33,213,000
USA	Washington DC	303,835,000

Name	Population	Area
Russia	140,000,000	17075200
Germany	80,000,000	357168
United States of America	300,000,000	9629091

Figure 2.3: Data matching results of the country sources.

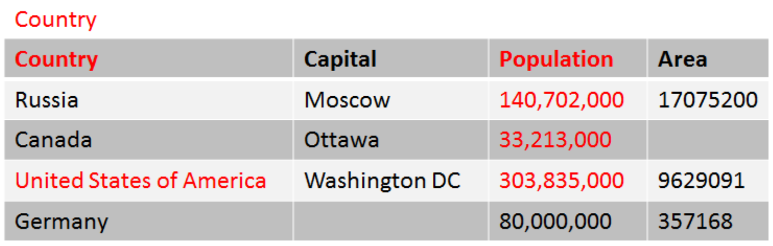
Example 2.3 In Figure 2.3, the entities referring to the same real-world object are shown. For example, both entities named *Russia* refer to the same country.

Similar to the schema matching, deviations in the names, e.g., *USA* and *United States of America*, can be detected. The knowledge about equivalent schema elements, e.g., *Country* and *Name*, is an important indication to identify same real-world objects. More information on commonly applied methods are depicted in Section 2.3.4. In order to match web tables to knowledge bases, both tasks schema as well as data matching need to be addressed.

Fusion The last step of the data integration process, the fusion, has the following goal [Bleiholder and Naumann, 2009]:

Definition 2.3 *(Fusion) Duplicate representations are combined and fused into a single representation while inconsistencies in the data are resolved.*

Example 2.4 Figure 2.4 shows how the sources have been fused. Based on the results from the schema and data matching, we know that the schema elements *Population* and *Pop.* have the same meaning as well as both sources include partially the same countries. For the values marked by red color, a decision has been taken which value to overtake since different values are found in the sources.



Country

Country	Capital	Population	Area
Russia	Moscow	140,702,000	17075200
Canada	Ottawa	33,213,000	
United States of America	Washington DC	303,835,000	9629091
Germany		80,000,000	357168

Figure 2.4: Fusion results of the country sources.

As we can derive from the example, the fusion needs to decide for the best suitable values. For example, the fusion chooses a value based on how reliable a source is estimated. We will present different fusion strategies that we apply on the results of the matching of web tables to knowledge bases in Chapter 5.

2.2 Challenges

Even for a small number of sources that provide structured data - the fully automated data integration scenario is notoriously hard [Doan et al., 2012]. This situation is deteriorated if not only a small amount of sources is taken into account. The challenges of big data integration are classified into four dimensions, also referred to as the "V" dimensions [Dong and Srivastava, 2015]:

- **Volume** Sources can have a huge volume and the number of sources grows.
- **Velocity** Sources are dynamic with continuous changes and updates.
- **Variety** Sources can be heterogeneous even if they describe the same topic.
- **Veracity** Sources differ in their quality, coverage, accuracy, and timeliness.

All three tasks need to tackle these challenges. Especially the variety poses one of the main challenges for both matching tasks. The reason for varying sources are different kinds of heterogeneities.

Heterogeneities occur on the schema as well as data level whenever data is either described or structured differently. Many classification schemes have been proposed to categorize different types of heterogeneities. We classify heterogeneities into three types as presented by [Euzenat and Shvaiko, 2007] for ontologies and [Busse et al., 1999] for databases:

- **Syntactic Heterogeneity** occurs when two sources are different on the syntactic level, e.g., the use of different ontology languages or protocols for databases.
- **Terminological Heterogeneity** arises due to variations in names when referring to the same real-world objects in different data sources. This can

mean that two things can either have the same name but a different meaning (homonym) or other names but the same meaning (synonyms). Other reasons are the usage of technical sublanguages or abbreviations. Another term for terminological heterogeneity is semantic heterogeneity.

- **Conceptual Heterogeneity** describes differences in modeling the same domain. It can be divided into three subtypes: coverage, granularity, and perspective. Variations in the coverage mean that only partially overlapping domains are described. Following, a different granularity refers to a description using a different level of detail. Another perspective, also called scope, indicates that fully overlapping domains are expressed on the same granularity level but the sources focus on different aspects, e.g., describing a political map and a geological map of a country. Especially in the database community, the expression schematic heterogeneity is used.

When integrating a set of sources, usually all kinds of heteoregeneties occur. Thus, matching methods need to find ways to overcome these heteoregeneties.

2.3 Matching

Matching refers to the mission of finding semantically corresponding concepts. This can either be on the schema or on the data level. In this section, we explain how the matching process looks like and which similarity measures are commonly applied to estimate the similarity of sources. Further, we describe schema and data matching techniques which try to overcome the presented challenges. Finally, we depict the evaluation criteria that are used to evaluate different matching approaches. Everything presented in this section provides the foundations we build on for matching web tables to knowledge bases.

2.3.1 Process

Similar to [Euzenat and Shvaiko, 2007], we define the matching process as follows:

Definition 2.4 *(Matching Process) A matching process can be seen as function f that takes two inputs i_1, i_2 and returns a set of correspondences, called mapping M, $M = f(i_1, i_2)$.*

The input can either be a single source, e.g., a database, or a set of sources like millions of web tables.

Definition 2.5 *(Correspondence) A correspondence is a quadruple $< e_1, e_2, r, c >$ that holds between the elements e_1 and e_2 from the input i_1 resp. i_2 given a relation r and a confidence c.*

Relations indicate how the elements are connected. The most common relation is the equivalence relation stating that two elements have the same meaning. Other

relations are subsumptions or complex ones relating more than two elements [Ritze et al., 2010]. In this thesis, we only consider equivalent relations.
The confidence value specifies how likely the relation holds. It varies between 0 and 1 while 1 declares that the correspondence is most likely to hold. The more evidence we obtain to support a correspondence, the more reliable the confidence value becomes. For example, if a correspondence between two entities is generated only based on comparison of entity labels, a high similarity and in turn a high confidence score does not necessarily lead to a correct correspondence. A mapping is per se not restricted to only contain one correspondence per element. However, we focus on so called one-to-one (1:1) mappings which specify that only one correspondence for each element can exist [Euzenat and Shvaiko, 2007].

The general matching process as presented in Figure 2.5 consists of four steps: preprocessing, matcher execution, aggregation, and classification [Do and Rahm, 2002, Christen, 2012]. In the following, we describe each step and present common methods.

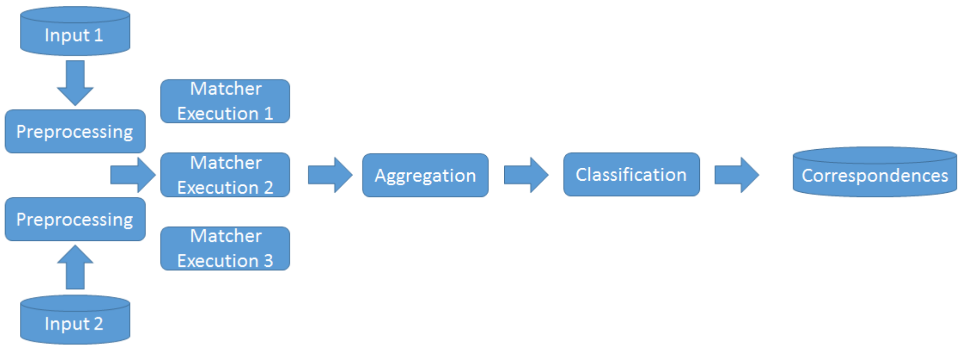


Figure 2.5: General matching process with two sources as input and a set of correspondences as output.

Preprocessing During the preprocessing, everything is arranged to be able to apply the matchers in a meaningful way. This includes the unification of the data since it can vary in the format, structure and content. For both, schema and data matching, a set of techniques has been proposed [Christen, 2012, Euzenat and Shvaiko, 2007, Rahm and Do, 2000]. A first step is the standardization that transposes values into a consistent representation by lower casing all values and applying specific normalization methods if possible, e.g., to unify addresses. Afterwards, unwanted characters or words are removed. This can include punctuations but also stop words that do not present relevant information. Additionally, abbreviations can be expanded or misspellings corrected. Different structures are resolved by structural transformation like the segmentation. For example, one database contains two attributes covering the first and last name and another database contains one attribute with combined first and last name. By splitting the content of the attributes, both databases cover the information in the same way.

Matcher Execution One of the main matching challenges is to overcome heterogeneities. As introduced, heterogeneities arise due to different ways of expressing knowledge, e.g., by using other terminologies or considering different levels of details. Therefore, a set of matchers is executed. Illustrated in Figure 2.6, a matcher takes two features of two elements as input and applies a similarity measure. For example, to match databases about persons, the names of the persons are useful features. As result, a matcher returns a similarity matrix including pairwise similarity scores to indicate how similar the features and in turn the elements are. The reason for using more than one matcher is that a single feature is usually not enough to overcome the heterogeneities. Further, the more features are used, the more evidence can be gained. Evidence can either be positive or negative to support or disapprove a possible correspondence. Gaining more evidence helps to better decide if a correspondence is likely to be correct. Within the next sections, we will go into more detail about similarity measures (Section 2.3.2) and matching strategies for schema and data matching (Sections 2.3.3 & 2.3.4).

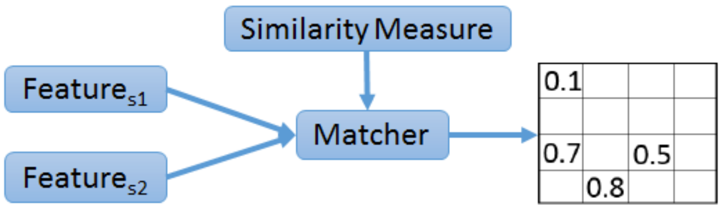


Figure 2.6: Functionality of a matcher with a similarity matrix as output.

Aggregation Each matcher generates a similarity matrix holding the pairwise element similarities. To get an overall estimate whether two elements are similar, these matrices need to be combined, resulting in one similarity score per pair. Figure 2.7 shows how the aggregation looks like. In this example, the two similarity matrices are combined in an aggregated similarity matrix by summing up the individual similarity scores. The most straightforward aggregation strategies are to take the sum, minimum, maximum or the average of the matrix elements [Do and Rahm, 2002]. However, these strategies do not consider whether one matrix created by a particular matcher is more reliable or trustful.

Thus, weighted-based aggregation methods, also called combiners, have been introduced. They assign weights to matchers to indicate their influence on the final similarity score [Doan et al., 2012]. One way to define the weights is to use hand-crafted rules. Since the manual creation of such rules is time-consuming and knowledge about the topic is required, it is not feasible to create rules for cross-domain sources. Another option is to learn weights in a supervised fashion. Independent of the strategy, a combined similarity matrix is generated that serves as

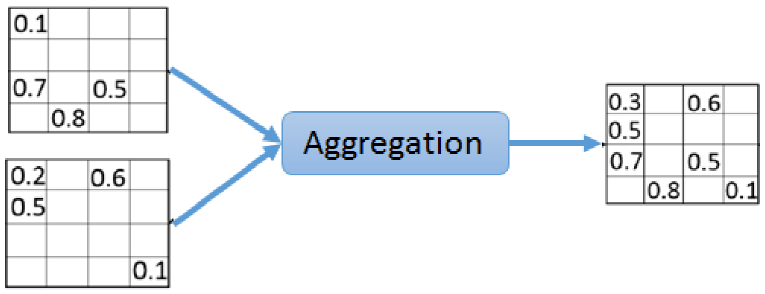


Figure 2.7: Aggregation of two similarity matrices.

input for the last step, the classification.

Classification During the classification, it is decided whether a correspondence is likely to hold and in turn will be part of the mapping. A simple classification method, called threshold-based classification, divides correspondences into matches and non-matches by applying a threshold on the similarity score [Euzenat and Shvaiko, 2007]. Other classification methods are supervised approaches that train a binary classifier. Especially for data matching, supervised classifier using decision trees have been shown as valuable [Verykios et al., 2000].

Independent of the input or whether the schema or data level is considered, the matching process has to be performed. For each step, a variety of strategies have been proposed while some of them are specific for schema or data matching. Especially the matcher execution, which is the most fundamental part of the matching process, highly depends on the matching task and input. Thus, in the next sections, we focus on the matcher execution and present commonly used similarity measures as well as common methods for schema and data matching.

2.3.2 Similarity Measures

Due to different kinds of heterogeneities reviewing whether two features are equal is usually not enough to decide whether the elements correspond. Therefore, matchers utilize a similarity measure that allows deviations. We will focus on general measures that can be used by matchers for schema as well as data matching. Most similarity measures presented in this section are took up in subsequent chapters for matching web tables to knowledge bases.

Definition 2.6 *(Similarity Measure) A similarity measure is a real-valued function identifying the similarity between features x, y: $sim(x, y) \rightarrow [0, 1]$.*

A similarity score of 1 states equality. The lower the similarity score is, the less similarity is assumed. Similarity measures can be seen as the inverse of distance metrics: if two features are not similar at all, the distance score is 1.

We will start with an introduction of string similarity measures, see [Cohen et al., 2003] for an overview. In more detail, we discuss edit-based, token-based as well as hybrid measures. Since similarity measures are rarely useful for numeric or date values, we present how values of other data types can be compared.

Edit-based String Measures The idea behind edit-based string measures is to determine how one string needs to be edited to present another string. Such measures are especially useful to identify typos or character swaps. One edit-based string measure has been introduced by [Levenshtein, 1966]. To compute the Levenshtein distance, three edit operations are considered: insertion, deletion, and replacement. All of them have the same costs of 1.

Definition 2.7 *(Levenshtein Distance) Given two strings s_1 and s_2, the Levenshtein distance $dist_{lev}$ is the minimum number of edit operations that are necessary to transform s_1 to s_2.*

Definition 2.8 *(Levenshtein Similarity) The Levenshtein similarity $dist_{lev}$ is defined as the inverse normalized Levenshtein distance.*

$$sim_{lev}(s_1, s_2) = 1 - \frac{dist_{lev}(s_1,s_2)}{max(|s_1|,|s_2|)}$$

Example 2.5 Let s_1 = *United States*, s_2 = *United Stats*

$$sim_{lev}(s_1, s_2) = 1 - \frac{1}{13} = 0.92$$

Other edit-based string measures vary in set of edit operations or the costs of an operation: the Jaro-Winkler similarity adds a transposition operator while the Smith-Waterman distance adapts the cost of an edit operation depending on the position of the character. A weakness of edit-based string measures is that they always consider a string as a whole regardless whether it consists of single terms. For example, if two strings contain the first and last name but the order is inverted, they will not be considered as similar at all.

Token-based String Measures To compensate the disadvantages of edit-based string measures, token-based string measures focus on the comparison of the individual terms or tokens a string consists of. Common measures are the Dice and the Jaccard similarity. Here, we will focus on the Jaccard similarity [Jaccard, 2006].

Definition 2.9 *(Jaccard Similarity)*

$$sim_{jacc}(s_1, s_2) = \frac{|tokenize(s1) \cap tokenize(s2)|}{|tokenize(s1) \cup tokenize(s2)|}$$

Example 2.6 Let s_1 = *United States of America* and s_2 = *United States*

tokenize(United States of America) = {United, States, of, America}
tokenize(United States) = {United, States}

$$sim_{jacc}(s_1, s_2) = \frac{2}{4} = 0.5$$

The Jaccard measure weights each token independently of its relevance. Hence, the string "Federated States of Micronesia" has the same similarity to "United States of America" as "United States" because in both cases two tokens overlap. However, the tokens "states" and "of" are less relevant. Therefore, the TF-IDF concept has been introduced that considers if the overlapping terms are distinguishing [Salton and McGill, 1983].

Definition 2.10 *(Term Frequency) Let t be a term and d a document*

$$tf(t,d) = |\{t \in d|\}$$

Definition 2.11 *(Inverse Document Frequency)*

$$idf(t) = log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Document vectors are built for each document and cover the TF-IDF scores of the terms in the document: $v_d(t) = tf(t,d) \cdot idf(t)$. These vectors serve as input for the cosine similarity which computes the cosine of the angle between the two document vectors in an n-dimensional space.

Definition 2.12 *(Cosine Similarity)*

$$sim_{cosine}(p,q) = \frac{\sum_{t \in T} v_p(t) \cdot v_q(t)}{\sqrt{\sum_{t \in T} v_p(t)^2} \cdot \sqrt{\sum_{t \in T} v_q(t)^2}}$$

In contrast to edit-based measures, token-based measures always assume that the tokens are equal and do not allow any deviation. Thus, typos or slight variations can result in a low similarity.

Hybrid Measures The introduced measure families show mutual disadvantages. Thus, hybrid measures combine edit- and token-based measures by allowing deviations within a token-based comparison. One hybrid measure is the hybrid Jaccard similarity which uses an inner edit-based similarity measure, e.g., the Levenshtein similarity, and afterwards applies the Jaccard similarity [Herschel and Naumann, 2010].

Definition 2.13 *(Hybrid Jaccard Similarity) Let S be the set of shared tokens and $U(x), U(y)$ the sets of unique tokens:*

$$S = \{(x_i, yi) | x_i \in tokenize(x) \wedge y_i \in tokenize(y) : sim(x_i, y_i) \geqslant \theta\}$$
$$U(x) = \{x_i | x_i \in tokenize(x) \wedge y_i \in tokenize(y) \wedge (x_i, y_i) \notin S\}$$
$$U(y) = \{y_i | y_i \in tokenize(y) \wedge x_i \in tokenize(x) \wedge (x_i, y_i) \notin S\}$$
$$sim_{hybJacc}(x,y) = \frac{\sum_{(x_i,y_i) \in S} sim(x_i,y_i)}{\sum_{(x_i,y_i) \in S} sim(x_i,y_i) + |U(x)| + |U(y)|}$$

Example 2.7 Let s_1 = *nba wnba mcgrady* and s_2 = *macgrady nba*

$$\text{tokenize(nba wnba mcgrady)} = \{nba, wnba, mcgrady\}$$
$$\text{tokenize(macgrady nba)} = \{macgrady, nba\}$$
$$sim_{lev}(nba, nba) = 1 \qquad sim_{lev}(mcgrady, macgrady) = 0.83$$
$$sim_{hybJacc}(s_1, s_2) = \frac{1.83}{1.83+0+1} = 0.65$$

Depending on the characteristics of the strings, a different similarity measure can be more beneficial. All introduced measures are useful to compare strings but they yield unsatisfactory results for other data types like numerical values, e.g., comparing the years "1999" and "2000" as strings results in no similarity at all [Herschel and Naumann, 2010]. Therefore, type-based similarity measures have been developed. Following, we present a similarity measure for numeric values and one for dates.

Numeric Similarity Measure Most obvious, the similarity of two numeric values can be determined based on their deviation. The deviation similarity introduced by [Rinser et al., 2013] is defined as follows:

Definition 2.14 *(Deviation Similarity)*

$$sim_{num}(n_1, n_2) = \begin{cases} 1 & \textit{if } n_1 = n_2 \\ 0.5 \cdot \frac{min\{|n_1|,|n_2|\}}{max\{|n_1|,|n_2|\}} & \textit{otherwise} \end{cases}$$

Example 2.8 Let $n_1 = 19.3$ and $n_2 = 20.0$

$$sim_{num}(n_1, n_2) = 0.5 \cdot \frac{19.3}{20.0} = 0.48$$

Date Similarity Measure For dates, the deviation needs to be calculated in another way. The weighted date part similarity bases on the idea that more emphasize should be put on the year and less on the specific day.

Definition 2.15 *(Weighted Date Part Similarity)*

$$sim_{date}(d_1, d_2) = \frac{sim_{day} \cdot weight_{day} + sim_{month} \cdot weight_{month} + sim_{year} \cdot weight_{year}}{weight_{day} + weight_{month} + weight_{year}}$$
$$sim_{day}(d_1, d_2) = \frac{31 - |day(d_1) - day(d_2)|}{31}$$
$$sim_{month}(d_1, d_2) = \frac{12 - |month(d_1) - month(d_2)|}{12}$$
$$sim_{year}(d_1, d_2) = 1 - \frac{|year(d_1) - year(d_2)|}{yearRange}$$

The year range determines between which years the date values can be found. If dates between 3000 BC and 2000 AD are compared, a difference of one year might be tolerable. In contrast if only dates between the years 1986 and 1987 are considered, one year makes a big difference.

Example 2.9 Let d_1 = *24/10/1986* d_2 = *1/11/1986*,
$weight_{day} = 1$, $weight_{month} = 3$, $weight_{year} = 5$, $yearRange = 100$

$$sim_{day}(d_1, d_2) = \frac{|31-8|}{31} = \frac{23}{31} \; sim_{month}(d_1, d_2) = \frac{12-1}{12} = \frac{11}{12}$$
$$sim_{year}(d_1, d_2) = 1 - \frac{1986-1986}{100} = 1 - \frac{0}{100} = 1$$
$$sim_{date}(d_1, d_2) = \frac{\frac{23}{31}\cdot 1 + \frac{11}{12}\cdot 3 + 1\cdot 5}{1+3+5} = \frac{0.74+2.75+5}{9} = 0.94$$

In conclusion, a variety of similarity measures exists and each measure can be more suitable depending on the characteristics of the features. Since similarity measures are the core of the comparison and in turn of the matching, the choice of the similarity measures has a strong influence on the matching result. Besides general measures, more specific ones can be applied that are tailored to the particular matching task or the input type.

2.3.3 Schema Matching

One of the major bottlenecks of the data integration is to get semantic mappings between the sources [Halevy et al., 2006]. Thus, plenty of schema matching methods have been developed. A classification scheme for the approaches has been proposed by [Rahm and Bernstein, 2001] and extended by [Bilke, 2006]. The scheme is depicted in Figure 2.8 with the following division of matching approaches:

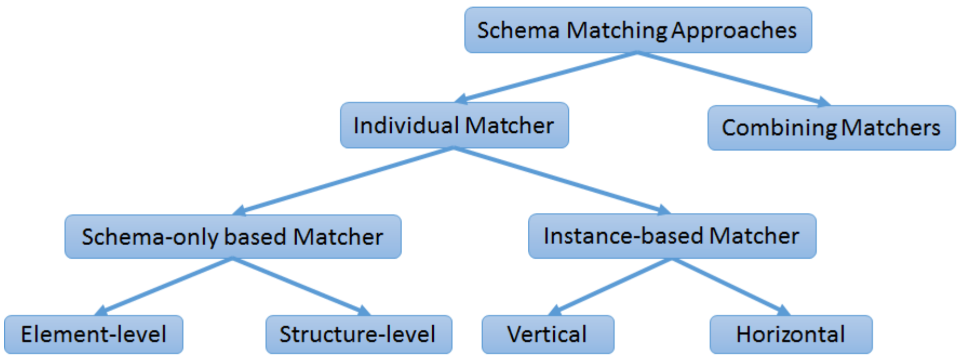


Figure 2.8: Schema matching approaches according to [Bilke, 2006].

- Individual vs. Combined: Matchers either rely on one or a set of features.
- Schema vs. Instance: Methods consider either only the meta data available on schema-level or the data itself.
- Element vs. Structure: Schema-based approaches focus on features extracted from single elements, e.g., element name or data type, or exploit complex schema structures, e.g., graph matching.
- Vertical vs. Horizontal: Instance-based strategies compute characteristics of the data, e.g., average length of values of an element, or perform a duplicate-based method that detects duplicated entities.

Each type of approaches is appropriate for sources with specific characteristics. Within the country example, applying a schema-based method that only compares the attribute name *Country* and *Name* will not find a similarity between the semantically corresponding attributes. Contrary, a duplicate-based method is able to detect the correspondence since it can rely on the knowledge that the values of the entities representing the same countries are similar. To overcome different heterogeneities, schema matching systems usually include matchers from more than one category. The most prominent systems to perform schema matching for databases are COMA [Do and Rahm, 2002] and Cupid [Madhavan et al., 2001]. Both provide a selection of methods on the element as well as structure level. Exemplary matchers compare attribute names under consideration of synonyms or abbreviations, exploit data types as well as use the tree that is spanned when representing the database graphically. Similar to most systems, they do not consider the instance level. Systems including instance-level matchers are LSD [Doan et al., 2001] with its successor GLUE [Doan et al., 2004a], iMAP [Dhamankar et al., 2004] as well as COMA's successor COMA++ [Aumueller et al., 2005]. However, they all focus on vertical instance-based strategies. Only the Dumas system [Bilke and Naumann, 2005] considers the entities in a duplicate-based method. For matching ontologies, systems like RiMOM [Li et al., 2009] or LogMap [Jiménez-Ruiz and Cuenca Grau, 2011] include a variety of methods. Detailed overviews are provided by [Rahm and Bernstein, 2001] and [Euzenat and Shvaiko, 2007].

To find out strengths and weaknesses of individual schema matching systems, the Ontology Alignment Evaluation Initiative[1] (OAEI) performs an annual evaluation. For this purpose, a variety of ontologies together with correspondences has been made publicly available, each test set focusing on a specific aspect. Hence, a comparability of the results provided by the systems is enabled. Besides the benchmark provided by the OAEI, other commonly used datasets are Purchase Order or Web Directory.[2] To evaluate schema matching on large scale, systems are applied to match knowledge bases like DBpedia and Freebase.

2.3.4 Data Matching

For data matching, it is assumed that at least some schema correspondences are available. This is straightforward in case of deduplication where duplicates within the same source are detected. Otherwise, the correspondences between the schemata are known by a preceded schema matching. One of the main challenges for data matching is the quadratic number of pairwise comparisons. With a native approach, $n \times m$ comparisons are required with n and m entities in the sources, respectively.

Blocking To reduce the number of comparisons, blocking, also named indexing, is applied. It removes pairs of entities that are unlikely to correspond [Chris-

[1]`http://oaei.ontologymatching.org/`

[2]`https://dbs.uni-leipzig.de/bdschemamatching`

ten, 2012]. For this, each entity is inserted into one or more blocks according to a blocking/sorting key and only entities within the same blocks are compared. Common blocking keys rely on identifiers, if existing, or on the similarity of the entity label sound determined by a phonetic encoding function. The standard blocking approach puts each entity into exactly one block [Fellegi and Sunter, 1969]. Thus, entities can be efficiently blocked but the quality heavily depends on the choice of the blocking key. An alternative that tries to overcome this issue is the sorted neighborhood method [Hernández and Stolfo, 1995]. It sorts the sources according to a key and uses a sliding window to go through the sources. Whenever two entities are in the same window, a candidate pair is generated. Independent of the blocking strategy, all subsequent steps are only performed on the pairs that are worth comparable. An overview of blocking methods, including advanced approaches like Q-Gram Indexing or Canopy Clustering is provided by [Christen, 2012].

As for schema matching, another challenge is the presence of heterogeneities. This includes different naming of the entities, varying entity descriptions using diverse attributes or the same attributes but with differing values due to mistyping, naming conventions, missing values, and so on. A variety of approaches trying to solve the data matching problem has been proposed in literature [Christen, 2012, Christophides et al., 2015]. One set of methods create matching or linkage rules that specify which conditions need to hold to consider entities as equivalent. The condition is usually a linear weight of the similarities which are computed using attribute-specific similarity measures. A possible linkage rule for the country example is to compare the entity labels as well as the values of the population attributes using the Jaccard and deviation similarity and generate a correspondence if the sum is above 0.5. Similar to the weighted aggregation, one possible way is to use hand-crafted rules which is very time-consuming and knowledge about the topic is required. Therefore, machine learning methods are commonly used: either unsupervised methods that cluster entities or supervised ones which learn a rule.

Overviews of data matching systems are provided by [Elmagarmid et al., 2007, Köpcke and Rahm, 2010]. Systems are either provided with matching rules by the user like Limes [Ngomo and Auer, 2011] or learn the rules based on training data like Active Alias (decision trees) [Tejada et al., 2001], MARLIN (SVM) [Bilenko and Mooney, 2003] as well as the Silk framework [Isele and Bizer, 2012] and RAVEN [Ngomo et al., 2011]. Silk and RAVEN both apply a genetic algorithm in combination with active learning to optimize the training data generation. The TAILOR [Elfeky et al., 2002] and FEBRL [Christen, 2008] frameworks provide a toolbox including both, supervised and unsupervised strategies. All mentioned systems focus on the comparison of attribute values and do not include any other features. However, other systems consider so-called context features [Köpcke and Rahm, 2010]: contextual information, e.g., semantic relationships, in a graph structure that allows the propagation of similarities to related entities. One system exploiting contextual information is the Context Based Framework [Chen et al., 2009].

As for the schema matching, the OAEI also provides data sets for the evaluation of data matching systems. Further, two common benchmarks to measure the quality and performance of data matching approaches are the Cora [McCallum et al., 2000] and Restaurant dataset [Tejada et al., 2001]. While the Cora dataset covers research papers, the Restaurant dataset describes restaurants with their addresses. A larger and more complex dataset is the DBpediaDrugbank dataset in which the knowledge base DBpedia and the DrugBank database[3] are compared.

2.3.5 Evaluation Criteria

Methods performing either schema or data matching are evaluated by stating their quality together with additional performance measures, if desired. The quality of a matching approach is usually specified using the standard measures precision, recall, and its combination F-measure. These measures are computed based on a manually determined perfect result, mostly provided by a gold standard containing all correct correspondences that exist between the sources. The set of correspondences generated by a matching method is comprised of True Positives (TP) and False Positives (FP), i.e., correctly and falsely proposed correspondences.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

$$F - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.3}$$

Instead of taking the harmonic mean of precision and recall, different weights can be assigned to precision and recall for the F-measure computation. Two other commonly F-measures are the F-measure$_2$ and the F-measure$_{0.5}$. While the F-measure$_2$ puts a higher weight on recall, the F-measure$_{0.5}$ weighs recall lower than precision. A higher weight for precision can for example be useful if an application requires precise correspondences and at the same time a lower amount of correspondences is acceptable. However, since most matching systems focus on the harmonic mean between precision and recall, we will use it in the evaluations of this thesis to ensure the comparability of the results.

In addition to the quality, performance measures like the execution time, memory consumption, scalability, or user-related measures like the level of user input effort are applied [Euzenat and Shvaiko, 2007]. While performance measures are less important for schema matching, it is an essential factor for the data matching [Köpcke and Rahm, 2010] due to the large amounts of comparisons. For example, the blocking performance is measured in terms of pair completeness and reduction ratio.

[3] `http://wifo5-04.informatik.uni-mannheim.de/drugbank/`

Chapter 3

Knowledge Bases

The large amounts of data have become one of the greatest database research challenge of our time [Abadi et al., 2014]. To be able to handle the enormous volume of data that is generated every day, automatic methods to analyze this data are required. However, this is a knowledge-intensive task which can be facilitated whenever knowledge about the domain is available. Knowledge can be captured in knowledge bases, also called knowledge graphs. They contain information of the subjects of one or more domains. Knowledge bases are already used as background knowledge in an increasing range of applications like web search, natural language understanding, data integration, and data mining [Lehmann et al., 2015].

A knowledge base is defined as the combination of an ontology and instances of the classes in the ontology [Staab and Studer, 2009]. Usually, classes are arranged in class hierarchies. Properties defined in the ontology are used to describe relationships that hold between the instances (object properties) as well as give details about the instances (datatype properties). Knowledge bases follow the Semantic Web vision to give information a well-defined meaning and to enable computers and humans to work in cooperation [Berners-Lee et al., 2001]. This is realized by the linked data principles including the recommendation to use standards.

In this chapter, we introduce knowledge bases which are one input of the web table to knowledge base matching process. In Section 3.1, we describe the preliminaries, including the Resource Description Framework (Section 3.1.1) and the Linked Data principles (Section 3.1.2) which are utilized by knowledge bases. Then, we characterize the most commonly applied knowledge bases in Section 3.2. The knowledge base we use in this thesis is DBpedia which is portrayed in more detail in the last section of the chapter (Section 3.3).

3.1 Preliminaries

To enable the Semantic Web vision, a collection of recommendations have been proposed. One of them is to use the Resource Description Framework to represent data. In addition, the linked data principles provide a set of rules how to describe the data. In this section, we provide an introduction to both since they build the basis for knowledge bases.

3.1.1 Resource Description Framework

The Resource Description Framework (RDF) [Klyne and Carroll, 2004] is a standard introduced by the World Wide Web Consortium (W3C). It is the most common way to represent data in the Semantic Web. In general, the RDF framework describes resources using a generic graph-based data model. Information about resources are represented by triples having the following format:

$$< subject >< predicate >< object > \tag{3.1}$$

Each triple state a relationship between the subject and the object, represented by the predicate. While the subject always refers to a resource, objects can either point to resources or to literals like strings or numbers. Resources and predicates are identified by a unique identifier, more precisely by an Internationalized Resource Identifier (IRI). IRIs are an extension of Uniform Resource Identifiers (URIs) to overcome the limitation that only ASCII characters can be used. The only exception where resources do not have an IRI are blank nodes. Blank nodes, also called anonymous resources, are resources without an identifier. They are for example used to describe reification, e.g., to provide provenance information, where it is not necessary to have an explicit identifier.

RDF itself only provides the model and does not define the format of the data. To this end, several serialization formats have been introduced. Exemplary serialization formats are RDF/XML, N-Triples or Turtle. Each format has been developed with another purpose in mind, e.g., Turtle focuses on the readability for humans. Listing 3.1 shows a RDF description, serialized in Turtle, of the city Mannheim in Germany. The first three rows specify the namespaces such that they do not need to be repeated each time. While line five defines that the resource is a city, line six states that the name of the resource is Mannheim. The meaning of the predicates is defined by the predicate provider. Ideally, the IRI of a predicate refers to its specification. The remaining part states the area code of Mannheim by using a literal in the object position as well as indicates the relation of the city to the country Germany.

Upon RDF, ontology languages like RDFS or OWL have been defined to express the ontology in more detail. For example, class hierarchies or characteristics of the properties can be defined, like determining that a property is functional using

Listing 3.1: Example of triples about the city Mannheim in the Turtle syntax.

```
@prefix exr: <http://example.org/resource/> .
@prefix exo: <http://example.org/ontology/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

exr:Mannheim rdfs:type exo:City .
exr:Mannheim rdfs:label "Mannheim"@en .
exr:Mannheim exo:areaCode "0621" .
exr:Mannheim exo:country exo:Germany .
```

owl:FunctionalProperty. The most common way to store RDF triples is to use a triplestore. On top of the storage, triplestores often provide access to the data by enabling queries, using the query language SPARQL. Altogether, an environment is created in which data can be described in a uniform way.

3.1.2 Linked Data Principles

In order to achieve the goal of creating a Web of Data which is one part of the Semantic Web vision, Linked Data constitutes a paradigm of publishing and interlinking datasets on the Web [Bizer et al., 2009a]. For that reason, Tim Berners-Lee defined the four core principles of linked data [Berners-Lee, 2006]:

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF(S), SPARQL).
4. Include links to other URIs so that more things can be discovered.

The first three principles are concerned with publishing sources on the Web to provide a well-defined meaning and to enable humans to work in cooperation. In addition, the forth principle gives attention to connecting sources among each other. The success of the Linked Data idea becomes visible when having a look at Linked Open Data (LOD) Cloud as depicted in Figure 3.1. It contains over 1 00 datasets covering different topics that are interlinked. Some datasets like DBpedia stands out for being central and highly interconnected. In the next section, we will present some of these datasets which are also referred to as knowledge bases.

[1]Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

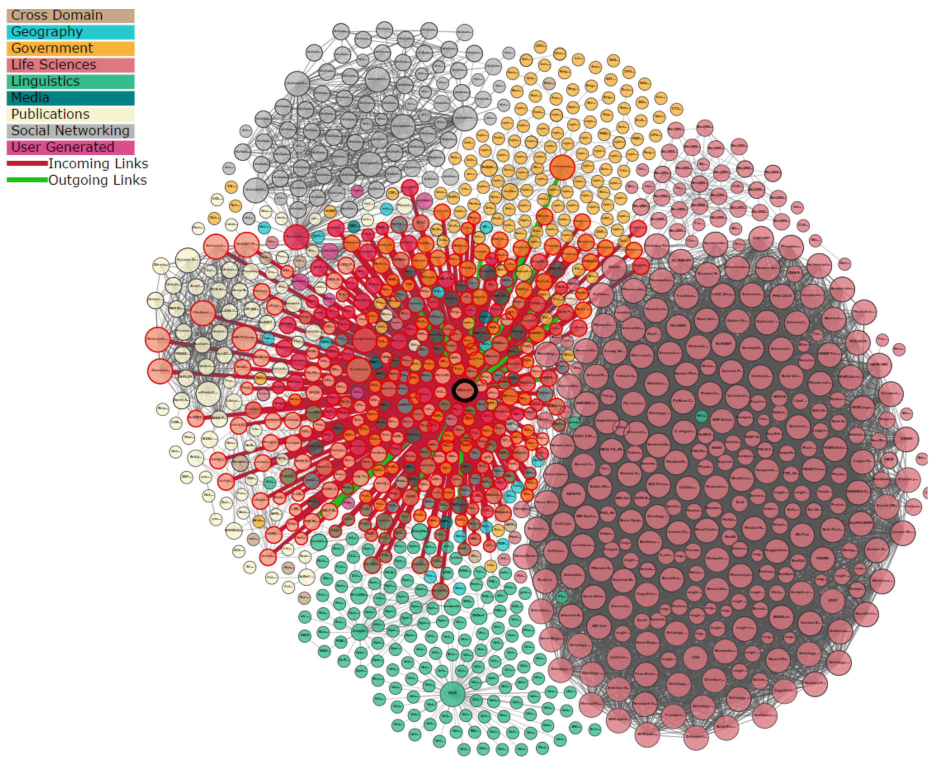


Figure 3.1: Linked Open Data Cloud as of June 2017.[1]

3.2 Common Knowledge Bases

Knowledge bases serve as background knowledge in various applications. For example, the Google Knowledge Graph is used to improve the Google's Web search [Singhal, 2012]. While some knowledge bases focus on specific domains, e.g., on food microbiology [Hignette et al., 2007], cross-domain knowledge bases cover a wide range of topics. Some of the most common cross-domain knowledge bases are DBpedia, Yago, Freebase, and Wikidata. Contrary to knowledge bases owned by companies, these knowledge bases are publicly available. In the following, we provide an overview of the knowledge bases and describe their individual characteristics. A more detailed overview is given by [Färber et al., 2016].

DBpedia [Bizer et al., 2009b] has positioned itself as center of the LOD Cloud as it is highly interconnected with many datasets. DBpedia is automatically extracted from Wikipedia by utilizing infoboxes, categorization information, geo-coordinates and so on. To map the information extracted from the infoboxes to the classes and properties of the ontology, templates are generated via a world-wide crowd-sourcing effort. Altogether, the current version of DBpedia (version 2016-10) covers 6.6 million instances that contribute in over 1.7 billion facts. As a specific characteristic, the ontology is very broad with 1 000 different properties.

YAGO, Yet Another Great Ontology, automatically extracts information from Wikipedia, WordNet and Geonames. The latest version, YAGO 3 [Mahdisoltani et al., 2015], covers about 10 million instances and 120 million facts. As the relative number of instances compared to facts indicates, the YAGO ontology covers only about 100 properties. In DBpedia, the quality of the mappings between the attributes found in infoboxes and the properties depend on the quality of the templates that are generated by humans. To generate YAGO, a fully automated approach is applied which is correct in about 95% . A special feature of YAGO is that its ontology is time and space aware. Thus, instances and facts are annotated with temporal and special information. For example, the fact that Barack Obama is the president of the United States is only valid for a certain duration.

Freebase [Bollacker et al., 2008] is a collaboratively created knowledge base. It integrates data from Wikipedia, NNDB, FMD, and MusicBrainz. In contrast to YAGO and DBpedia, an interface is provided such that users can edit the data. With more than 1 billion facts described by 7 000 properties, Freebase is one of the largest knowledge bases. However, the project has been shut down and superseded by the Wikidata project in 2015.

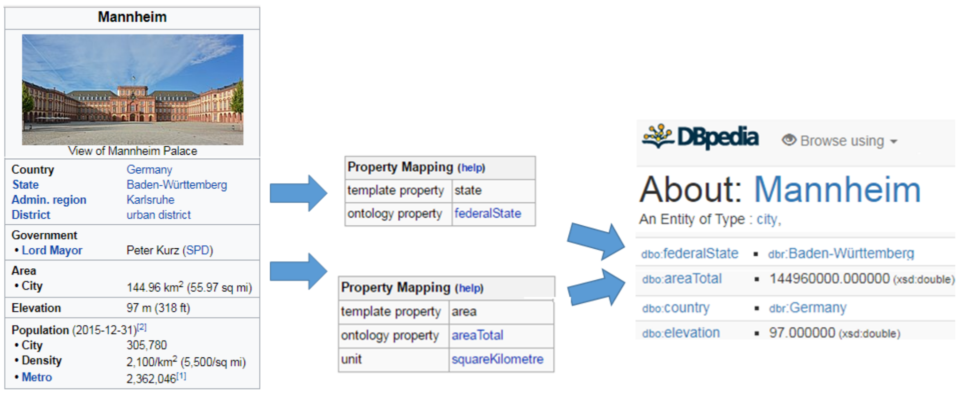
Wikidata [Vrandečić and Krötzsch, 2014] is the successor of Freebase. The data is added and maintained by thousands of users. About 25 million instances are described in more than 350 languages. One important aspect of Wikidata is the provenance of facts. Each fact is assigned with its source such that it is possible to track where the information comes from.

Google, Yahoo!, and Microsoft Knowledge Graph Despite the publicly available knowledge bases, companies like Google [Singhal, 2012] or Yahoo! [Torzec, 2014] or Microsoft [Qian, 2013] own knowledge graphs. Details about the particular content or the size are hardly available.

3.3 DBpedia

DBpedia [Lehmann et al., 2015] has recently be titled as "one of the cornerstones of Semantic Web and Linked Data research being the subject and center of a large number of research papers over the past few years" [Neumaier et al., 2017]. The extensive usage of DBpedia in the Semantic Web becomes apparent from the LOD Cloud depicted in Figure 3.1. The DBpedia knowledge base is located in the center of the cloud, highlighted by a black circle. Edges between the datasets indicate their interlinking. All red lines show that links from a dataset to DBpedia exist, green lines symbolize connections from DBpedia to a dataset. In the LOD Cloud, datasets from different topical domains provide links to DBpedia. Topical domains are for example geography, government or publications. Thus, DBpedia is perceived as cross-domain knowledge base. Beside the use of DBpedia by the research community, it is also taken into account in commercial environments, e.g., by BBC [Kobilarov et al., 2009] and the New York Times [Sandhaus, 2010].

All facts in DBpedia are automatically extracted from Wikipedia articles. More precisely, the infoboxes found on the top right of Wikipedia articles are used since they provide a structured summary of the instance that is described in the article. By using templates to map the infobox content to the DBpedia ontology, RDF triples are generated. Templates contain instructions which attributes from the infoboxes correspond to which properties in the DBpedia ontology. In Figure 3.2, the triple extraction for the city Mannheim is illustrated. On the left-hand side, the infobox of the city Mannheim in Wikipedia is shown. It contains information like the state or the area in a structured way. The template for German locations specifies that the attribute with the name *area* is mapped to the property *areaTotal* as well as *state* to *federalState*. Based on the mappings, the information are extracted and triples are generated. If additionally a unit is given, the values are converted into the according unit. In the example, the area which has been given in square kilometers is converted into square meters as required by the property.



Mannheim	
View of Mannheim Palace	
Country	Germany
State	Baden-Württemberg
Admin. region	Karlsruhe
District	urban district
Government	
• Lord Mayor	Peter Kurz (SPD)
Area	
• City	144.96 km² (55.97 sq mi)
Elevation	97 m (318 ft)
Population (2015-12-31)[2]	
• City	305,780
• Density	2,100/km² (5,500/sq mi)
• Metro	2,362,046[1]

Property Mapping (help)	
template property	state
ontology property	federalState

Property Mapping (help)	
template property	area
ontology property	areaTotal
unit	squareKilometre

About: Mannheim
An Entity of Type : city,

dbo:federalState	▪ dbr:Baden-Württemberg
dbo:areaTotal	▪ 144960000.000000 (xsd:double)
dbo:country	▪ dbr:Germany
dbo:elevation	▪ 97.000000 (xsd:double)

Figure 3.2: DBpedia extraction process using the template for German locations.

Since Wikipedia infoboxes are generated by humans without compliance review, various inconsistencies can occur. One inconsistency is given by the format of a value. Although some attributes provide a recommendation, any arbitrary format can be inserted. In contrast, the templates strictly apply rules which can result in incorrectly extracted triples. Another inconsistency concerns the naming of the attributes. If the attribute name does not correspond to the name given in the template, a correct assignment of the property is not possible. However, the triple is nevertheless generated but inserted into the property namespace (dbprop:) instead of the ontology namespace (dbo:). Since it is not clear how to interpret the data, we will not include properties from the dbprop namespace. Altogether, the quality of the triple extraction depends on whether the infobox is properly build and on the quality of the template.

Knowledge bases usually organize their classes in class hierarchies. This means that super- and subclass relations can be established. All instances belonging to a subclass automatically belong to the superclass. Similarly, properties defined for the superclass can also be used by the subclass. The most upper class in the DBpedia class hierarchy is *Thing*, such that all instances are considered as things. Other knowledge bases provide varying class hierarchies, e.g., YAGO's class hierarchy is much more fine-grained than the DBpedia hierarchy while Freebase only covers a few classes.

Table 3.1: Frequent classes in DBpedia, version 2014.

DBpedia Class	#Instances
Agent	1 974 653
+ Person	1 649 646
| − Athlete	336 091
+ Organisation	302 657
| − EducationalInstitution	51 943
Place	816 837
| − Country	3 831
Work	425 044
+ MusicalWork	193 205
+ Software	31 737
Species	261 435

The DBpedia knowledge base (English version 2014[2]) describes $4\,584\,616$ instances using $2\,795$ different properties and 685 classes. Table 3.1 shows frequent classes from the first three levels of the DBpedia class hierarchy ('+': second level, '‖-': third level). Almost half of all instances are of type *Agent*. Other frequent first level classes are *Place*, *Work*, and *Species*. With about 75%, these four classes cover most of the instances in DBpedia. The distribution of instances per class provides an overview of the topics covered by the knowledge base.

In the following chapters, we will use DBpedia as knowledge base. Though, it is as an example of any knowledge base that covers information in a similar way. If not denoted otherwise, we will shorten the common namespace prefixes of DBpedia with *db:* and use the prefix *dbo:* for the DBpedia ontology namespace.

[2]`http://wiki.dbpedia.org/data-set-2014`

Chapter 4

Web Tables

Over the last years, web content in form of tables has gained an increasing attention among researchers. Compared to unstructured text, web tables provide a structured format which is beneficial for the interpretation of the data. Thus, web tables have been taken into account for applications like query answering, table search or knowledge base augmentation. At first, web tables have to be extracted from web pages. Until 2009, only big search engine companies owned large collections of web pages. With the introduction of publicly available web corpora like the Common Crawl, also researchers outside these companies have the chance to participate in the extraction of web tables.

In this chapter, we introduce web tables as a data source together with all necessary extraction processing steps. More precisely, we are interested in relational web tables that cover a set of entities described by attributes. Relational tables are a type of so-called genuine tables: tables in which rows and columns are syntactically and semantically coherent [Penn et al., 2001]. Since a majority of tables, about 99% according to [Cafarella et al., 2008a], is used for layout purposes, a distinction between genuine and non-genuine tables is required. This is followed by the metadata recovery, in which necessary information about the table is collected. The metadata recovery includes the task of determining in which column the names of the entities are specified, to give an example. Altogether, the relational web table extraction process consists of the crawling of web pages, the extraction of tables from web pages, the classification into different categories of tables, and the metadata recovery. By applying the relational web table extraction process to the Common Crawl web corpus, we generate the WDC Web Table Corpus 2012. It is the first publicly available relational web table corpus covering about 150 million relational web tables. With such a corpus, it is possible to analyze the characteristics of web tables and get an impression which topics are covered.

In Section 4.1, we describe the foundations of the relational web table extraction process, followed by the description of how we generated the WDC Web Table

Corpus 2012 and its successor the WDC Web Table Corpus 2015 in Section 4.2. Existing methods for all steps of the process are presented in Section 4.3. Finally, the last section (Section 4.4) compares the characteristics of the WDC Web Table corpora to other web table corpora.

The extraction of the WDC Web Table Corpus 2015 as well as parts of the statistical description have been published in [Lehmberg et al., 2016]. I performed the extraction of the WDC Web Table Corpus 2015, co-created the statistics, and provided the description of the dataset.

4.1 Relational Web Table Extraction

In this section, we first provide the definition of web tables and especially of relational web tables. Afterwards, we explain the relational extraction process that needs to be performed to come from a web page to a relational table. It includes steps like the detection of the table on the page or the table classification. At the end of the extraction process, a corpus covering relational web tables annotated with all information required for subsequent tasks like the matching is available.

4.1.1 Definitions

Tables provide a two-dimensional structure which enables a compact visualization of data [Zanibbi et al., 2004]. They are used in relational databases and spreadsheets or occur in documents like publications or on web pages. Each individual table type poses its own characteristics and challenges. While tables in relational databases are set up for automatic processing, tables in documents are designed for human readers. Thus, document tables can give a more detailed view on a topic but at the expense of a reduced automated understanding. In this thesis, we focus on a specific type of document tables: web tables. According to [Lautert et al., 2013], web tables are defined as follows:

Definition 4.1 *(Web Table) 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.*

Table 4.1 shows the structure of a table with x rows and y columns. Each intersection between a row and a column determines a cell $c_{i,j}$ with the value $v_{i,j}$ where $1 \leqslant i \leqslant x$, $1 \leqslant j \leqslant y$. A value can be a number or string but it can also be empty, contain a set of values or even another table.

Not all tables found on web pages actually contain valuable content. Many of them are only used for layout purposes, for example they serve as navigation on web pages. The tables we are interested in are *genuine web tables*. In genuine web tables, the values are simple structures, e.g., do not include other tables, and the rows and columns are syntactically and semantically coherent [Penn et al., 2001].

$v_{1,1}$	$v_{1,2}$	...	$v_{1,y}$
$v_{2,1}$	$v_{2,2}$	...	$v_{2,y}$
...	...	...	...
$v_{x,1}$	$v_{x,2}$	...	$v_{x,y}$

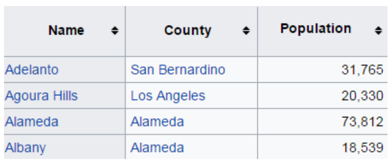
Table 4.1: Structure of a table.

One type of genuine tables are *relational web tables*, also referred to as horizontal web tables [Lautert et al., 2013]. Relational tables base on the relational model that has been introduced by [Codd, 1970]. Following the relational model definition together with the terms given by [Abiteboul et al., 1995], we define a relational web table as follows:

Definition 4.2 *(Relational Web Table) A web table is relational if each row provides data about specific objects, called entities, and the columns represent attributes that describe the entities.*

	attributes		
entities	$v_{2,2}$	...	$v_{2,y}$
	...	...	...
	$v_{x,2}$	...	$v_{x,y}$

Table 4.2: Structure of a relational table.



Name	County	Population
Adelanto	San Bernardino	31,765
Agoura Hills	Los Angeles	20,330
Alameda	Alameda	73,812
Albany	Alameda	18,539

Figure 4.1: Example relational table about cities.

According to the definition, we assume that exactly one concept is mentioned per table, i.e., a table describes countries only rather than countries as well as football clubs. Another assumption we make is that attributes are binary: they relate a single entity to a value. Table 4.2 shows the structure of a relational web table and Figure 4.1 depicts an example relational web table about cities which follows this structure. While the first row contains the attribute names, also named attribute label row, the first column covers the entity names (entity label column). Both - the attribute label row and the entity label column - refer to exactly one row and column. Further, as defined by the relational model, each row represents exactly one entity which is described by a set of attributes. In the example, each row covers one city with the attributes country and population.

4.1.2 Extraction Process

Extracting relational tables from the Web requires a series of steps as illustrated in Figure 4.2: Crawling, Extraction, Classification, and Metadata Recovery.

Crawling We refer to the process of collecting web pages as crawling. A common strategy is to use a list of seeds web pages which contain hyperlinks.

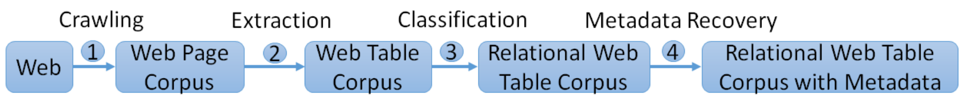


Figure 4.2: Steps of the relational web table extraction process.

The crawler follows these links, and where it finds further links, the procedure is repeated. There are different selection strategies that describe which links to follow. Choosing the selection strategy has a strong influence on the resulting corpus [Dhenakaran and Sambanthan, 2011].

Extraction During the extraction, the tables are detected and their content is extracted. For tables embedded in HTML using markups, the extraction is straightforward: whenever an HTML table tag <*table*> is detected, everything between the opening and closing tag is considered as table. Further, the HTML tags for table row <*tr*> and table data <*td*> indicate the beginning of the next row and value, respectively. The HTML tag <*th*> which defines the header of a table occurs too rarely such that we are not considering this tag. An example of a table embedded in HTML is depicted in Listing 4.1. Each line starts with the according tag for a new row and each value is surrounded by the table data tag.

Listing 4.1: Example of a web table embedded in HTML.

```
<table>
<tr><td>Name</td><td>Country</td><td>Population</td></tr>
<tr><td>Adelanto</td><td>San Bernardino</td><td>31,765</td></tr>
<tr><td>Agoura Hills</td><td>Los Angeles</td><td>20,330</td></tr>
<tr><td>Alamenda</td><td>Alamenda</td><td>73,821</td></tr>
<tr><td>Albany</td><td>Alamenda</td><td>18,539</td></tr>
</table>
```

Tables which are not embedded in HTML require different extraction methods. For example, a table in a PDF first needs to be identified and its boundaries need to be specified. If tables are represented in a textual format, visual features like white spaces or fonts are considered to separate rows and columns [Pinto et al., 2003].

Classification To distinguish between different types of web tables, various classification schemes have been proposed. The schemes range from a binary classification [Cafarella et al., 2008a, Limaye et al., 2010, Yakout et al., 2012] to a taxonomy with multiple types [Crestan and Pantel, 2011]. A scheme that extends the binary classification but still restricts the complexity is shown in Figure 4.3 and has been introduced by [Eberius et al., 2015]. The binary classification is indicated by a different color, i.e., the binary classification decides whether a table is relational or non-relational. On the first level of the classification scheme, web tables are divided into layout and genuine tables.

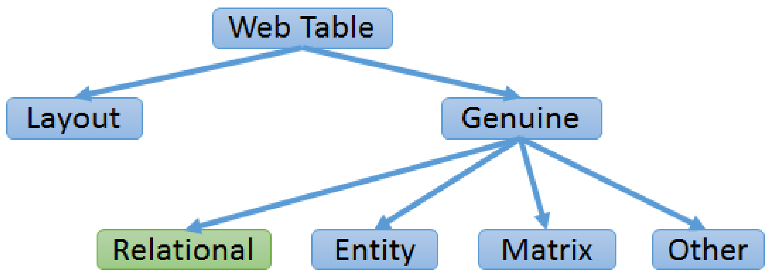
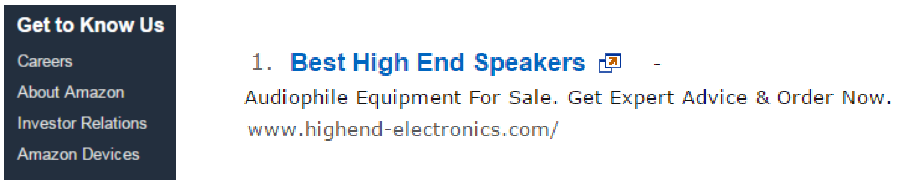


Figure 4.3: Web table classification scheme according to [Eberius et al., 2015].

- **Layout tables** are strictly used for layout purposes and do not contain any relational content. Two examples of layout tables are shown in Figure 4.4. Mostly, they are used to visually organize elements or present a navigation.
- **Genuine tables** are any kind of two dimensional tables that include simple cells, e.g., they do not contain complex structures, and the rows and columns are syntactically and semantically coherent [Penn et al., 2001].
- **Relational tables** include a set of entities described by attributes.
- **Entity tables**, also called attribute-value tables, cover information about one entity but the entity itself is not mentioned in the table. An entity table providing information about the city San Francisco is given in Figure 4.5. Within the table, the name of the entity is not stated but it can be derived from the context. A use case for entity tables is the extraction of information about products [Petrovski et al., 2014].
- **Matrix tables** are mostly used to provide statistics. An example of a matrix table is shown in Figure 4.6. The table indicates the demographic profile of groups of people and how it changed over time. In the table itself, no information is given about the content. On the one hand, it is not stated that the table is about groups of people in California and on the other hand it is not specified that the values represent the relative percentage of each group. Analogously to entity tables where the entity name is not mentioned, matrix tables often do not contain the name of the attribute(s).
- **Other genuine tables** contain valuable information but do not fit to the given types. As example, a web table containing a list of all U.S. states covers useful information but is not a relational table since no attributes are available.

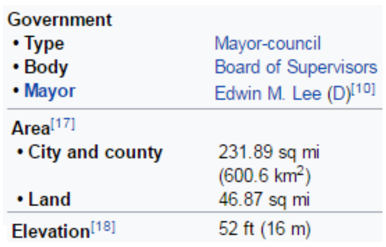
In general, all genuine tables can be converted to relational tables, e.g., an entity table can be transposed to represent a relational table with one entity. However, a lot of effort is potentially necessary to transpose data into a relational format. In the subsequent chapters, we strictly focus on relational tables.



Get to Know Us
Careers
About Amazon
Investor Relations
Amazon Devices

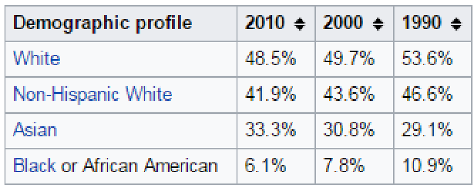
1. Best High End Speakers -
Audiophile Equipment For Sale. Get Expert Advice & Order Now.
www.highend-electronics.com/

Figure 4.4: Example layout tables extracted from *amazon.com*.



Government	
• **Type**	Mayor-council
• **Body**	Board of Supervisors
• **Mayor**	Edwin M. Lee (D)[10]
Area[17]	
• **City and county**	231.89 sq mi (600.6 km^2)
• **Land**	46.87 sq mi
Elevation[18]	52 ft (16 m)

Figure 4.5: Example entity table about the city San Francisco.



Demographic profile	2010	2000	1990
White	48.5%	49.7%	53.6%
Non-Hispanic White	41.9%	43.6%	46.6%
Asian	33.3%	30.8%	29.1%
Black or African American	6.1%	7.8%	10.9%

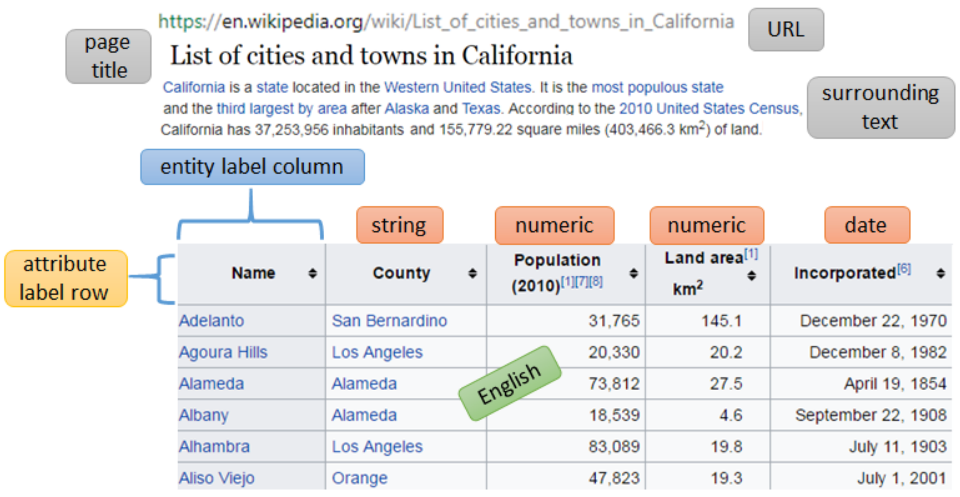
Figure 4.6: Example matrix table about the demographic changes in California.

Metadata Recovery In contrast to sources like databases, the tables lack any formal metadata or explicit schema [Cafarella et al., 2008a]. Depending on the use case, different metadata information is nevertheless required. To match web tables to knowledge bases, the entity label column, the attribute label row, the column data types, and language information are necessary metadata. Detecting the entity label column and attribute label row is also referred to as functional analysis as it describes the process where the function of the cells and their abstract logical relationship is identified [Göbel et al., 2012]. Additional contextual metadata including the URL of the web page, the page title, and the text surrounding the table are extracted. As we will discuss in Chapter 7, the context can be valuable to match web tables to a knowledge base.

Example 4.1 Figure 4.7 shows the recovered metadata. In this example, the entity label column is the first column, covering the names of the cities. Analogously, the first row contains the attribute labels. The remaining columns are of type string, numeric and date. Further, the table covers content in English. The URL, page title and text above the table are considered as contextual information.

Another type of metadata is the orientation of the table since a table can either be vertically or horizontally arranged [Galkin et al., 2015]. However, as we found out, only about 6% of the relational tables are vertically arranged such that we do not consider vertically arranged tables.

For all steps of the extraction process, different methods and strategies have been introduced. In the following section, we will show how the steps have been implemented to extract the WDC Web Table Corpora. Afterwards, Section 4.3 presents a general overview of approaches proposed in literature.



https://en.wikipedia.org/wiki/List_of_cities_and_towns_in_California

List of cities and towns in California

California is a state located in the Western United States. It is the most populous state and the third largest by area after Alaska and Texas. According to the 2010 United States Census, California has 37,253,956 inhabitants and 155,779.22 square miles (403,466.3 km^2) of land.

Name	County	Population (2010)[1][7][8]	Land area[1] km^2	Incorporated[6]
Adelanto	San Bernardino	31,765	145.1	December 22, 1970
Agoura Hills	Los Angeles	20,330	20.2	December 8, 1982
Alameda	Alameda	73,812	27.5	April 19, 1854
Albany	Alameda	18,539	4.6	September 22, 1908
Alhambra	Los Angeles	83,089	19.8	July 11, 1903
Aliso Viejo	Orange	47,823	19.3	July 1, 2001

Figure 4.7: Web table with annotated metadata: entity label column, attribute label row, column data types, language, and context.

4.2 Extraction of the WDC Web Table Corpora

In this section, we present the extraction of relational tables that results in the first publicly available web table corpus, the WDC Web Table Corpus 2012. The focus is not on inventing any new approaches but on reusing the methods that have been proposed in previous works. Besides the 2012 corpus, we also generated another version, the WDC Web Table Corpus 2015. Whenever changes to the extraction steps have been performed, we will report on them.

4.2.1 Common Crawl

Web crawling is the process of collecting data that is found in the Web. For many years, only the big search engine companies had access to large quantities of Web data, as they were the only ones possessing large web crawls. However, the situation has changed with the advent of the Common Crawl Foundation in 2009.[1] Common Crawl is a non-profit foundation that crawls the Web and regularly publishes the resulting web corpora for public usage. The crawler uses a list of seed URLs that are ranked according to their PageRank [Page et al., 1999]. The seed URLs for the 2012 Common Crawl Corpus have been gathered during previous Web crawls. Since the end of 2012, another list of seeds is used which is provided by the search engine company blekko [Lindahl, 2012]. With this changeover, the crawling quality should be improved, e.g., by avoiding spam or porn. All Common Crawl corpora are publicly available, provided as WARC files.[2]

[1] http://commoncrawl.org/
[2] https://www.iso.org/standard/44717.html

Table 4.3: Statistics of the 2012 and July 2015 Common Crawl corpora.

Dataset	#HTML pages	#PLDs	Size
2012 Common Crawl	3.5B	40M	101TB
July 2015 Common Crawl	1.81B	15M	145TB

Table 4.3 shows the statistics on the two Common Crawl corpora we use to extract web tables. The 2012 Common Crawl has been one of the largest corpora regarding the number of crawled pages. About 3.5 billion pages from over 40 million pay-level domains (PLDs) have been gathered. All web pages for which the same administrative authority is responsible for, belong to the same PLD. The second corpus, released in July 2015, comprises about 1.8 billion pages from 15 million PLDs. As indicated, a main reason might be the seed list changeover.

Besides the extraction of web tables, the Common Crawl corpora have also been used to gather other types of data. This includes Microdata and Microformats but also an isA database including hypernymy relations from text.[3] The Common Crawl corpora can be seen as a sample of the known or public Web [Meusel, 2017]. A bias towards the known parts of the Web is always given as soon as a selection strategy based on page rank is used. However, a representative subset of the whole Web is anyway not realistic since it would require a random access to every web page and even search engines are not able to index all pages in the Web.

4.2.2 Web Table Extraction

For our extraction, we assume that the web tables are embedded in HTML using designated markups. As described in the previous section, such an extraction is straightforward by using the HTML table tags. The web table extraction is integrated into the Web Data Commons (WDC) Extraction Framework[4] which focuses on distributed processing of web crawls [Mühleisen and Bizer, 2012]. Besides web tables, the framework is also used to gather web data like Microdata or web graphs. The WDC extraction framework runs in the cloud infrastructure environment Amazon Web Services (AWS).[5] The main components are the EC2 cloud instances that represent virtual servers and the S3 file system. Figure 4.8 shows the general workflow. The master node can either be a local server or a cloud instance. With WARC files as input, the following six steps are performed [Seitner et al., 2016].

1. The queue, AWS simple service queue (SSQ), is filled with all the documents from the corpus that should be processed.
2. A set of EC2 instances are launched by the master node.

[3]http://webdatacommons.org/
[4]http://webdatacommons.org/framework/
[5]https://aws.amazon.com

3. Each instance requests a document from the queue.
4. The requested document is downloaded from the file system S3.
5. The extraction is performed and the resulting data is uploaded to S3.
6. If the queue is empty, the master collects the data from S3.

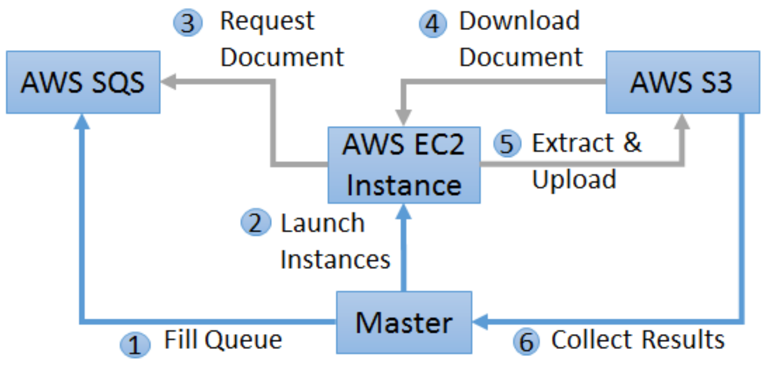


Figure 4.8: Overview of the web corpus extraction framework workflow.

Once the queue is filled and the instances are started, the extraction runs automatically. Only the collection of the data needs to be triggered by the user. In order to collect web tables, a web table extractor is integrated in step 5. The extractor parses each page using the HTML parser jsoup[6], detects the table tags and extracts the content. If a table is not an innermost but covers tables as values, it is discarded.

Table 4.4 presents the corpora statistics. For the WDC WTC 2012, over 10 billion innermost tables have been detected. Slightly less tables are contained in the 2015 corpus. Aggregated per PLD, this results in 3.4 and 5.75 innermost tables.

Table 4.4: Statistics of the WDC Web Table corpora.

Dataset	#Innermost tables	#Innermost tables per PLD
WDC WTC 2012	11.25B	3.4
WDC WTC 2015	10.24B	5.75

4.2.3 Web Table Classification

For the classification of the tables, two different methods have been applied: a binary classification for the 2012 corpus and a multiclass classification for the 2015 corpus. Thus, the WDC WTC 2015 also covers other types of genuine tables.

[6]http://jsoup.org/

WDC WTC 2012 Classification A binary classification into relational and non-relational tables has been applied. At first, all web tables that contain less than 5 cells or 3 rows are filtered out. The remaining tables serve as input for a classifier that decides if a table is relational. Therefore, a mixture of layout and content features is used that has been proposed by [Wang and Hu, 2002b].

Layout Features:

- Average number of columns and its standard deviation
- Average number of rows and its standard deviation
- Average cell length (number of characters) and its standard deviation
- Average cumulative length consistency which measures the degree of cell length consistency in the column

Layout features try to capture the structure of a table. Usually, relational tables cover at least a certain amount of rows and columns and show a consistency regarding the cell length. Further, the length of the cells in one column should be consistent, e.g., the cells of a column representing the capital of countries will roughly have a similar length with respect to the number of characters.

Content Features:

- Percentage of links, form, image, empty, digital, and text cells
- Cumulative type consistency

Content features do not consider the structure but the content of a table. Since relational tables are assumed to cover relational information, the cells are more likely to contain digits and text than links or images. Further, we expect columns in relational tables to mostly contain cells of the same type.

In order not to miss relational tables, we tuned the classifier, a decision tree, for recall at the cost of precision. An evaluation on a test set of about $10\,000$ tables results in a precision of 0.58 with a recall of 0.62.

WDC Web Table Corpus 2015 Classification The classification of the web tables from the 2015 Common Crawl is based on the classification that has been used to create the Dresden Web Table corpus (DWTC).[7] In contrast to the previously described classification, a multiclass classification is applied. As defined by the classification scheme, other genuine tables like entity tables are detected. First, a set of heuristics is applied to discard most of the tables that are considered as non-genuine, e.g., all tables with less than 2 columns or 3 rows. The classification itself uses a mixture of global and local features [Eberius et al., 2015]. The global

[7] `https://wwwdb.inf.tu-dresden.de/misc/dwtc/`

features are layout and content features of the binary classification. In addition, local features are considered which operate on a subset of the table. More precisely, local features are computed for the first two rows and columns as well as the last row and column of a table since the first rows and columns usually cover the entity and attribute labels. Knowing where the entity and attribute labels are located provides an important indication for the table type.

Depending on the classifier, $81-87\%$ of the tables classified as genuine are actually genuine. The distribution of table types is as follows: 60% entity tables, 39% relational tables and 1% matrix tables. These numbers are in coincide with the ones reported by [Eberius et al., 2015].

Table 4.5 reveals the statistics about the two corpora after the classification. While the 2012 corpus covers 147 million relational tables, the 2015 corpus includes 90 million. The amount of relational tables among all tables in both corpora is about 1% which is in line with the results reported by [Cafarella et al., 2008b]. For the next step, the metadata recovery, only relational tables are considered.

Table 4.5: Statistics of the relational WDC Web Table corpora.

Corpus	#Relational Tables	%of all Tables
WDC Relational WTC 2012	147M	1.3
WDC Relational WTC 2015	90M	0.88

4.2.4 Metadata Recovery

During the metadata recovery, the formal schema of the web table is recreated. This includes the attribute label row, the column data type, the entity label column as well as the language detection. All methods have been manually evaluated on a set of 1 000 randomly chosen web tables. Practicing the same methodology as [Cafarella et al., 2008a], non-relational tables in the evaluation set are not considered. The evaluation provides a lower bound for the metadata recovery performance that we expect during the matching of web tables to a knowledge base.

Attribute Label Row Detection, also called header detection, determines in which row the names of the attributes are stated. To identify the attribute label row, we apply a simple-rule based header detection as it has been introduced by [Pinto et al., 2002]. Based on the assumption that web tables are made for human readers, we always chose the first row as header if it contains at least 80% non-empty cells. The precision of the attribute label row detection is estimated with 0.82, using the random sample of 1 000 tables.

Column Data Type Detection assigns a data type to each column. We distinguish between three different types: date, string, and numeric. Other types like

Listing 4.2: Pseudocode for the data type detection given the set of values V of a column C.

```
dateCount <- 0
numericCount <- 0
stringCount <- 0
for(value v in V) {
 if(v is date) {dateCount++}
 else if(v is numeric){numericCount++}
  else {stringCount++}
}
dataType($C$) = max(dateCount, numericCount, stringCount)
```

boolean or coordinates occur too rarely. Listing 4.2 depicts the method. For each value in a column, the individual data type is guessed. Since all values can be represented as strings, we employ a cascading strategy that tries to parse each value into the most specific type and if it fails, it continues with the next type. First, it tries to parse the value into a date, then into a numeric and in case of fail, the data type string will be assigned. Finally, the data type with the highest frequency is chosen. If a value is represented as list, all values are considered.

- **Date** To detect whether a value is of type date, we apply a set of patterns, expressed as regular expressions, that describe different date representations. As basis, we use a set of existing patterns[8] and extend it, resulting in 66 patterns. Besides the patterns, restrictions are included, e.g., a year needs to be between 0 and 2020. Temporal expressions like "last Wednesday" are, in contrast to other works [Schilder and Habel, 2001], not detected as dates.
- **Numeric** Numerical values in web tables do not only cover numbers but also decode units of measurement. To know whether a number is represented in a unit is important for two reasons: first, a value like "5sqmi" should be detected as numerical value and second, knowing that the value is given in square miles is necessary to properly compare values. Thus, before parsing the numeric value itself, a unit detection is performed. We generated a catalog of 23 quantity types with 164 different units. An example of a quantity type is "area", covering units like "square kilometer" or "acre". Each quantity type comes with a canonical unit and according conversion factors. Further, for each unit a set of alternative names, abbreviations, and symbols is available. For example, for the unit "square miles", the abbreviation "sqmi" and the symbol "mi^2" is listed. Additionally, the catalog also includes dimensionless quantities like billions and currency names. Using the abbreviations and symbols in the catalog, we check whether a unit is mentioned either in the attribute label or in a cell. If a unit is detected in the attribute label, we assume that all values in the column are represented by the

[8]http://balusc.omnifaces.org/2007/09/dateutil.html

unit. All values belonging to a unit are converted into the according canonical unit. As example, the value “5sqmi” will be turned into “12.95” which is the value given in square kilometers. After the unit detection, we use the Java Double parser[9] to see whether the value is representable as a floating-point number. The parser also considers representations with exponent parts and hexadecimal digits.

Table 4.6: Results of the column data type detection method.

	String	Numeric	Date	All
Precision	0.87	0.94	0.74	0.89
Proportion	0.64	0.36	0.02	1.00

Table 4.6 shows the results of the data type detection. The overall precision on the set of 1 000 randomly chosen web tables is 0.89, varying between 0.74 for dates and 0.94 for numerics. Since only three data types are recognized, the types of some columns cannot be detected correctly. This includes telephone numbers or periods of time. For date detection, the performance is the lowest but has only a slight effect since only 2% of all columns are of type date. In contrast 64% are of type string and 36% of type numeric.

Entity Label Column Detection The entity label column, also called subject column or key column, contains the names of the entities and in turn serves as natural key. In contrast to keys as they are defined in databases, the entity label column can contain duplicate labels. Although it does not necessarily reflect the reality, we assume each relational web table to have exactly one entity label column. Listing 4.3 illustrates the algorithm to determine the entity label column. The column needs to be of data type string and its values are required to be not too long and not too short. If the according attribute label contains either the term “name” or “title” and the uniqueness rank, see Equation 4.1, is high enough, the column is chosen. Since the header is not always available and arbitrary terms can be used, only relying on the header results in insufficient performance.

Although the entity label column can cover duplicates, the uniqueness of the values is a strong indicator whether a column contains entity labels. The computation of the uniqueness rank is shown in Equation 4.1. For each column, the amount of unique values is taken into account under consideration of empty values.

$$uniquenessRank(c) = \frac{\#uniqueValues}{\#values} - \frac{\#emptyValue}{\#values} \tag{4.1}$$

The presented detection results in a precision of 0.84 and a recall of 0.76 on the random sample of 1 000 tables.

[9] `https://docs.oracle.com/javase/6/docs/api/java/lang/Double.html#valueOf%28java.lang.String%29`

Listing 4.3: Pseudocode to find the entity label column given the set of columns C of a table.

```
for(column c in C) {
 if(data type(c) == string) {
  if(∅ value size ⩾ 3.5 && ∅ value size ⩽ 200) {
   if(header(c) contains ``name'' or ``title'') {
    if(uniquenessRank(c) ⩾ 0.3) { return c }
   }
   else {
    if(uniquenessRank(c) == max(uniquenessRank(C)) {
     if(uniquenessRank(c) ⩾ 0.3) { return c }
    }
   }
}}}
```

Language Detection Depending on the use case, only tables of a certain language need to be considered. For example, if we match the tables to the English version of a knowledge base, it does not make sense to keep tables covering non-English content. To determine English tables, we apply a simple heuristic: we only keep web tables that have been extracted from web pages of the following top-level domains (TLDs): *com*, *org*, *net*, *eu*, and *uk*. By using the language detection, about 10% of the web tables do actually not cover content in English.

Beside the web table itself, the web page on which the table has been found is stored during the extraction. Thus, additional information like the URL of the web page, the page title or the words surrounding the table are provided. After the metadata recovery, a relational web table corpus annotated with metadata is available. Together with a knowledge base, it serves as input for the matching.

4.3 Related Work

In this section, we present the related work for all steps of the relational web table extraction process. The main focus lies on the classification and metadata recovery since they pose the main challenges.

4.3.1 Web Crawling & Table Extraction

For many years, only search engine companies had access to large web crawls. Due to its sheer size, the Web provides challenges regarding its scalability and efficiency [Cambazoglu and Baeza-Yates, 2015]. Due to efforts of initiatives like the Common Crawl or the Lemur project[10], web crawls are now publicly available.

[10]http://www.lemurproject.org/

The Lemur project released two large web corpora, one in 2009 and one in 2012. The ClueWeb09 corpus covers over one billion pages in ten languages with a compressed size of 5 tera-byte. In contrast, the ClueWeb12 corpus focuses on English web pages resulting in 700 million pages. Thus, both corpora cover less pages than the web corpora provided by the Common Crawl. The Altavista dataset[11] published by Yahoo! contains URLs and hyperlinks for over 1.4 billion public web pages indexed by the Yahoo! AltaVista search engine in 2002. However, not much is known about the strategy how the pages have been crawled and only the URLs and not the pages themselves are available. In contrast to crawl a corpus by oneself, we have to rely on the provider that publishes the corpus.

Extracting tables from the Web can be very challenging if not only HTML tables but also tables represented in a textual format are considered. In this case, the first step is to detect the table. Common features for the table detection are separators like white spaces or text associated with tables, see [Zanibbi et al., 2004] for an overview. By uniquely considering tables embedded in HTML, the table detection is straightforward by searching for the HTML table tag. This strategy is applied by most approaches, starting from the first introduction given into this topic by [Hurst, 1999]. One exception is the work of [Gatterbauer et al., 2007] who use a model of the visual representation of web tables as they are rendered by a web browser. However, since billions of web tables can be extracted by using the HTML tags, we consider this strategy as sufficient.

4.3.2 Web Table Classification

Classifying web tables into table types, e.g., relational and non-relational or genuine and layout depending on the classification scheme, is essential since the presence of a HTML table tag does not necessarily indicate the presence of a table covering information [Hurst, 2001].

As one of the first, [Chen et al., 2000] propose a set of heuristics and cell similarity measures to distinguish between genuine and layout tables. The heuristics are straightforward: all tables with less than two cells and tables with too many hyperlinks, forms, and images are considered as layout tables. To evaluate the performance of the approach, a test set with about 3 000 tables including airline information gathered from the Chinese Yahoo! website has been created. In this test set, about 29% of the tables are genuine which is far more than expected for a random sample of tables found on the Web. Applying the simple filtering rules, 98.9% of the genuine tables are classified as such. However, about 20% of the layout tables cannot be excluded by the rules. For the remaining tables, a set of cell similarity metrics is introduced. Depending on the data type, a similarity metric is applied to measure if neighboring cells are similar. If the percentage of simi-

[11] `https://webscope.sandbox.yahoo.com/catalog.php?datatype=g`

lar neighboring cells is above a threshold, the table is considered as genuine. The method results in an F-measure of 0.87. Altogether, the simple approach achieves reasonable results although the test set is restricted on one topic and does not represent the distribution between genuine and layout tables in the Web.

Another set of heuristics has been suggested by [Penn et al., 2001]. A web table is considered as genuine if it fulfills the following requirements: it is an innermost table, has more than one row and more than one column, the cells do not cover lists, frames, forms or images but have at least one non-text-level formatting tag and the length of the cells is less than a certain threshold. The method has been evaluated on a set of 75 tables originating from news provider, radio, and corporate sites. Overall, an F-measure of 0.88 is measured. Similarly to [Chen et al., 2000], the approach is evaluated on a domain dependent set of web tables.

Machine Learning-based Methods [Wang and Hu, 2002a] are one of the first to introduce a method that relies on machine learning techniques. It uses the same set of layout features that we consider with additional word group features. Word group features follow the idea that similar words are mentioned in genuine tables which in contrast do not occur in layout tables. The computed features serve as input for decision trees and support vector machines (SVM). A test set with 1 383 web tables has been generated by querying Google with key words like "table" or "weather". As result, an F-measure of 0.99 is reached where only slight differences regarding the learning algorithm can be determined. Moreover, it is concluded that the word group features have only a small influence on the result. To enable comparison to previous work, the method by [Penn et al., 2001] has been applied on the same data, resulting in an F-measure of 0.87. Thus, the machine learning approach outperforms the heuristics on a set of cross-domain web tables but the data is biased towards the used key words.

[Cafarella et al., 2008a] combine the ideas of previous works: First, heuristics are applied to filter out the most obvious layout tables: tables with less than two rows or columns, tables embedded in HTML forms, and tables presenting calendars. As analyzed on a web table corpus covering more than 14 billion tables, these obvious layout tables account for 90% of all tables. After excluding these layout tables, a rule-based classifier is learned with features that are similar to the layout and content features proposed by [Wang and Hu, 2002a]. Tuning the system towards recall results in an F-measure of 0.76 on a set of thousands of randomly chosen web tables. On the one hand, they are the first to apply the classification on millions of web tables and on the other hand, they evaluate the performance on a random set of tables providing a more realistic view on the classification task.

The WDC WTC 2012 classification uses the idea of [Cafarella et al., 2008a] to first filter out all obvious layout tables and afterwards to apply a machine learning approach taking the features proposed by [Wang and Hu, 2002a] as input.

Multiclass Classification Methods Besides the binary decision, [Crestan and Pantel, 2011] introduce a multiclass classification consisting of 12 table types. Thus, a classification approach needs to be able to distinguish between the different types. However, the decision if a table is genuine or not is similar to the binary classification. First, a filtering is applied which discards all tables with less than two rows, less than two columns, and tables with cells including more than 100 characters. On a random sample of 200 tables from their web table corpus, 93% of the excluded tables are indeed layout tables. For the more fine-grained classification into the different genuine types, gradient boosted decision trees are used. As input, global and local layout features, content features as well as lexical features are provided. For tables we consider as relational, an F-measure of 0.72 is stated. [Lautert et al., 2013] extended the classification scheme such that it covers tables that have not been considered before, e.g., nested tables.

[Eberius et al., 2015] also introduce a multiclass classification strategy with global and local layout and content features. While the classification into genuine and non-genuine tables achieves about 0.9 F-measure, the F-Measure regarding the individual genuine table types varies between 0.22 for matrix tables and 0.9 for relational tables. The evaluation has been performed on about 25 000 tables including 2 000 genuine tables. For WDC WTC 2015 corpus, the multiclass classification by [Eberius et al., 2015] is reused.

4.3.3 Metadata Recovery

For each type of metadata to be recovered, a set of methods has been proposed.

Attribute Label Row Detection One of the first and simplest techniques has been introduced by [Pinto et al., 2002]: take the first row of the table as header row. As we are dealing with web tables embedded in HTML, another method is to consider the HTML table header tag. However, only about 36% [Jung and Kwon, 2006] or 20% [Pimplikar and Sarawagi, 2012] of the tables actually contain a table header tag which is too less to rely on. Thus, approaches consider features coming from the HTML formatting but they do not rely on the table header tag, instead they take visual features like fonts, colors or spans into account [Limaye et al., 2010, Jung and Kwon, 2006]. Together with features that are computed on the table itself like the fraction of text cells, [Jung and Kwon, 2006] achieve an F-measure of 0.82. Other methods determine the header by learning how header rows look like. This can for example be accomplished with a context free grammar [Seth et al., 2010] or with rules that consider the amount of non-string data in the first row [Cafarella et al., 2008a]. Besides the features computed within the table, [Cafarella et al., 2008a] use an attribute co-occurrence statistical database to know whether the terms in a potential header row often occur together. Without the co-occurrence database, an F-measure of 0.81 and with the additional knowledge of 0.87 can be reached. An alternative approach by [Yoshida and Torisawa, 2001] learns from a set of examples which terms encountered in tables can be used to distinguish between header and non-header rows. As another background knowledge, [Wang et al., 2012] use the knowledge base Probase. Thus, the possible header row is queried and if suitable concepts are returned, it is said to be the header. With this strategy, they correctly identify 91% of the headers.

As already mentioned, our header detection heuristic is one of the most simple ones as described by [Pinto et al., 2002]. By using more sophisticated approaches that require external knowledge like co-occurrence databases, the results of the header detection can be improved. However, the increase in performance is rather small. On our evaluation set, a precision of 0.82 is achieved which is only a few percentage less as reported by more sophisticated methods. Further, the simple heuristic has the advantage that it can be efficiently applied on a web-scale corpus.

Data Type Detection Most commonly, regular expressions or patterns are applied to detect different data types. [Mulwad et al., 2013] use regular expressions to decide if values refer to objects or literals. A more detailed classification into data types like number, date or long text is introduced by [Zhang, 2016] where syntactic features such as the number of words in a cell or mentions of months or days are taken into account. [Kim and Lee, 2005] apply a set of patterns that results in more fine-grained data types like postal code or weight.

Regarding unit detection, [Sarawagi and Chakrabarti, 2014] build a quantity catalog called QuTree covering about 750 units. Besides the catalog, they use co-occurrence statistics between quantity types and mine units and phrases from headers in a table corpus. [Zhang and Chakrabarti, 2013] also uses information found in the header and a set of conversion rules that has been created by domain experts. For specific domains, ontologies of units have been generated using vocabularies like the ontology of units of measure.[12] While [Buche et al., 2013] provide a unit ontology for chemical risks, [Hignette et al., 2007] uses an ontology to detect units in tables from the microbiology domain, resulting in a precision of 0.98.

Our data type detection only includes the basic data types which cover most of the values. Since the checking of each value of a column is expensive when taking millions of web tables into account, we focus on patterns which are especially easy to verify. Similarly to [Sarawagi and Chakrabarti, 2014], we use a quantity catalog to be able to convert units. A qualitative comparison to previous work is difficult since other data types are used and/or the performances are not mentioned.

Entity Label Column Detection A simple heuristic for determining the entity label column is to assume that the left-most column contains the entity names [Pinto et al., 2002, Cafarella et al., 2008b]. [Venetis et al., 2011] extended the

[12] `http://www.wurvoc.org/vocabularies/om-1.8/`

heuristic by taking the left-most column that does neither contain numbers nor dates. With this rule, an accuracy of 0.83 can be achieved. Besides the simple rule, a classification approach is introduced that uses SVMs with 25 features like the fraction of unique cells, the average number of words but also the column index as input. This more sophisticated classification method results in an accuracy of 0.94. A related strategy has been presented by [Zhang, 2016]. In addition to features like the fraction of unique content in the cells, the header of the attribute is searched on the web page and a web search is performed for the values in each row. Both strategies help to find indications if a column might contain names of entities. An approach that considers the values itself instead of the column characteristics is described by [Yoshida and Torisawa, 2001]. From a set of example tables, the algorithm learns which terms are likely to appear as labels. Based on this classification, the most likely position for the entity label attribute is inferred. [Wang et al., 2012] queries the knowledge base Probase for the values of each attribute and checks if entities with the according labels are returned. An accuracy of 0.87 is reported on tables gathered from Wikipedia. Such methods are tied to the coverage of external source, in contrast to the methods relying on characteristics of the columns.

Similarly to the other entity label column detection methods, we use a set of simple heuristics as introduced by [Venetis et al., 2011]. Applying such heuristics results in a performance that is comparable to the results we obtain, around 85% correct assignments.

4.4 WDC Web Table Corpora

Many use cases can benefit from web tables but the main users have been companies like Google, Microsoft or IBM since they are in possession of web crawls from which web tables can be extracted. Unfortunately, they do not give public access to their data since the data is part of their business value. Thus, large amounts of web tables are not available to other researchers which on the one hand makes it impossible to perform web-scale experiments and on the other hand, the results reported by companies are not reproducible. Further, the characteristics and the topical coverage of web tables are also unknown such that the benefit for a particular use case cannot be estimated. For example, if the task is to collect information about schools but schools are rarely represented in web tables, it does not make sense to focus on web tables as primary data source.

To overcome this situation, we present the first publicly available web table corpus: the WDC WTC 2012 with its successor the WDC WTC 2015. All web tables mentioned in this thesis originate from the WDC WTC 2012 corpus. First, we provide statistics about the corpus, e.g., about the size of the tables. Further, we compare the corpus to existing corpora as far as possible. A topical analysis of the corpus will be provided subsequently in Chapter 5.

4.4.1 Statistical Analysis

In general, the WDC WTC 2012 & 2015 contain all web tables that have been extracted from the Common Crawl 2012 and July 2015, respectively. Since relational web tables are most valuable for the use cases we consider, the web tables in the WTC 2012 have been classified into relational and non-relational tables. The resulting set of relational web tables is a subset of the WDC WTC 2012, named *Relational WDC WTC 2012*. For these tables, the metadata has been recovered which means the attribute names and data types as well as the entity label column has been determined. For the matching, we require a web table to cover an entity label column and at least five rows and three columns. Further, since we are only considering the English version of a knowledge base, we expect the table content to be English. Thus, besides the relational subset, a second subset of web tables, called *Relational Mappable WDC WTC 2012* has been generated that contains all English relational web tables with an entity label column. While the statistical analysis of the Relational WTC 2012 provides general information about the characteristics of web tables, analyzing the subset indicates the characteristics of web tables expected for the matching. As introduced in the previous section, a multiclass classification into different types of genuine can be performed. Such a multiclass classification has been applied to the WDC WTC 2015, resulting in the *Genuine WDC WTC 2015*. The analysis of this corpus focuses on the distribution of genuine table types.

Relational WDC WTC 2012 All relational tables make up 147 million which corresponds to 1.1% of the tables found on pages from the Common Crawl 2012. Table 4.7 shows the basic statistics. About half of the tables are found on web pages belonging to the *.com* top-level domain. All other TLDs play a subordinate role. This is in line with the amount of web pages from the *.com* TLD in the corpus as shown by [Spiegler, 2013]. Due to the requirements defined for relational web tables, all tables cover at least 2 columns and 1 row. The maximum amount of columns found in tables is $2\,368$ with a maximum number of rows of $70\,068$. On average, a web table in the corpus contains 3.49 columns with 12.41 rows. However, most of the tables tend to contain less rows since the median of rows is only 6. While 15% of the tables only contain two rows, 48% of the tables only comprise two columns. In summary, the tables tend to be small and narrow.

After recovering the metadata, we can derive statements about the names of the attributes, headers, as well as about their data type. In Table 4.8, the top 10 headers are depicted. For almost 50% of the tables, we do not detect a header at all. According to our attribute label row detection, this means that the first row mostly contains empty cells. About 3% of all relational tables include at least once the header “name”. This is a typical name assignment for entity label columns but it does not provide any information about the described entities. Other commonly used headers like “price” or “5 star”, indicate that the corpus contains a large amount of tables about products.

Table 4.7: Table characteristics of the Relational WDC WTC 2012.

#Tables per Top 5 TLD				
com	de	org	net	co.uk
75M	8.6M	7.7M	7.3M	4.1M

	#Columns & #Rows			
	Minimum	Maximum	Average	Median
Columns	2	2 368	3.49	3
Rows	1	70 068	12.41	6

Table 4.8: Most frequent attribute labels in the Relational WDC WTC 2012.

Header	#Tables
NULL	45 691 861
name	4 653 155
1	3 755 432
price	3 706 006
date	2 728 291
5 star	2 505 161
title	2 121 028
artist	2 105 497
description	1 979 639
my price	1 689 805

Table 4.9: Distribution of the data types in the Relational WDC WTC 2012.

Column Data Type	%Columns
String	69.5%
Numeric	27.4%
Date	3.1%

The data type of a column determines which content can be expected. Table 4.9 presents the distribution of the data types. More than 69% of all attributes contain string values. The second most common data type is numeric with 27.4%. Dates are covered by only about 3% of all attributes.

Relational Mappable WDC WTC 2012 Altogether, 33 403 411 tables fulfill the requirements for matching: they are English relational web tables with an entity label column. The amount of relational mappable tables corresponds to one third of all relational tables from the WDC WTC 2012.

Table 4.10 shows statistics about the distribution of TLDs as well as the table sizes. Around 81% of all tables come from web pages of the *com* domain. By excluding TLDs on which we do not expect English content, the percentage of the tables coming from the *com* domain even raises from about 50% to 81%. Regarding the sizes of the tables, the average amount of rows is 21.5 with 4.1 columns while the median of rows is lower, only 10. In contrast to all relational tables, the tables of this subset tend to be wider and larger. One reason is of course the additional size restriction. Another reason is the entity label detection. If a table consists only of a few rows, it is more difficult to decide whether the table has an entity label column and if so which of the columns represent the entity labels.

Table 4.10: Characteristics of the Relational Mappable WDC WTC 2012.

#Tables per TLD				
com	org	net	eu	uk
26.7M	3M	3M	216K	6K

#Columns & #Rows				
	Minimum	Maximum	Average	Median
Columns	1	713	4.12	4
Rows	3	35 641	21.50	10

Compared to the relational corpus, the distribution of attribute headers only slightly changes. Similarly, only small deviations in the distribution of data types are visible. While the amount of columns of data type string is decreased by a few percentage, the number of numeric columns increases. Chapter 5 provides an in-depth analysis on this subset of the relational WDC WTC 2012.

Genuine WDC WTC 2015 With a more fine-grained classification, we can determine which type of genuine tables can be found most often. In Table 4.11, the distribution of the genuine table types of the WDC WTC 2015 is shown. Entity tables present the largest proportion of genuine tables, followed by relational tables. Contrary, matrix tables occur rarely which has also been reported by [Eberius et al., 2015]. Thus, besides relational tables, entity tables propose a source type with large amounts of information, expected to cover a broad topical coverage.

Table 4.11: Distribution of genuine table types of the WDC WTC 2015.

Type	#Tables	%Tables
Entity	139 687 207	1.40
Relational	90 266 223	0.90
Matrix	3 086 430	0.03
Sum	233 039 860	2.25

4.4.2 Comparison with other Corpora

A set of corpora covering relational web tables have been introduced but are mostly not publicly available. Table 4.12 provides an overview of the corpora with their sizes, the underlying web crawl, the amount of relational web tables, and the information about the public availability.

[13] approximated, 2.5% of 8.2 billion tables

Table 4.12: Overview of existing relational web table corpora.

Corpus	#Tables	Crawl	%Rel	Available
[Yakout et al., 2012]	573M	Bing	n/a	×
[Crestan and Pantel, 2011]	205M[13]	Yahoo	2%	×
WebTables [Cafarella et al., 2008a]	154M	Google	1.1%	×
WDC Relational WTC 2012	147M	CC 2012	1.3%	✓
WDC Relational WTC 2015	90M	CC 2015	0.9%	✓
DWTC [Eberius et al., 2015]	59M	CC 2014	2%	✓
[Pimplikar and Sarawagi, 2012]	25M	n/a	10%	×

The sizes of the corpora vary between 573 and 25 million tables. Except for the corpus provided by [Pimplikar and Sarawagi, 2012], the amount of relational tables among all extracted tables is about $1-2\%$. Regarding both, the absolute and relative amount of relational tables, the Relational WDC WTC 2012 is in the middle range indicating that the relational extraction methods are comparable. The largest corpora by [Yakout et al., 2012] and [Crestan and Pantel, 2011] as well as the smallest one by [Pimplikar and Sarawagi, 2012] are not publicly available and do not provide any further insights about the characteristics of the tables.

Among the other corpora, only the WebTables corpus introduced by [Cafarella et al., 2008a] performs a binary classification into relational and non-relational tables. Thus, it is the corpus that is closest to the Relational WDC WTC 2012. Figure 4.9 compares the sizes of tables in both corpora using the metrics that have been used to describe the WebTables corpus.

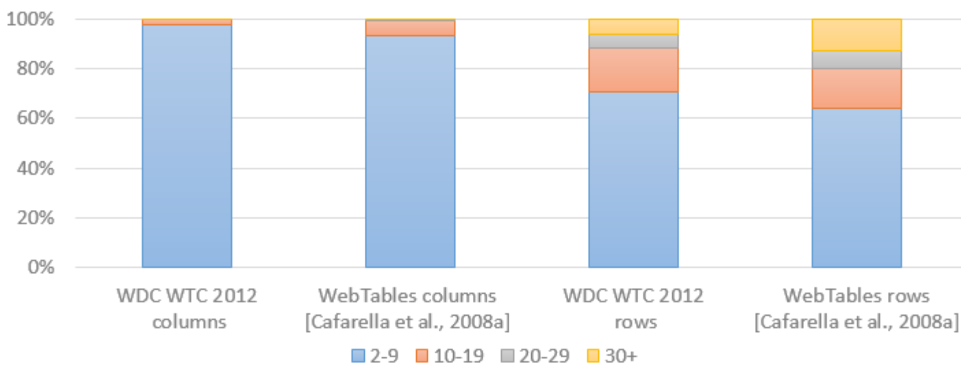


Figure 4.9: Comparison of table characteristics of the Relational WDC WTC 2012 and the WebTables [Cafarella et al., 2008a] corpus.

The number of columns and rows is divided into four categories: $2-9$, $10-19$, $20-29$ and $30+$. Most tables of both corpora cover between two and nine columns. Regarding the rows, 64% of the tables in the WebTables corpus include between two and nine rows which is a bit less than the 64% we can determine for the WDC corpus. For both categories in the middle, the amounts are alike. Tables with more than 30 rows can be found for 12% of tables in the WebTables corpus but only for 6% of the WDC tables. Despite of some fluctuations, the distributions of rows and columns are comparable.

Both other available web table corpora, the DWTC and the WDC WTC 2015, also base on Common Crawl datasets but use a multiclass classification scheme. To create the DWTC corpus, our WDC Extraction framework has been evolved. In turn, we build the extraction of the WDC WTC 2015 on the improved framework. Thus, the code-cooperation enabled us to easily apply the implemented multiclass classification for the WDC WTC 2015 corpus. Table 4.13 contrasts the minimum, maximum and average number of columns and rows for tables classified as relational. In contrast to the Relational WDC WTC 2012, the tables of both other corpora tend to be both, larger and broader. While an average table of the Relational WDC WTC 2012 covers 3.49 rows with 12.41 columns, an average table contains more than 14 rows with 5 columns in the other two corpora. One possible reason for the differences is the misclassification of entity tables as relational tables when applying a binary classification. As already indicated by [Crestan and Pantel, 2011], a misclassification of entity tables into relational tables is a common behavior which occurs in about 10% of the cases. Since entity tables usually only describe one entity, the number of rows and columns is less than in relational tables.[14] The amount of misclassifications will be lower in a multiclass approach since it explicitly includes features to distinguish between relational and entity tables. Regarding the attribute labels, the same most common labels can be detected, e.g., "date" or "name". Similarly holds for the column data types, i.e., columns covering strings are most common. Hence, the three corpora extracted from Common Crawl datasets show a compliant behavior. Regarding the distribution of different table types, both the WDC WTC 2015 and the DWTC report coincide statistics: more than 50% of all web tables are entity tables, followed by relational tables (40%). The remaining table types are negligible.

Table 4.13: Characteristics of web table corpora extracted from web pages of the Common Crawl.

		Minimum	Maximum	Average	Median
WDC WTC 2012	Columns	2	2 368	3.49	3
	Rows	1	70 068	12.41	6
WDC WTC 2015	Columns	2	18 106	5.20	4
	Rows	2	17 033	14.45	6
DWTC	Columns	2	7 291	5.79	-
	Rows	2	28 891	17.17	-

[14] http://webdatacommons.org/webtables/2015/entityStatistics.html

In summary, existing web table corpora show consistent characteristics regarding the absolute and relative amount of relational web tables. Since most of the corpora are not publicly available and no further statistics are stated, we cannot compare to them in detail. However, the distribution of rows and columns is compliant among the corpora for which these information are provided.

4.5 Conclusion

In this section, we gave a detailed overview of web tables as data source together with their extraction process. First, a corpus of web pages needs to be available to extract the web tables from the pages. Not all extracted tables contain useful content. Classifiers are applied to distinguish between layout and genuine tables. One type of the genuine tables are relational tables which describe a set of entities by a set of attributes. As last step of the extraction process, we described the metadata recovery. Its goal is to recover information like the entity label column, the attribute label row, the datatypes, the language, and contextual information. The metadata is required for subsequent steps like the matching. We further showed how the process has been applied to two Common Crawl corpora, resulting in the WDC WTC 2012 and its successor the WDC WTC 2015. The WDC 2012 has been the first publicly available web table corpus and presents the first contribution of this thesis. The following key statistics about our corpora are derived:

- From 3.5 billion web pages (Common Crawl 2012), 147 million relational web tables originating from 1.01 million PLDs can be extracted.
- Around 99% of the tables do not contain relational content.
- A relational table has as median of 3 columns and 6 rows.

We compared the characteristics of our web table corpora to existing corpora. The sizes of the tables and absolute as well as relative amount of relational tables that can be extracted is in line with results reported for other web table corpora, e.g., the WebTables corpus by [Cafarella et al., 2008a]. Further, we confirmed the finding by [Eberius et al., 2015] that the largest amount of tables are entity tables, followed by relational tables.

All web tables used for experiments in this thesis result from the WDC WTC 2012. A detailed topical profile of the WDC WTC 2012 will be provided in the next chapter (Chapter 5).

Chapter 5

Profiling the WDC Web Table Corpus

The preceding chapter introduced the first publicly available web table corpus, the WDC WTC 2012, and provided a statistical analysis. In this chapter, we analyze the tables originating from this corpus in more detail and present a topical profile with respect to the knowledge base DBpedia. Finally, we show the potential of web tables for the use case of filling missing values in DBpedia.

Web tables have already been applied in a variety of use cases like fact search [Yin et al., 2011] or knowledge base augmentation [Dong et al., 2014]. The used methods are either evaluated on small and thus not representative web table datasets or on corpora owned by search engine companies. For these corpora, information about the content and the coverage is not published. This makes it impossible to generalize and scientifically verify the research results. At the time we started with profiling, none of the existing publications answers the question which topical areas of the knowledge bases can be complemented using web table data. With the publicly available WDC WTC 2012, we are able to provide an in-depth profile of its contents. This profile can greatly benefit the research community by getting an impression for which use cases web tables are beneficial. The profile also serves as a common ground for the evaluation of knowledge base augmentation methods. In addition, we use the profile as foundation for the T2D gold standard (Section 6.3) to know which tables are included in a web table corpus.

In Section 5.1, we present the dimensions of data profiling together with the process we apply to create the profile of the web table corpus. The basic statistics as well as the topical, schematic, and data overlap are depicted in Section 5.2. Section 5.3 discusses both, profiles of other web data sources and existing works about knowledge base augmentation. Finally, Section 5.4 presents the potential of web tables for the use case of filling missing values in the knowledge base DBpedia.

The statistics of the WDC WTC 2012 as well as the profiling has been published in [Ritze et al., 2016]. I contributed to the implementation, execution, and analysis of the large-scale matching of the WDC WTC 2012, and to the evaluation of the data fusion.

5.1 Web Table Profiling

In this section, we introduce data profiling with the dimensions described by [Naumann, 2014]. These dimensions have been derived from the profiling of databases. First, we describe data profiling in general and afterwards show how we profile web tables with the use case of knowledge base augmentation in mind.

5.1.1 Profiling Dimensions

In general, data profiling is the process of examining the data and collecting statistics and information about it. The most basic form of profiling is to provide statistics about various counts like the number of values. The profile can either refer to one source or to a set of multiple sources [Naumann, 2014]. According to [Doan et al., 2012], especially when integrating multiple sources, profiling the overlap of the sources is essential.

Profiling multiple sources is classified into three dimensions [Naumann, 2014]:

- Topical Overlap: The topic that is described by a source. For two sources, a topical profile should be able to tell whether the sources topically overlap.
- Schematic Overlap: The degree to which source schemata complement each other and the degree to which they overlap.
- Data Overlap: The number of overlapping real-world objects that are represented in the sources.

Often, the profiling of sources is driven by a use case. For example, if missing values should be filled in a knowledge base, it is important that the sources share a schematic and data overlap with the knowledge base but at the same time the sources contain values for these schemata and real-world objects that the knowledge base does not cover. For other use cases, sources with a different profile are more promising.

5.1.2 Profiling Process

In this thesis, we focus on the profiling of web tables with the use case of filling missing values in a knowledge base. Thus, we are particularly interested in the overlaps of web tables with the knowledge base. Since a web table corpus covers millions of web tables from different domains, we face the profiling of multiple sources. The proposed dimensions of multiple source profiling are topical, schematic, and data overlap. They exactly correspond to the three matching tasks of matching web tables to a knowledge base: class, property, and instance matching. While the class and property matching present the schema matching as introduced in Chapter 2, the instance matching task refers to the data matching. Assigning a class to a table within the class matching task defines the topic of the table. Further, the property matching generates correspondences that present the schematic overlap of the web tables with a knowledge base and similarly performing the instance matching provides information about data overlap.

Figure 5.1 illustrates how our profiling process looks like. The first step is the matching of the web table corpus to the knowledge base. Therefore, we use T2K Match as it will be described in Chapter 6 together with the prefiltering and the most important features determined in Chapter 7. T2K Match performs all three matching tasks in an integrated fashion. As we will describe later, we expect a sufficient matching performance with an F-measure around 0.94 for the class, 0.7 for the property and 0.82 for the instance matching task, see Section 6.5 for details. As data sources, we use the Relational Mappable WDC WTC 2012 (Section 4.4.1) and the knowledge base DBpedia (Section 3.3). The Relational Mappable WDC WTC 2012 contains all tables from the WDC that are relational, provide English content and have an entity label column.

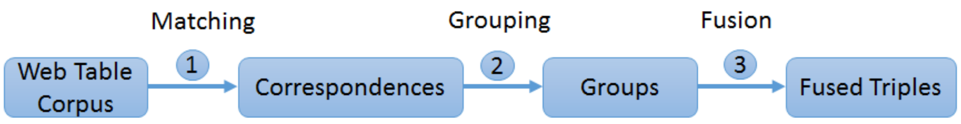


Figure 5.1: Profiling process of the WDC WTC 2012.

As result, the set of correspondences for all three matching tasks is created. Based on the correspondences, the overlaps indicate which topics are covered in the web tables as well as which data is described using which schemata. This can be useful to estimate whether web tables provide the information required for a certain use case. However, since more than one web table can contain the same information, the overlaps do not present the amount of facts that can actually be extracted from the tables. Hence, for use cases like knowledge base augmentation, the potential is not yet clear. Therefore, the grouping is performed which groups all triples describing the same fact. The groups can be generated based on the correspondences according to their instance-property combination. For example, all triples with <db:Germany> as subject and <dbo:populationTotal> as predicate are located in the same group. In turn, each group represents one fact and in turn the number of groups determines that amount of facts that can be extracted. Further, the groups indicate to which extent the web tables overlap among each other. Since the tables do not necessarily contain the identical values for the same instance-property combination, the fusion tries to overcome uncertain and conflicting values [Bleiholder and Naumann, 2009]. Thus, for each group the fusion strategy decides which of the available values is the best choice. Hence, the fused triples present the extracted facts and can in turn be used for slot filling if these facts are not yet included in the knowledge base. Further, based on the fused triples that are actually found in the knowledge base, we can estimate the quality of the generated triples.

The results of the profiling process until the fusion are presented in Section 5.2. These results are independent of the use case and provide a general overview of the corpus content. The fusion together with the potential of web tables for filling missing values in DBpedia is discussed in Section 5.4.

5.1.3 Representativity

By now, approaches using web tables as source have either been evaluated on small and thus not representative datasets or on corpora that are not publicly available. We state that the WDC WTC 2012 corpus is both, publicly available as well as representative. In turn, we claim that the statistics are not only valid for the corpus but can be seen as general statements about web tables. This assumption basis on the the following findings.

The Common Crawl corpora can be seen as a sample of the known or public Web [Meusel, 2017]. A bias towards the known parts of the Web is always given as soon as a selection strategy based on the popularity of a web page, e.g., page rank, is used. However, a representative subset of the whole Web is anyway not realistic since it would require a random access to every web page and even search engines like Google are not able to index all pages that exist in the Web. As shown by [Meusel, 2017] on the same Common Crawl corpora, since the crawl covers billions of web pages, the sampling error is expected to be very low, e.g., for the extraction of semantic annotations the sampling error is between 0.035% and 0.007% with a confidence level of 99%. Thus, we consider both WDC WTC as representative for web tables found on the Web. However, in contrast to semantic annotations, we cannot rely on the use of vocabularies to create the profile. As indicated, by matching the web tables to a knowledge base we can only find topical, schematic, and data overlap if the knowledge base covers the according classes, properties and instances. Thus, the created profiles can only be representative regarding the considered knowledge base.

5.2 Statistical Analysis

Besides the profile that actually considers the content of a web table, basic statistics about the web tables can already provide hints whether the data is suitable for a certain use case. For example, if the task is to extract release dates of films but dates are rarely covered in web tables, another source is more beneficial. In this section, we first provide statistics about the sizes of web tables in the corpus. To get an impression about the covered topics independent of the knowledge base, we depict from which web pages the tables have been extracted as well as present common attribute header. Afterwards, we present the topical, schematic, and data overlap as well as statistics about the generated groups that indicate the overlap between the web tables.

5.2.1 Table Size Distribution

In Chapter 4, we already provided basic statistics about the number of rows and columns of web tables in Relational Mappable WDC WTC 2012. In this part, we deepen the analysis on the value level.

Table 5.1: Statistics of the Relational Mappable WDC WTC 2012.

	#Columns, Rows, and Values				
	Numeric	Date	String	Average	Sum
Columns	46M	4M	86M	4.12	137M
Rows	-	-	-	21.50	716.6M
Values	995M	101M	1.9B	88.61	2.95B

Table 5.1 presents the number of columns, rows, and values in total and per data type. Columns without obvious data type are excluded. The number of values is approximated based on the data type of the columns and the corresponding number of rows. Most of the values are of data type *string*, followed by *numeric* values. On average, a web table in the corpus has about 4 columns and 22 rows which roughly results in 90 values per table. The total amount of values can be seen as upper bound for the amount of facts that can be generated. For example, if the web tables are used to construct a knowledge base, at most 3 billion facts can be generated, assuming that all entities and attributes are distinct.

5.2.2 Domain & Header Distribution

As we will discuss in more detail later, the topical profiling depends on the knowledge base to which the web tables are matched. If the knowledge base does not cover a certain topic, this topic will not be found in the profile. Thus, we first investigate from which PLDs the tables have been extracted and which attribute labels occur frequently. Both, the PLD as well as the attribute label, provide insights about the covered topics independent of a knowledge base.

The tables in our corpus originate from 97 932 different PLDs. Table 5.2 shows the most frequent PLDs and headers. The most prominent PLD is *apple.com* (iTunes Music) while the other PLDs often refer to sport websites, e.g., *baseball-reference.com* or retailers such as *amazon.com*. Besides the impressions we get about the covered topics, the list of PLDs can also be useful to find websites covering large amounts of tables about a certain topic. For example, if someone is interested in baseball, the website *baseball-reference.com* presents a beneficial source with thousands of tables. Frequently used headers are for example “5 star” and “price”, indicating that the corpus contains a large amount of tables about products. Further, headers like “replies” or “latest post” imply that the corpus contains data from blogs or forums. About 8.5% of all attributes do not provide a header.

Table 5.2: Most frequent PLDs and headers found in the Relational Mappable WDC WTC 2012.

PLD	#Tables	Header	#Tables
apple.com	50 910	no label	14 495 456
patrickoborn.com	45 500	5 star:	2 402 376
baseball-reference.com	25 647	name	1 813 064
latestf1news.com	17 726	price	1 771 361
nascar.com	17 465	date	1 603 938
amazon.com	16 551	amazon price	1 178 559
baseballprospectus.com	16 244	formats	1 066 836
wikipedia.org	13 993	title	9 132 60
inkjetsuperstore.com	12 282	time	856 401
flightmemory.com	8 044	description	773 883
sportfanatic.net	7 596	size	692 251
tennisguru.net	7 504	replies	605 075
windshieldguy.com	7 305	used from	589 278
donberg-electronique.com	6 734	new from	589 259
citytowninfo.com	6 293	year	579 726
juggle.com	5 752	location	546 856
deadline.com	5 274	album	526 375
blogspot.com	4 762	type	501 747
7digital.com	4 462	latest post	421 737
electronic-spare-parts.com	4 421	discussion	412 672

5.2.3 Correspondence Statistics

With the generated correspondences to DBpedia for all three matching tasks, we identify the topical, schematic, and data overlap. Table 5.3 shows statistics about the matched web tables with respect to their corresponding DBpedia class (not a complete list). T_0 is the set of tables for which at least one entity has been matched to DBpedia. T_c covers all tables with at least one property correspondence. V_c is the amount of cells (values) contained in tables of T_c. Thus, V_c expresses how many triples can be generated. These numbers are further divided according to their data type, depicted in the middle four columns. Lastly, the relative amount of tables (T_0) assigned per class and the according DBpedia distribution is depicted.

Table 5.3: Correspondence statistics of the Relational Mappable WDC WTC 2012.

DBpedia Class	Number of Tables/Values			V_c Data Type				Relative Amount	
	T_0	T_c	V_c	Numeric	Date	String	Reference	T_0	DBpedia
+ Person	265 685	103 801	4 176 370	2 117 793	1 588 475	266 628	203 474	27.97	35.72
| − Athlete	243 322	95 916	3 861 641	2 084 017	1 435 775	163 771	178 078	25.61	7.28
| − Artist	9 981	2 356	18 886	3	11 527	3 499	3 857	1.05	2.08
| − Politician	3 701	1 388	18 505	10	7 725	3 393	7 377	0.39	0.87
| − Office Holder	2 178	1 435	131 633	30	66 762	59 332	5 509	0.23	1.05
+ Organisation	194 317	36 402	573 633	99 714	187 370	100 710	185 839	20.45	6.55
| − Company	97 891	6 943	203 899	58 621	83 001	34 665	27 612	10.30	1.85
| − SportsTeam	50 043	2 722	31 866	2 206	22 368	43	7 249	5.27	6.04
| − Educational	25 737	14 415	238 365	38 056	64 578	13 334	122 397	2.70	1.12
| Institution									
| − Broadcaster	14 515	11 315	93 042	564	13 095	52 186	27 197	1.53	0.61
Work	269 570	127 677	2 284 916	109 265	1 354 923	33 091	787 637	28.38	9.20
+ MusicalWork	138 676	80 880	1 131 167	64 545	396 940	7 610	662 072	14.60	4.18
+ Film	43 163	9 725	256 425	10 844	198 913	14 382	32 286	4.54	1.89
+ Software	39 382	23 829	486 868	418	414 092	9 194	63 164	4.15	0.69
Place	133 141	24 341	859 995	413 375	273 510	84 111	88 999	14.02	17.68
+ PopulatedPlace	119 361	21 486	787 854	405 406	257 780	57 064	67 604	12.56	11.86
| − Country	36 009	6 556	208 886	93 107	66 492	31 793	17 494	3.79	0.083
| − Settlement	17 388	2 672	17 585	4 492	6 662	2 444	3 987	1.83	10.15
| − Region	12 109	427	5 625	3 097	897	292	1 339	1.27	0.47
+ ArchitecturalStructure	10 136	1 815	46 067	3 976	7 387	23 110	11 594	1.07	3.26
+ NaturalPlace	1 704	254	2 568	866	696	340	666	0.18	1.36
Species	14 247	4 893	83 359	-	7 902	38 682	36 775	1.50	5.66
Σ	949 970	301 450	8 037 562	2 751 105	3 437 420	536 526	1 312 511	100	100

Tables Altogether, 949 970 of 33.3 million web tables have a correspondence to at least one DBpedia instances (T_0). These tables have correspondences to a total of 361 different classes from the DBpedia ontology which represent about 50% of all DBpedia classes. Since all these tables contain real-world objects which can also be found in the knowledge base, they are potentially useful for set expansion [Wang and Cohen, 2008] which adds missing instances to the knowledge base. If we additionally require at least one property correspondence, we find 301 450 tables (T_c) which match altogether 274 different DBpedia classes. These tables are potentially useful for slot filling [Surdeanu and Ji, 2014] to add missing values to the knowledge base. Altogether, the tables contain a total of 8 million values (V_c) which either already exist in or might be new to the knowledge base. The observation that only 2.85% of all relational web tables in the corpus can be matched to DBpedia indicates that the topical overlap between the tables and the knowledge base is rather low, assuming that the matching step detected all correspondences. However, this is consistent with the analysis of the PLDs and attribute headers presented in the previous section: the web table corpus covers for example tables describing products which cannot be matched to DBpedia since products are rarely represented in DBpedia.

Classes To profile the topical overlap between web tables and DBpedia, we picked the most frequently matched DBpedia classes from the first levels of the DBpedia class hierarchy and provide detailed statistics in Table 5.3. Almost 50% of the web tables describe *Persons* and *Organisations*, followed by tables covering *Works*. It comes as no surprise that we find a large amount of correspondences for the class *Person*, as it is, apart from *Agent*, the most frequent classes in DBpedia (see the last two columns in Table 5.3). In contrast, about 28% of the web tables cover *Works* but only about 9% of all instances in DBpedia are *Works*. Hence, these classes are overrepresented in the web table corpus, compared to the topical distribution in DBpedia. However, it can also be the other way around. The second most frequent class in DBpedia, *Place*, is less frequent in the web tables, although it is almost twice as large as *Work* in the knowledge base. This either indicates that places are underrepresented in the web table corpus or that the matching framework has trouble detecting this class. Hence, the used knowledge base defines which topics can be detected, c.f., [Hassanzadeh et al., 2015], but depending on the particular topic, a different distribution can be found in web tables.

With respect to tables that additionally cover at least one property correspondence, we can further see that only 18% of the tables about *Places* have a property correspondence. Thus, beside of being underrepresented, we also find signs for a schema mismatch between the DBpedia ontology and the web tables.

Data Types In the web table corpus, the majority of values are of data type *string*, followed by *numeric* and *date*. Among the values in the matched tables T_c, the majority is now formed by *date*, followed by *numeric*, *string*, and *reference*. Columns are of data type *reference* if their attribute corresponds to an object property. In this case, the generated triple will refer to a real-world object at the object position instead of a string. As the *reference* type requires a matching step, these values appear as *string* in the statistics about the corpus. Reasons for the change in the distribution can be the following: Either the web tables have a tendency towards factual data, like dates and numbers, or the schema overlap between the tables and DBpedia consists mainly of properties with these data types. Another reason could be that the matcher allows for more variation in the values with these data types than for strings, resulting in more overall correspondences.

Instance Distribution In total, we find 13 726 582 instance correspondences for 717 174 unique instances, which corresponds to 15.6% of all instances in DBpedia. Figure 5.2 shows the complementary cumulative distribution function, also called tail distribution, of the fraction of instances (y-axis) that have correspondences in a given number of web tables (x-axis). From this figure we can see that around 70% of all instances are referred in more than one web table. 55% have three or more sources and 25% have at least ten sources. Thus, for more than two thirds of all instances, we find evidence in more than a single web table. Looking at the other end of the distribution, about 3% of the instances are described within more than 100 tables. In general, the more popular an instance is, the more sources we expect to find it in.

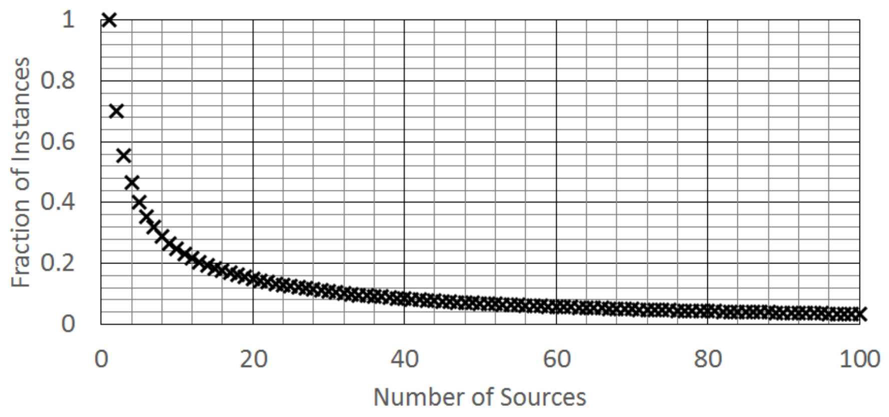


Figure 5.2: Distribution of instance correspondences.

Property Distribution Aggregated over all tables, we find a total of 562 445 property correspondences for 721 unique properties, about 25% of all properties in DBpedia. Figure 5.3 shows the tail distribution of the fraction of properties (y-axis) that have correspondences in a given number of web tables (x-axis). 88% of all properties have correspondences from at least two web tables. 81% can be found in three or more web tables and 60% of all properties have correspondences from at least ten web tables. About 30% of all properties have more than 100 correspondences.

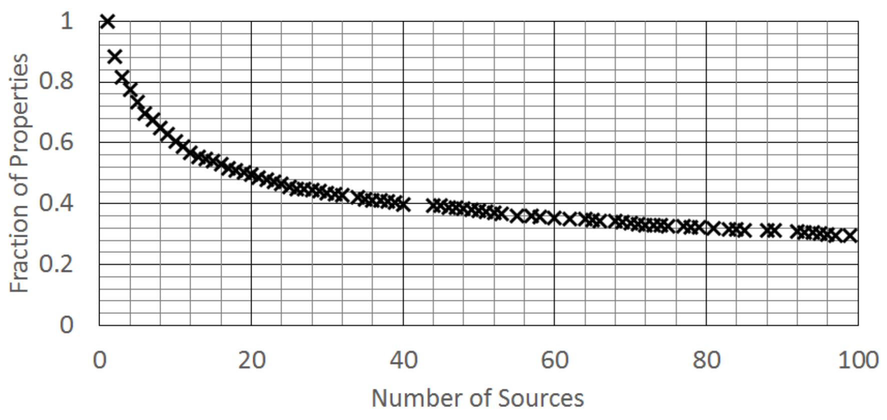


Figure 5.3: Distribution of property correspondences.

Table 5.4 lists some examples for frequent instances and properties for selected classes in order to give an impression of the detected correspondences. All of these instances are more or less commonly known, which is in line with the intuitive expectation that more popular instances are found more often. Nevertheless, also tables including long-tail entities can be found. For example, at least 200 tables describe racing horses, using the property *sire* to indicate the father of a horse.

Table 5.4: Examples for frequent instances and properties.

Class	Instance	#Corr.	Property	#Corr.
Athlete	Jeff Gordon	15 826	team	7 982
	Fernando Alonso	14 870	championships	4 464
Country	China	13 515	capital	965
	France	13 300	currency	508
Office Holder	John McCain	329	religion	74
	Barack Obama	328	vicePresident	66
Company	Toshiba	59 112	formationDate	1 016
	Nortel	45 573	iataAirlineCode	714
Musical	Can’t Help Falling in Love	1 403	releaseDate	60 473
	Hold It Against Me	1 801	musicalBand	27 832
Educational Institution	University of Phoenix	2 486	state	998
	Purdue University	2 325	numberOfStudents	707
Species	Great Egret	541	genus	3 706
	Rainbow trout	329	sire	207

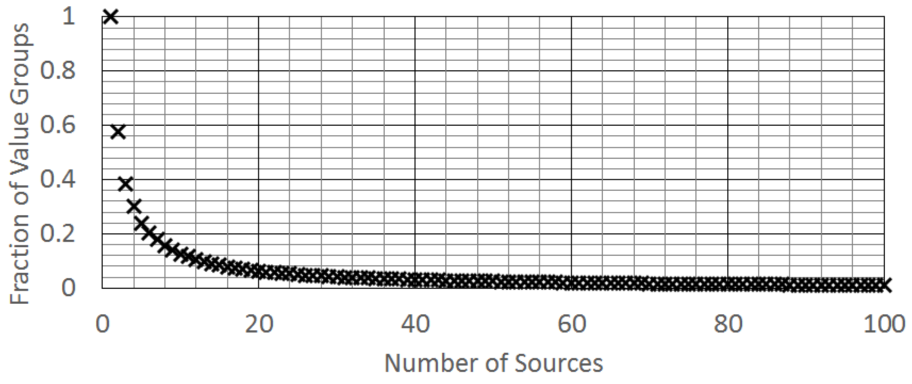


Figure 5.4: Distribution of group sizes.

5.2.4 Group Statistics

In addition to the overlap with DBpedia, we present statistics about the internal topical, schematic, and data overlap within the web table corpus. These statistics can be derived by creating groups as described in Section 5.1.2.

Group Size Distribution Out of the 8 million triples that we can generate from the web tables, 929 170 groups of triples can be formed. Figure 5.4 shows the tail distribution of group sizes. 58% of all groups contain triples from at least two sources, 39% from at least three sources. Triples from ten or more sources can be found for 13% of all groups. Very frequent groups, which are supported by at least 100 sources, constitute 1% of all groups. Assuming that the matching step found all correspondences, this distribution in combination with the low overlap between the web tables and DBpedia, which we observed earlier, shows that the web tables contain a wide range of different triples, but most of these triples are only provided by a small number of sources. Such triples are more likely to be new to the knowledge base, as we expect frequently stated triples to be already existing. However, these new triples come with a drawback: As they are only supported by few sources, it will be difficult for a fusion strategy to find the correct values in the groups. For 42% of the instance-property combinations only a single value is present (group size$=1$), such that a fusion strategy can just decide to accept or discard the value.

Classes Table 5.5 depicts the distribution of groups per selected class. The second column indicates the number of groups G that were formed for the respective class and the third column states the ratio of this number to the total number of triples (column V_c in Table 5.3). This ratio is high if we cannot group many triples for a class. If it is low, this means that most of the triples have been grouped. This can be the case if there is only a small number of instances belonging to a certain class. An example is the class *Country*. We can assume that the majority of the web tables with correspondences to *Country* are about the 200 countries that are commonly acknowledged today and not to all the historic countries which are also contained in DBpedia.

Table 5.5: Distribution of groups per class.

DBpedia Class	G	G/V_c
+ Person	366 048	0.09
| − Athlete	284 213	0.07
| − Artist	6 842	0.36
| − OfficeHolder	6 559	0.35
| − Politician	11 362	0.09
+ Organisation	87 527	0.15
| − Company	25 164	0.12
| − SportsTeam	2 453	0.08
| − EducationalInstitution	35 736	0.15
| − Broadcaster	21 687	0.23
Work	331 071	0.15
+ MusicalWork	201 186	0.18
+ Film	56 610	0.22
+ Software	33 552	0.07
Place	100 673	0.12
+ PopulatedPlace	71 981	0.09
| − Country	5 709	0.03
| − Settlement	1 879	0.11
| − Region	1 193	0.21
+ ArchitecturalStructure	17 697	0.38
+ NaturalPlace	12 037	0.47
Species	23 809	0.29
Σ	929 170	0.01

Data Types Figure 5.5 and Table 5.6 show the data type distribution at different stages of our data integration process. At first, we have the full web table corpus (*Corpus*). Afterward, we match the corpus (*Matched*) and finally group the generated triples (*Grouped*). As we already discussed the change in the distribution between the full corpus and the correspondences, we now focus on the transition from correspondences to groups, where all triples with the same instance-property combination are put together. The last column in Table 5.6 shows the ratio of this grouping process. We see that, on average, each group contains 8.46 triples. The largest group sizes can be observed for *numeric* triples, where on average 13.59 triples form a group. *Date* groups are also larger with about 9 triples per group. *String* and *reference* groups, however, are quite small with only about 4 to 5 triples per group. Figure 5.5 shows the number of triples per data type as proportions in each step. Here it becomes obvious how the large fraction of *string* values in the complete corpus is replaced by *date* and *numeric* when only considering tables that overlap with DBpedia. In the grouped stage, we see how the relative size of *string* and *reference* increases again, as many *date* and *numeric* values are grouped.

Table 5.6: Distribution of data types per step.

Data Type	Corpus	Matched	Grouped	Ratio
Numeric	995M	2 751 105	202 362	0.14
Date	101M	3 437 420	379 240	0.09
String	19 000M	536 526	86 330	0.04
Reference	0M	1 312 511	261 238	0.05
Σ	20 096M	8 037 562	929 170	0.08

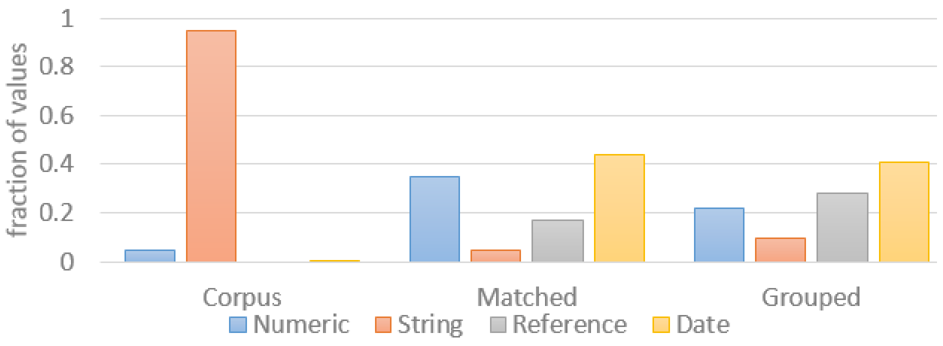


Figure 5.5: Distribution of data types aggregated by their steps.

5.3 Related Work

As one of the first, the database community started with profiling their data. Basic statistics like the distribution of values have been computed in order to estimate the costs of individual database operations [Mannino et al., 1988]. [Naumann, 2014] provides an extended classification scheme that includes the profiling of multiple sources. The classification scheme introduces the topical, schematic, and data overlap which are worth to consider especially for integrating data from various sources. All these dimensions are important to estimate the utility of sources for a certain use case. In this section, we describe the profiling of other web data sources like LOD or semantic annotations. Further, we introduce other profiles that have been generated for web table corpora. In the end, we present approaches that also focus on the use case of knowledge base augmentation.

5.3.1 Web Data Profiling

Since the Web provides an enormous amount of data, automatic methods are required to know which sources cover which content. Thus, the profiling of different web sources has been addressed in literature. This includes methods to determine topics of websites but also the schematic overlap of structured data sources like Linked Open Data or semantic annotations.

Web Page Profiling To get an estimate of the topical distribution of websites, [Chakrabarti et al., 2002] annotate websites with a topic taxonomy by applying a classifier that determines the topic based on the words that are found on the sites. Their findings show that most of the sites talk about *computers*, e.g., software manuals, followed by sites assigned with the topics *society* and *businesses*. For websites, the topical overlap is the only meaningful profiling dimension. In contrast, for structured data extracted from the Web, profiles that go beyond the topic overview have been generated.

LOD Profiling [Schmachtenberg et al., 2014] analyzed the topical and schematic overlap of datasets belonging to the LOD Cloud. Altogether, the LOD Cloud covers about 1 000 well-defined datasets containing descriptions of various entities. The topic of these datasets is either provided by the datahub catalog in which the datasets are registered or are manually assigned. Due the size of the LOD Cloud, a manual topic annotation is feasible but recently introduced approaches like the Roomba framework try to automate this process [Assaf et al., 2015]. The analysis shows that almost 50% of the datasets are about social networking describing people and their relations. Other common topics are *publication*, *government*, and *life science*. Since vocabularies are used to describe the data, the schematic overlap can be computed by counting how often a property of a vocabulary is referred to. The most frequently applied properties are “owl:sameAs” and “rdfs:seeAlso” which underlines the focus of linked data to link datasets among each other.

Semantic Annotation Profiling Semantic annotations include the meaning of information directly in the underlying code of the HTML page. The HTML markup is extended by semantic markup languages providing an additional set of attributes which can automatically be read by machines. One of the most observed semantic markup language is Microdata. The data itself is described using a vocabulary like schema.org.[1] Schema.org characterizes entities and relationships that hold between them. [Meusel et al., 2016] analyzed to which classes of the vocabulary the entities belong to and which properties are used. Most commonly, the unspecific class *thing* has been detected, followed by *creative work* and *intangible*. Concerning the schematic overlap, properties like *name*, *image*, and *url* are frequently used. In addition, several properties describing addresses can be found which indicates that websites about businesses are frequently annotated with Microdata.

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

None of the topical profiles of other web data sources is in line with the topics we detected for web tables which is plausible since other sources are used for other purposes. For example, on websites about computers, we do not expect a lot of relational tables and even if tables are found, the according entities are rarely covered by DBpedia. The LOD cloud only covers about a thousand datasets which results in a rather limited topical coverage. Semantic annotations focus on improving the machine-readability which is mostly exploited by search engines to enrich the presentation of search results. Thus, semantic annotations will mainly be found on websites that benefit from providing information to the search engines. For both structured sources, vocabularies are applied which facilitates the determination of the topical and schematic overlap. Since the meaning of the vocabularies is well-defined, a matching to a knowledge base is not required to recover the semantics. The data overlap is not taken into account by all profiles.

5.3.2 Profiling of Web Table Corpora

In Section 4.4 we already introduced the set of existing web table corpora. As most of them are not publicly available, little is known about their content. The only indications about the covered topics can be derived by considering which tables they use for the evaluation of their methods. For example, the corpus created by [Yakout et al., 2012] is used to augment tables. Within the evaluation, tables with the following topics have been augmented: *cameras*, *movies*, *uk-pm*, *baseball*, *albums*, *us-gov*. Thus, we can infer that these topics are covered in the corpus.

The work of [Hassanzadeh et al., 2015] extends parts of our work and identifies the topical overlap of web tables to the cross-domain knowledge bases DBpedia, Wikidata, YAGO, and Freebase. For their study, they use the Relational Mappable WDC WTC 2012 corpus. Slightly different to our strategy, their goal is to assign classes to any column covering named entities. The matching between columns and classes is performed by first normalizing the values and computing the overlap with labels of instances belonging to the classes. The comparison with the labels only checks for equality and does not allow any deviations or alternative names.

Due to the different goals and annotation strategies, the resulting topic distributions are not entirely comparable but allow to derive tendencies. In Table 5.7 the 10 most frequently annotated DBpedia classes of the profiling approaches are shown. In both profiles, the most commonly detected class is *Agent*. Further, classes like *Person* or *Place* can be found to the same frequent extent. In contrast, our approach discovers much more works but no chemical substances or compounds. One possible reason are the different matching strategies.

By creating profiles to more than one knowledge base, the differences in the topical coverage can be obtained. According to [Hassanzadeh et al., 2015], DBpedia has a good coverage for persons and places while YAGO covers more abstract classes. Freebase clearly includes more media-related classes. Summarizing, the

Table 5.7: Comparison with the annotated DBpedia classes found by [Hassanzadeh et al., 2015].

T2K Match		Approach by [Hassanzadeh et al., 2015]	
Class	#Tables	Class	#Columns
Agent	460 002	Agent	242 410
Work	269 570	Person	186 332
Person	265 685	Place	120 361
Athlete	243 322	PopulatedPlace	112 647
Organisation	194 317	Athlete	85 427
MusicalWork	138 676	Settlement	60 219
Place	133 141	ChemicalSubstance	57 519
PopulatedPlace	119 361	ChemicalCompound	57 227
Company	97 891	Work	53 959
Sportsteam	50 043	Organisation	50 509

choice of the knowledge base that is used for topical profiling has a strong influence since only classes of the knowledge base can be assigned. Although [Hassanzadeh et al., 2015] provide a broader picture of the topics, the schematic, and data overlap with the knowledge bases are not taken into account.

Going beyond the topical overlap, [Barbosa et al., 2014] analyze the schematic overlap among web tables encoding urban data. They profile a set of web tables to enable an easier and better search. For example, based on the schematic overlap, the subset of tables can be determined that includes information about populations. Since the method only takes web tables of a specific topic into account, a general overview about the schemata cannot be drawn.

5.3.3 Knowledge Base Augmentation

Some works propose to augment knowledge bases with internal knowledge base data, e.g., to find new properties based on existing data [Bühmann and Lehmann, 2013]. Considering external data, a notable set of works focuses on Wikipedia as a text corpus annotated with entities. By seeking for patterns based on existing properties, [Aprosio et al., 2013] finds additional properties for DBpedia. [Paulheim and Ponzetto, 2013] propose to identify common patterns in Wikipedia list pages using a combination of statistical methods and textual evidence. These patterns are applied to extract additional classes as well as new facts. For example, from the Wikipedia list with African-American writers it can be derived that all mentioned instances are writers, have an African ethnicity and their nationality is American. The approach has only been evaluated on the mentioned list.

Besides finding new properties and instances, other approaches focus on slot filling. One set of approaches focuses on extraction information from unstructured text, called open information extraction. We will go into more detail about open information extraction in the related work part of Section 6.4. [Dutta et al., 2015] match facts extracted by two open information extraction frameworks, NELL and REVERB, to DBpedia. Further, they analyze how many facts overlap and how many new facts can be generated. From the REVERB ClueWeb text corpus[2], about 15 million facts have been extracted. After filtering out all phrases occurring too rarely, facts with low confidence and facts with numeric expressions (literals), a set of about 1 million facts remains. With the best strategy, 78 085 new facts for altogether 41 properties with an expected precision of 0.78 can be generated.

[Sekhavat et al., 2014] propose a strategy that augments an existing knowledge base with facts from web tables by levering a web text corpus and natural language patterns associated with properties in the knowledge base. Within each row of the table, pairs of strings are extracted and the corpus is searched for all sentences containing both strings. All words in between the mentions are extracted and matched against a set of patterns. An inference model is applied to compute the probabilities of each relation given the patterns. This method is performed for all rows of a table to generate an overall ranking of the relations. For the experiments, the set of PATTY patterns[3] extracted from the New York Times together with 25 Wikipedia patterns are used with YAGO as knowledge base and the ClueWeb text corpus. Applied on two tables about songs and NBA players, 17 of 48 creation facts and 8 of 100 is-affiliated-to facts can be newly extracted.

[Dong et al., 2015] proposed “Knowledge Vault”, a framework for extracting triples from web sources and aims at constructing a knowledge base, using Freebase as source of prior data. Beside web tables, text documents, HTML trees and human annotated pages are taken into account as sources. Altogether, 1.6 billion fused triples are extracted, about 75% originating from HTML trees and only 0.59% from web tables. Requiring a confidence of at least 0.9, 100 million triples can be found, including 0.59 million triples from web tables. Compared to the other sources, triples extracted from web tables only contribute little. With an improved method, 271 million triples are generated out of which 33% ($\sim$ 90 million) are new facts that were not yet in Freebase. Information about the augmented instances, properties, or classes are not provided.

5.4 Knowledge Base Augmentation Potential

In this section, we analyze the potential of the web tables for the use case of filling missing values in a knowledge base. First, we establish our evaluation methodology which uses generated triples that overlap with the knowledge base as ground truth (local closed world assumption). Second, we compare the performance of three different data fusion strategies: a voting-based baseline approach, one strat-

[2]http://lemurproject.org/clueweb09.php/

[3]https://d5gate.ag5.mpi-sb.mpg.de/pattyweb/

egy with a knowledge-based quality measure and one that uses PageRank as an external quality indicator. Finally, we show how many new triples with which quality can be generated and in turn where the largest potential for augmenting knowledge bases with data from web tables lies.

5.4.1 Evaluation Methodology

We evaluate the correctness of the generated triples by comparing them to triples which already exist in DBpedia (overlapping triples). For this comparison, we apply the Local Closed World Assumption (LCWA) [Dong et al., 2014]:

Definition 5.1 *(Local Closed World Assumption) For triples with subject s, predicate p and object o, let $O(s,p)$ be the set of objects for s,p in a knowledge base (overlapping triples). If a triple $(s,p,o) \in O(s,p)$, we assume the triple to be true. Otherwise, if $(s,p,o) \notin O(s,p)$ and $O(s,p) \neq \varnothing$ the triple is incorrect. If $O(s,p) = \varnothing$, we exclude the triple from the evaluation (non-overlapping triples).*

We use the LCWA to enable a large-scale automatic evaluation of the fusion results (which we double-check with a manual evaluation described in Section 5.4.3). As we cannot expect data from web tables to be perfectly clean, we allow for minor deviations when comparing generated triples to triples from the knowledge base. We use a deviation similarity for numeric values, a weighted date similarity for dates, a hybrid Jaccard similarity for strings and equality for references.

By checking which groups contain correct triples that already exist in the knowledge base, we can estimate the upper bound of the data fusion performance, meaning that we can estimate the maximal number of correct triples that could be produced by a hypothetical, ideal fusion strategy. For 691 622 of the 929 170 groups, the set of objects $O(s,p)$ for s,p is not empty, meaning that they overlap with DBpedia. On these groups, we apply the similarity measures described above and find that the number of groups containing the correct triple is $correct_{max} = 310\,284$. We use this number to compute the recall since it is the upper bound of the fusion performance. Thus, only 45% of all groups G with non-empty sets of objects contain the correct triple at least once. The main reason is that generating a correct triple requires correct matching decisions for the class, property, and instance and, in the case of a *reference* data type, also a correct transformation of the string into an instance. Multiplying the errors happening during all matching tasks and taking into account that information in the web tables might also be incorrect or outdated explains the low percentage.

5.4.2 Fusion Strategies

The goal of the fusion step is to decide which triple of a group with the same instance-property combination will be selected as output and used by subsequent steps such as slot filling. We compare the following three data fusion strategies:

1. **Majority/Median (MM)** The majority/median fusion strategy selects the most frequent value in the group as output (simple voting). For groups of data type *string* and *reference* the majority is taken and for groups of data type *numeric* and *date* the median.[4] The MM strategy is a simple baseline which does not take any quality indicators into account. Thus, each value has the same weight during the voting independent of the table it origins from.

2. **Knowledge-based Trust (KBT)** We extend the MM strategy by assigning a trust score to each attribute. For *string* and *reference*, we then apply a weighted vote and for *numeric* and *date* a weighted median. The trust score is calculated for each attribute as the number of correct overlapping triples, normalized by the total number of overlapping triples. In addition to weighting the values, we completely filter out all attributes and in turn triples with a trust score below 0.35. By calculating the score from the overlapping triples, we create a measure of correctness for the attribute that is the source of the respective triples. This corresponds to the concept of knowledge-based trust [Dong et al., 2015, Yin and Tan, 2011]: we weight each triple by the correctness of the other information provided by same source with respect to information that is considered to be trustworthy (the knowledge base).[5]

3. **PageRank-based Trust (PRT)** This strategy works like the KBT strategy, with the difference that the score assigned to each triple is the normalized PageRank [Page et al., 1999] of the website that is the source of the web table. Over the last decade, PageRank has been widely used to assess the quality of web content and has also previously been used for fusion [Pasternack and Roth, 2010]. The PageRank scores are calculated on the host-level using the 128 billion hyperlinks contained in the Common Crawl 2012.[6] Filtering does not improve the results for the PageRank scores. In contrast to KBT, PRT relies on hyperlinks as quality indicators while KBT relies on comparing web data to previously trusted data.

In the following, we report the performance of the fusion strategies. First, we determine which strategy works best for fusing web tables. This strategy is then used to report about the potential of web tables for slot filling. Second, we examine the claim by [Dong et al., 2015] that knowledge-based trust outperforms a strategy with PageRank as quality indicator [Dong et al., 2014].

Table 5.8 shows the number of overlapping fused triples F_o and non-overlapping fused triples F_{no} that are generated by the fusion strategies. Note that the non-overlapping triples are the triples which could be added to the knowledge base.

[4]Note that we do not take the modal value since the groups tend to be small such that outliers could be determined as output.

[5]Note that as we use the same methods for the calculation of the trust score and the evaluation, we apply a 5-fold cross-validation for this strategy.

[6]These scores and the corresponding web graph are available for download at `http://webdatacommons.org/hyperlinkgraph/2012-08/download.html#toc4`

The last three columns present the performance in terms of precision, recall, and F-measure. The performance values are calculated using the LCWA.

Table 5.8: Number of (non-)overlapping triples and evaluation results per fusion strategy.

Strategy	F_o	F_{no}	Precision	Recall	F-measure
MM	691 622	237 548	0.37	0.82	0.51
KBT	378 892	64 237	0.64	0.79	0.71
PRT	691 622	237 548	0.37	0.81	0.50

The baseline approach, MM, does not apply any filtering, hence the precision can maximally be 45% ($correct_{max}$ divided by F_o). Taking this into account, the achieved precision of 0.37 is at an acceptable level for a simple approach. The MM fusion is able to identify the correct value for 82.3% of all groups. This also includes all groups of size one, where the fusion cannot choose from multiple triples and just forwards the received input as output. The second approach, KBT, filters out attributes with a low trust score and can hence decide to not create a triple for a given group. This results in a 0.27 increase in precision and only has a very small trade-off in recall, which decreases by 3.8. The third strategy, PRT, does not result in any improvement over the MM baseline. Thus, we can confirm the finding of [Dong et al., 2015] that the quality of a web source is not necessarily determined by its popularity. For example, gossip pages are frequently interlinked but do not necessarily provide high-quality web tables. As KBT performs best, we choose this fusion strategy for further investigations.

5.4.3 Manual Evaluation

[Dong et al., 2014] show that the LCWA assumption is a valid approximation to estimate the quality of non-overlapping triples. We perform two manual evaluations in order to verify the results. First, we test the LCWA by manually evaluating a sample of overlapping fused triples. Second, we manually evaluate a sample of non-overlapping fused triples to determine whether the performance can be transferred to non-overlapping fused triples.

To test the LCWA, we manually evaluate a set of $1\,000$ overlapping fused triples. The automatic evaluation of this sample according to Section 5.4.1 results in a precision of 0.68, while three human annotators determine a precision of 0.72. Overall 958 out of $1\,000$ triples were evaluated correctly by the automatic evaluation, which results in an error rate of 4.2%. This result is a signal for the validity of the LCWA and justifies its application for our experiments. However, during the manual evaluation we spot some error categories, which shed light on possible shortcomings of this method:

- **Changes Over Time** Facts can be outdated in the knowledge base, leading to an incorrect evaluation of more up-to-date web tables.

- **Different Granularity** Facts can have different levels of granularity, e.g., the *city* of the *Emroy university* is *Druid Hills Georgia* in DBpedia. In the web tables, we find the object *"Atlanta"* which does not have a similar label. But knowing that *Druid Hills Georgia* is a community in the metropolitan area of Atlanta, this triple can be regarded as correct.
- **Missing Objects in Lists** If a list is incomplete in the knowledge base, the evaluation fails if the web table contains a correct, but missing value.

The second question we want to investigate is whether the performance estimated for overlapping fused triples can be transferred to the non-overlapping fused triples. As the non-overlapping fused triples are the candidates for slot filling, an evaluation that cannot be transferred would not be suitable for the use case. Hence, we manually evaluate another sample of 500 randomly selected, non-overlapping fused triples. On this sample, the KBT strategy achieves a precision of 0.62.[7] The determined precision is very close to the one that was estimated using the LCWA on the overlapping fused triples (0.64), which we take as an indication for the validity of transferring the performance to non-overlapping fused triples.

5.4.4 Fusion Results

In this section, we show the potential of web tables for slot filling. We present performance statistics for data types, classes, and properties.

Data Types Table 5.9 shows the fusion performance by data type. F_o presents the number of overlapping fused triples, F_{no} the non-overlapping ones. While the *date*, *reference*, and *string* data types have a comparable performance, the recall of data type *numeric* is significantly lower. Some *numeric* attributes tend to be more noisy due to conflicting objects, changes over time or different interpretations of certain properties. Thus, even correct triples are filtered out by the KBT fusion, as the trust score is not high enough.

Table 5.9: Knowledge-based trust fusion results per data type.

Data Type	F_o	F_{no}	Precision	Recall	F-measure
Numeric	28 364	10 613	0.64	0.45	0.53
Date	171 653	23 301	0.63	0.81	0.71
String	34 260	14 285	0.76	0.81	0.78
Reference	144 615	16 038	0.63	0.87	0.73

We further identified the following frequent causes of incorrect fusion results:

- **Conversion Issues** Some conversions are not performed correctly. As an example, the *birthDate* of *Jeff Zatkoff* is *"6/9/1987"* according to DBpedia

[7] 19 triples were excluded as the human annotators could not determine the correct object. This happened for example in the case of rare properties like *bSide* of a record or *upperAge* of colleges.

but we find the date *"9/6/1987"* in the web tables. Without knowing which date format is used within the web table, it is not possible to parse it correctly. This problem constitutes the largest part of the errors for the data type *date*.

- **Ambiguous Entities** The instance matching can make mistakes, especially if the label of the entity is ambiguous. This occurs with very common names of people or with musical works like cover versions of albums. A wrongly identified entity can lead to incorrect results for all data types.
- **Multiple correct values for the same property.** If the property is not well-defined or too broad, many values could be a correct match for that property. Unfortunately, if DBpedia only includes some of all the possible values, correct fused triples will be marked as incorrect. This problem constitutes about 55% of all errors for the *string* data type.

Classes Table 5.10 shows the fusion results for the classes. The second column contains the number of overlapping triples F_o per class while the third column shows the set of non-overlapping triples F_{no}. All performance measures in the last three columns are computed on F_o. We find the highest amount of non-overlapping fused triples for *Work*, especially *Film*, and for *Person*, especially *Athlete*. Hence, these classes are beneficial candidates for slot filling based on web tables. Concerning precision and recall, the best results are achieved for *Species*, and *Place*.

Properties Similarly to the classes, we expect the amount of overlapping and non-overlapping triples as well as the estimated quality to depend on the particular property. Table 5.11 shows the properties with the highest number of overlapping fused triples. The column "ratio" presents the ratio between the number of overlapping triples and the total number of triples using this property in DBpedia. Almost all properties with the most overlapping triples refer to properties describing works (*releaseDate*, *artist*, *director*) or persons (*birthDate*, *activeYearStartDate*). This reinforces the previous findings that the largest topical overlap between DBpedia and the web tables can be found for *Work* and *Person*. Concerning the precision, most properties are close to the average precision, with exceptions being *musicalArtist* and *number* with a lower precision. Supposedly, this is caused by number (e.g., the number of a baseball player in a certain team) being a time-varying property and *musicalArtist* potentially describing ambiguous entities.

In addition to the analysis of the overlapping triples, Table 5.12 depicts the properties with the largest amount of non-overlapping triples. The precision is approximated with the precision that was achieved on the overlapping fused triples for the same property. The ratio column indicates the potential for slot filling for a particular property. We can almost double the number of *publicationDate* triples and increase the amount of *releaseDate* triples in DBpedia by 11%.

Further, a selection of properties with a high precision and at least 50 non-overlapping fused triples can be found in Table 5.13. For triples of these properties, a slot filling approach results in very high-quality data. While the properties with

Table 5.10: Knowledge-based trust fusion results per class.

DBpedia Class	F_o	F_{no}	Precision	Recall	F-measure
Agent	138 100	0.64	0.72	0.68	23 062
+ Person	117 522	15 050	0.64	0.72	0.68
| − Athlete	84 562	9 067	0.65	0.68	0.66
| − Artist	2 019	427	0.71	0.83	0.77
| − OfficeHolder	3 465	510	0.70	0.85	0.77
| − Politician	3 124	1 167	0.53	0.77	0.63
+ Organisation	20 522	7 903	0.65	0.69	0.67
| − Company	6 376	2 547	0.70	0.83	0.76
| − SportsTeam	790	132	0.67	0.89	0.77
| − Educational Institution	8 844	3 132	0.64	0.71	0.67
| − Broadcaster	4 004	1 924	0.56	0.459	0.50
Work	189 131	27 867	0.61	.83	0.71
+ MusicalWork	118 511	8 427	0.60	0.83	0.70
+ Film	29 903	12 143	0.57	0.80	0.67
+ Software	17 554	2 766	0.59	0.76	0.67
Place	32 855	9 871	0.77	.858	0.81
+ PopulatedPlace	16 604	6 704	0.71	0.78	0.7
| − Country	2 084	433	0.74	0.69	0.71
| − Settlement	540	224	0.58	0.67	0.62
| − Region	362	70	0.59	0.78	0.67
+ Architectural Structure	10 441	1 775	0.83	0.94	0.88
+ NaturalPlace	743	64	0.84	0.94	0.89
Species	9 016	1 429	0.78	0.89	0.83

Table 5.11: Properties with most overlapping triples.

Property	F_o	Precision	Ratio
releaseDate	92 383	0.63	0.07
birthDate	61 636	0.77	0.06
artist	25 563	0.65	0.27
musicalArtist	20 663	0.29	0.53
musicalBand	18 160	0.50	0.46
director	8 082	0.62	0.10
activeYears StartDate	7 934	0.66	0.12
activeYears EndDate	7 861	0.71	0.14
deathDate	7 448	0.63	0.02
number	6 160	0.38	0.10

Table 5.12: Properties with most non-overlapping triples.

Property	F_{no}	Precision	Ratio
releaseDate	15 836	0.63	0.12
number	3 557	0.38	0.06
publicationDate	2 693	0.69	0.96
alias	1 471	0.44	0.01
locationCountry	1 304	0.56	0.09
country	1 242	0.67	0.00
synonym	1 240	0.56	0.01
status	1 116	0.42	0.04
birthDate	1 000	0.77	0.00
artist	971	0.65	0.01

the highest precision can only add a rather small number of triples, the properties *throwingSide* (for *BaseballPlayer*), *icaoLocationIdentifier* (for *Place*), and *family* (for *Species*) add thousands of triples with an above average precision.

Table 5.13: Properties with the highest precision results.

Property	F_{no}	Precision
numberOfIslands	157	1.00
province	67	1.00
seniority	60	1.00
sire	366	0.99
games	247	0.97
illustrator	81	0.97
iso6391Code	236	0.97
throwingSide	2 500	0.96
icaoLocationIdentifier	5 459	0.94
family	4 760	0.85

Due to the amount of instances, it does not make sense to provide any statistics about slot filling potential for particular instances.

5.5 Summary

In this chapter, we generated a profile of the WDC WTC 2012. The profile contains basic statistics, e.g., 3 billion values are included in the corpus. Further, we presented insights about the topical, schematic, and data overlap by matching the tables to DBpedia. The key findings are the following:

- The majority of relational web tables (97%) does not contain data that can be related to DBpedia, e.g., tables about products.

- Among the overlapping tables, the distribution of classes is mostly in line with the topical distribution in DBpedia.
- About 25% of all DBpedia properties are referred in web tables, this holds for 15.6% of the instances.
- 88% of the matching DBpedia properties and 70% of the instances are described within at least two web tables, indicating a large inter-table overlap.
- Grouping the triples results in about 1 million groups of alternative values (average group size of 8.5) which presents the amount of facts that is extracted from the corpus using the presented methodology.

Based on this profile, we know the characteristics of web tables and the topical coverage in the web table corpus. Thus, the profile provides a realistic view on the web table to knowledge base matching task which will be used during the creation of the T2D gold standard in Chapter 6. Further, the potential for an individual use case can be derived by the profile. However, since the profile strongly depends on the content of the knowledge base as shown by [Hassanzadeh et al., 2015], tables that do not overlap with DBpedia do not occur in the presented profile.

To estimate the potential for the particular use case of filling missing values in DBpedia, we evaluated three different fusion strategies using the LCWA. As result, the knowledge-based trust fusion outperforms a PageRank-based fusion as well as a voting-based baseline strategy. In addition, we confirmed the applicability of the LCWA and showed that the performance approximation for overlapping fused triples is transferable to fused triples with no overlap in the target knowledge base. By applying the knowledge-based trust fusion, about 65 000 triples can be generated that are not contained in DBpedia. Comparing the overlapping triples with the triples in DBpedia, a precision of 0.64 for the new triples is determined. Altogether, the potential for slot filling is greatly dependent on the particular class and property. As example, we can almost double the number of *publicationDate* triples in DBpedia.

Chapter 6

Web Table to Knowledge Base Matching

In the previous chapters, we introduced web tables as well as knowledge bases which serve as input for the matching. Further, we have already presented the foundations of the data integration process and more particularly of matching. Within this chapter, everything gets combined: how does the matching of web tables to knowledge bases looks like and which challenges need to be addressed.

To match web tables to a knowledge base, both the schema and data matching needs to be performed since correspondences are not known beforehand. Altogether, three matching tasks need to be conducted: firstly, the assignment of classes from the knowledge base to tables, secondly of properties to attributes, and lastly of instances to entities. Existing systems [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011, Zhang, 2016] for matching web tables to knowledge bases address the three tasks in an integral fashion which makes it possible to take advantage of the findings gathered within other tasks. On the one hand, most existing approaches restrict themselves to attributes covering named entities without considering attributes including literals [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011] . On the other hand, they have not been applied to large corpora covering millions of web tables such that the applicability for real-world scenarios has not been demonstrated [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011, Zhang, 2016]. To overcome this situation, we developed the T2K Match algorithm, a method that tackles all three matching task in a combined way and is not restricted to attributes covering entities. The algorithm has been implemented to be able to handle millions of web tables and in turn to provide a web-scale applicability as already demonstrated for the profiling (Chapter 5). Most important for this thesis, T2K Match provides us with the possibility to analyze the matching results for all tasks.

The gold standards on which state-of-the-art approaches have been evaluated suffer from similar drawbacks as the systems. First, they do not cover correspondences of attributes to datatype properties. Second, they are restricted regarding the representativity of large scale web table corpora since they either focus on a specific topic or on tables from one website. Finally, the proposed gold standards exclusively cover tables that overlap with the knowledge base. With the T2D gold standard, we present a gold standard covering a wide range of challenges, reflecting the matching task more realistically. Together with T2K Match, we are able to perform a systematic evaluation on a single gold standard that provides a variety of challenges that need to be addressed by web table to knowledge base matching systems.

In this chapter, we first introduce the challenges that arise when matching web tables to knowledge bases and provide an overview of the matching tasks (Section 6.1). In Section 6.2, we provide a description of the algorithm, followed by an introduction to the T2D gold standard in Section 6.3. While Section 6.4 discusses related works, Section 6.5 presents the evaluation of T2K Match applied to the T2D gold standard. In the last section, the chapter is summarized and issues that cannot be overcome by T2K Match are discussed.

A description of T2K Match together with parts of the evaluation on the T2D gold standard has been published by [Ritze et al., 2015]. I performed the table selection and parts of the annotation of the T2D gold standard, co-developed the T2K method and have co-authored most of the sections.

6.1 Introduction to Web Table Matching

Before web tables can be used for applications like knowledge base augmentation, the semantics of the web table need to be recovered: the table topic, the attribute meanings and the real-world objects that are referred by entities. In this section, we first describe challenges that occur when matching by web tables. Further, we introduce the matching tasks that need to be addressed in order to create a semantic table interpretation. The matching tasks comprise both, schema and data matching.

Terminology According to the definitions given in Section 4.1.1, web tables are composed of rows and columns. For relational tables, we use the terms *entity* and *attribute* since each row represents one entity and each column represents an attribute describing the entity. Thus, for relational tables the terms row and entity as well as column and attribute are interchangeable. To distinguish between web table and knowledge base elements, we employ the terms instance, property, and class for the according knowledge base elements as introduced in Chapter 3.

6.1.1 Challenges

By integrating web tables, all four data integration challenges as mentioned in Section 2.2 are posed: volume, velocity, variety, and veracity [Dong and Srivastava, 2015]. In the following, we discuss why these challenges are raised by web tables and which other aspects complicate the matching of web tables.

- **Volume** According to the statistics presented in Chapter 4, existing web table corpora cover millions of relational web tables. When matching these corpora, huge volumes need to be handled which especially requires a sufficient performance. In addition, while the variety of domains turns web tables into a rich source, this variety renders domain-specific techniques insufficient.
- **Velocity** Every day, web pages are added and updated. Thus, the information about the same topic presented in two tables do not need to refer to the same point in time. Hence, if values of time-dependent attributes are compared, similarities might not be detected.
- **Variety** As introduced in Chapter 2, different kinds of heterogeneities exist, varying from syntactical over terminological to conceptual heterogeneity. We are only using web tables of the same format such that syntactical heterogeneity does not need to be considered. In contrast, terminological heterogeneity can be discovered very frequently due to the fact that web tables do not adhere to controlled schemata which results in a linguistic diversity [Venetis et al., 2011]. Rather than databases or ontologies that are usually generated by domain experts who are aware of the common terms in a domain, web tables can be created by everybody. Another aspect is the natural space restriction which is given by the rendering of the web page. Thus, abbreviating terms is a popular solution. Besides the terminological heterogeneity, conceptual heterogeneities need to be overcome. As determined by [Lautert et al., 2013], web tables show a considerable structural variety. Only 18% of the tables are akin to traditional databases. The main reasons are values that cover lists and tables with spanning rows or columns. In addition to these characteristics, many different modeling styles naturally occur since web tables are generated by different humans.
- **Veracity** All data sources coming from the Web have to face differences in quality. Often, web data tends to be "dirty" in contrast to data that is maintained and updated by domain experts. Typical quality issues are the correctness and completeness of the data as well as the timeliness. Web tables are no exceptions regarding the quality. Often, the presentation of a complete view on a domain covering all entities is not the goal of a web table. Instead, a web table usually focuses on a specific topic giving a particular view on the data. As stated by [Cafarella et al., 2008a], there are very few tables with large numbers of attributes that provide a full picture of a topic.

In addition to the general challenges, some characteristics are quite specific for web tables and are of hardly any importance for other data sources: the absence of a formal schema, the source size, and missing names. These issues do not need to be tackled when matching databases or ontologies.

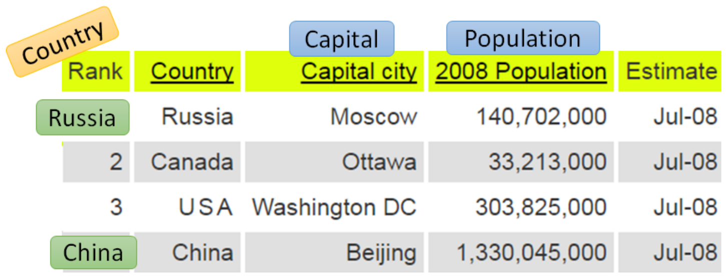
- **No schema** Web tables do not provide a formal schema. Information about the data types, the entity labels, and attribute headers need to be guessed. All mistakes that are made during the metadata recovery can have a strong influence on the matching result.
- **Small size** Web tables tend to be small and narrow regarding the number of entities and attributes [Cafarella et al., 2008b]. Thus, matching algorithms are required to rely only on a restricted set of available information and in turn less evidence can be gathered.
- **Missing headers** Web tables attributes often either lack headers or the headers are ambiguous and non-informative [Wang et al., 2012, Embley et al., 2006]. Within the WDC WTC 2012, 50% of the attributes do not have a header and the most common headers are very generic, e.g., "name" or "date". All in all, the completeness and the descriptiveness of headers directly impacts the matching efforts [Pimplikar and Sarawagi, 2012].

6.1.2 Matching to Knowledge Bases

To match web tables to a knowledge base, three tasks need to be addressed: the instance, the property, and the class matching. While instance matching refers to the data matching, the property and class matching are schema matching subtasks. As result of the matching process, each web table is assigned to a class, each attribute to a property, and each entity to an instance, if a knowledge base element exists.

Example 6.1 An example web table about countries matched to a knowledge base is depicted in Figure 6.1. The table has been annotated with the class *Country*. The attributes have been associated with the properties *Capital* and *Population*. The instance matching task generated correspondences between the entities named *Russia* and *China* to the according instances representing the countries. Note that not all table elements have counterparts in the knowledge base.

Over the last years, a set of systems for matching web tables to knowledge bases has been developed. Some of the systems focus on a certain task, e.g., [Bhagavatula et al., 2013, Quercini and Reynaud-Delaître, 2013] on the instance matching, the approaches of [Pham et al., 2016, Ermilov and Ngomo, 2016] on the property and the method introduced by [Zwicklbauer et al., 2013] on the class matching task. Since none of the tasks can rely on available correspondences, existing web table to knowledge base matching systems address all three matching tasks in an integrated fashion [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011, Zhang, 2014b]. Common strategies are probabilistic models [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011] and iterative approaches [Zhang,


Country
Capital
Population
Russia
China

Rank	Country	Capital city	2008 Population	Estimate
	Russia	Moscow	140,702,000	Jul-08
2	Canada	Ottawa	33,213,000	Jul-08
3	USA	Washington DC	303,825,000	Jul-08
	China	Beijing	1,330,045,000	Jul-08

Figure 6.1: Correspondences generated by the three matching tasks for an example table about countries.

2014b]. We discuss them in more detail in Section 6.4.

The mentioned approaches, e.g., by [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011], have a different definition of the matching tasks. Classes are assigned to all named entity columns instead of the table. Further and most important, the systems annotate pairs of named entity columns with object properties, also referred to as relations or relationships. Thus, columns covering literals are not taken into account and in turn, datatype properties are not considered. One exception is the TableMiner+ system [Zhang, 2016] which considers datatype properties. Depending on the use case, focusing on the matching of named entity columns is only partly sufficient. As determined during the web table profiling in Chapter 5, about 75% of the triples that can be used for slot filling are actually datatype properties, see Section 5.4.4, Table 5.9.

6.2 Methodology

In this section, we introduce the matching algorithm T2K Match.[1] T2K Match addresses all three matching tasks without restrictions like the non-compliance of literal columns. The three matching tasks mutually influence each other, relying on the findings of the other tasks. We first explain the general workflow, followed by descriptions of the matchers that build the core of the matching. The evaluation of T2K Match will be presented in Section 6.5. The analysis of the results serves as foundations for all following chapters.

6.2.1 Workflow

Within this section, the T2K Match workflow is described, as illustrated in Figure 6.2. All web tables are assumed to be relational with recovered metadata. Hence, the entity label column, the attribute label row, and the column data types have been determined as described in Section 4.2.4. In a knowledge base, the metadata

[1] `https://github.com/T2KFramework/T2K`

is automatically available. All steps that are expected by a matching process, see Section 2.3, are included: preprocessing, execution of matchers, aggregation, and classification. Since the instance, property, and class matching are addressed, each task needs to perform these four steps. During the matcher execution, the tasks mutually influence each other and rely on information provided by the other tasks. After the aggregation and classification, a set of correspondence for each matching task is returned. In the following, we describe each step in more detail.

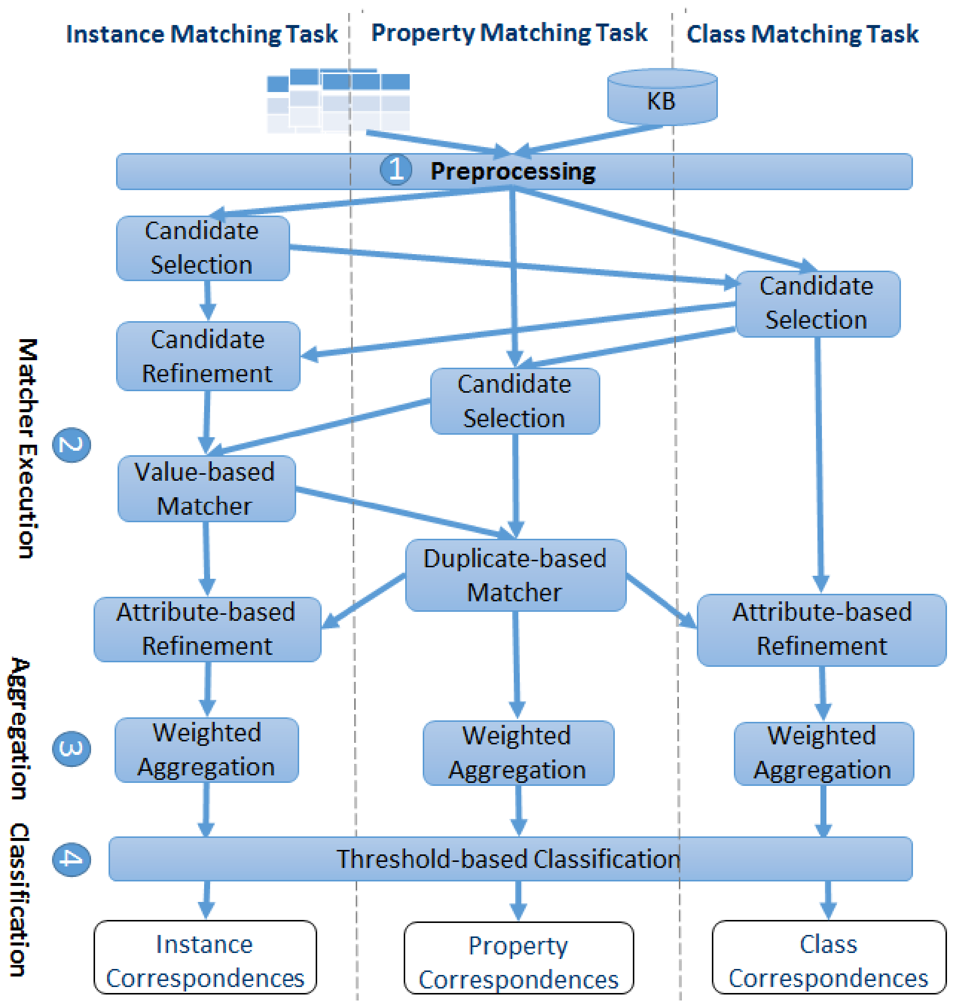


Figure 6.2: Workflow of the T2K Match algorithm.

Preprocessing The motivation for preprocessing is to normalize the data such that errors can be avoided during the matching. For example, the inconsistent use of separation characters like hyphens can decrease the similarity of two string which is calculated by a similarity measure. Hence, prior to matching of web tables to a knowledge base, we try to make sure that all values are available in an established standard format. During the metadata recovery, the data types have been determined and in turn, numeric and date values have already been transformed into a standardized format, including the unit transformation. Thus, the preprocessing focuses on the string values. In order to achieve a standard format for all string values, they first need to be cleaned. Since we are using data from the Web, the values can include HTML artifacts like HTML character entities, e.g., the non-breaking space "&nbsp;". Beside the HTML artifacts, we remove special characters like parentheses, punctuations or slashes. Further, any additional whitespaces are excluded. After the cleaning, the values are converted to lower case and a set of hand-crafted transformation rules is applied to resolve abbreviations, e.g., "co." is transformed into "company". Figure 6.3 presents an example how two string are cleaned and normalized. Details can be found in the T2K normalization project.[2]

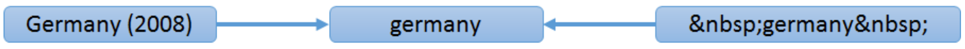


Figure 6.3: Normalization of two strings.

Further normalizations or structural transformations are not performed since we are not aware of the table's topic and the attribute's meaning. The values in the knowledge base are processed the same way. For object properties, we get the instance that is referred to by the URI and take it's name as defined by the *rdfs:label* property. As mentioned in Chapter 2, one main challenge of data matching is the number of comparisons. To enable efficient comparison, a Lucene Index[3] is built that contains the labels of all knowledge base instances. With this index, a fast look-up is possible which provides the foundation of a web-scale applicability.

Matcher Execution For each matching task, a set of matchers comparing different features is applied. At first, candidates are identified for the entities, attributes, and the table that serve as blocking keys in all following steps. A more detailed description of the candidate selection, including the candidate refinement, will be provided in Section 6.2.2. After the candidates have been determined, the similarities of the values are computed within the value-based matcher (Section 6.2.3). Using the instance candidates together with the value similarities, a duplicate-based matcher is applied to estimate which properties fit to the attributes (Section 6.2.4). Finally, an attribute-based refinement matcher is used for the instance and class matching task. It exploits the findings of the property matching task and in turn updates the similarities to the instance and class candidates (Section 6.2.5). Altogether, each task has an influence on the other two tasks.

Aggregation The goal of the aggregation is to combine the similarity matrices holding the similarities that have been computed by the matchers. We use a weighted aggregation for each task. As any other parameter, the weights are

[2]https://github.com/T2KFramework/T2K/tree/EDBT/de.dwslab.T2K.Normalisation

[3]https://lucene.apache.org/core/

determined using a genetic algorithm which will be described subsequent to the classification.

Classification Since we are requiring an one-to-one mapping (Chapter 2), we only create a correspondence to the knowledge base element with the highest similarity resulting from the aggregation. To identify whether a correspondence is likely to hold, we apply a threshold-based classification that filters out all correspondences with a confidence score below a given threshold. The threshold is determined during the parameter section.

Parameter Selection The parameters like the aggregation weights or the threshold for the classification are determined by a genetic algorithm. Genetic algorithms have been shown as promising for both schema and data matching [Ritze and Paulheim, 2011, Isele and Bizer, 2012]. For each parameter, a set of possible values is defined, at most 10 values are allowed. The genetic algorithm tests the parameter values in order to find the best possible combination according to an evaluation criterion. In our case, we define the best possible combination as the combination achieving the highest precision. We terminate the genetic algorithm as soon as we do not find a better combination. The specific parameters we used for the experiments are stated in the experimental setup of the evaluation, Section 6.5.1.

The following example presents the workflow applied on an example table.

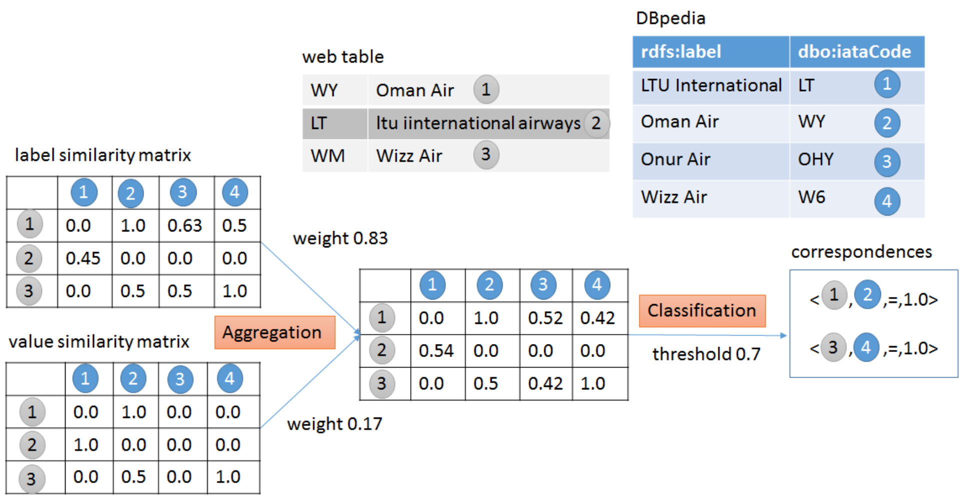
Example 6.2 In Figure 6.4, an excerpt of a table about airlines and the corresponding DBpedia class is depicted.[4] To detect instance correspondences, the entities and the instances in DBpedia are compared by applying a label-based and a value-based matcher. Neglecting the concrete functioning which will be described in the subsequent sections, both matchers generate a similarity matrix. During the aggregation, these matrices are combined using the weights, 0.83 for the label and 0.17 for the value matrix, determined by the genetic algorithm. The resulting similarity matrix contains the aggregated scores e.g., a similarity of 1.0 between the entity "Oman Air" and the according instance "Oman Air". During the classification, a threshold of 0.7 is applied to only generate correspondences that are likely to hold. As result, correspondences for the entities "Oman Air" and "Wizz Air" are generated with a confidence score of 1.0.

After introducing the overall workflow of the algorithm, we will continue in subsequent sections with a more detailed explanation of the matcher execution which forms the core of the matching.

6.2.2 Candidate Selection

With the goal to matching millions of web tables, we need to find a way to deal with massive amounts of comparisons especially for the instance matching task.

[4]The table originates from the T2D gold standard: 5873256_0_7795190905731964989.



web table

WY	Oman Air	1
LT	ltu iinternational airways	2
WM	Wizz Air	3

DBpedia

rdfs:label	dbo:iataCode	
LTU International	LT	1
Oman Air	WY	2
Onur Air	OHY	3
Wizz Air	W6	4

label similarity matrix

	1	2	3	4
1	0.0	1.0	0.63	0.5
2	0.45	0.0	0.0	0.0
3	0.0	0.5	0.5	1.0

value similarity matrix

	1	2	3	4
1	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0
3	0.0	0.5	0.0	1.0

	1	2	3	4
1	0.0	1.0	0.52	0.42
2	0.54	0.0	0.0	0.0
3	0.0	0.5	0.42	1.0

correspondences

<1,2,=,1.0>

<3,4,=,1.0>

Figure 6.4: T2K Match Workflow applied on an example table about airlines.

The candidate selection presents a blocking step to determine initial sets of instances, properties, and classes that fit to the according table elements. All further comparisons and similarity computations are only performed between a table element and its candidates, such that the number of comparisons drastically decreases. In the workflow, we start with the identification of candidates for entities, followed by candidates for the table and for the attributes.

At first, the instance candidates are detected by searching for each entity label in the already generated Lucene index. The index returns a list of instances, ordered by the internal Lucene ranking. In more detail, the order depends on the TF-IDF similarity of the instance labels to an entity label. To restrict the possibly large amount of candidates, only the top k candidates are kept if the similarity score is above a threshold. Thus, T2K Match will identify an instance as candidate if its label is similar to the entity label.

To find class candidates, we take the best candidate for each entity and determine the classes to which this candidate belongs. If an instance belongs to more than one class, the instance counts for all of them. In the end, we choose the most frequent classes (top k) as candidates for the table. Such a majority-based class decision is commonly used to find correspondences between tables and classes since it is straightforward but achieves a similar performance compared to more complex strategies, as analyzed by [Zwicklbauer et al., 2013]. In turn, the selected classes serve as basis for the property candidates: only properties that belong to one of the candidate classes are considered. Belonging to a class does not require the property to have the class as domain. Instead, a property belongs to the class if it is used by instances of that class in the knowledge base. By not relying on the domain restriction given by a knowledge base, we overcome difficulties that result from inconsistencies between domain definition and property usage.

At this point, candidates for all three matching tasks have been discovered. Since we started with the instance candidates, this task did not yet benefit from information that has been gathered during the candidate identification of the other tasks. Hence, we perform a second candidate selection for the instance matching task, called candidate refinement, that again queries the index. This time, only instances belonging to one of the candidate classes are allowed to be returned. In addition, a lower threshold is applied to allow large deviation of the labels. At the same time, all candidates that do not belong to a candidate class are removed. Hence, implausible instance candidates are discarded while more likely instance candidates are added. As we show in the evaluation, Section 6.5.2, on average 1.6 additional instance candidates are added for each entity in the table.
Since the candidate selections presents a blocking, a table element will never be compared with any other element of the knowledge base except for its candidates. We will analyze the performance of the blocking strategy in Section 6.5.2.

6.2.3 Value-based Matcher

After identifying candidates, we focus on the values found in cells. This is possible due to the blocking since we no longer need to compare each cell with each value of the knowledge base. We compute similarities between the values of an entity and all values of the instance candidates, regardless of which candidate property the values belong to. Usually, data matching approaches do not need to address this issue since at least a few schema correspondences are known. Additionally, we restrict the number of required comparisons by only comparing values of the same data type which has been determined as part of the metadata recovery, see Section 4.2.4. In case of multi-values contained in a cell, we calculate the similarity of all combinations of values and choose the maximum.

For each data type, a type-specific similarity measure is applied. For strings, a hybrid Jaccard with Levenshtein as inner measure, for numeric values the deviation measure and for dates the weighted date similarity measure is used. An overview of the similarity measures is given in Section 2.3.2. Note that similarities below a certain threshold, see experimental setup in Section 6.5.1, are set to 0.

Example 6.3 Figure 6.5 depicts the value comparison between values of an entity to the values of one instance candidate. Since only the attribute A is of type string, it's values are only compared to values covered by the string property A'. Apparently, the strings are equal such that a similarity score of 1.0 is specified. The attributes B and D have been determined as numeric attributes as well as the property B'. While the value of B is similar to the value of B', this does not hold for D. Lastly, the date contained in C is compared to the dates in C' and D'. Since nothing more than the years are stated in the web table, only the years can be taken into account. The resulting similarity of values in C to values in C' is 1.0, in contrast the similarity between the date of C and D' is 0.0. In addition to the functioning of the value-based matcher, we can see the difficulties that arise in the data type detection. By only considering the value of attribute C, we cannot state whether C is of data type date or numeric.

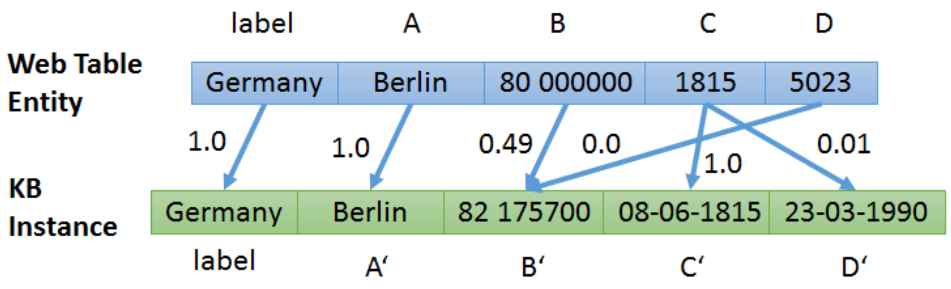


Figure 6.5: Value-based matching of an entity with one of its candidates.

The similarities that have been determined by the value-based matcher serve as basis for the duplicate-based matching which will be explained in the next section.

6.2.4 Duplicate-based Matcher

As introduced in Chapter 2, the duplicate-based schema matching exploits duplicated entities in order to estimate whether two schema elements have the same meaning [Bilke and Naumann, 2005]. The value-based similarities serve as basis to get an impression which entity-instance pairs are duplicates. The pseudocode for the duplicate-based matcher is given in Listing 6.1. For each web table entity, the best instance candidates according to their similarity are determined. Each value v_{ip} of the top k instances has an entity value v_{ea} which fits best according to the computed value similarities. The similarity of the attribute a and the property p, to which the values belong to, is increased by the similarity of the values $sim(v_{ea}, v_{ip})$ and weighted with the similarity of the entity to its candidate $sim(e, i)$. In summary, each value votes for an attribute-property pair.

The intuition is that an attribute including the same data as a property is expected to share many similar values on similar entity-instance pairs. After we applied the duplicate-based matcher, we know the similarities between attributes and properties that again serve as basis for the attribute-based refinement matcher. Note that we do not include any similarity between the attribute label and the property name since the headers are often missing or non-informative, see Section 6.1.

6.2.5 Attribute-based Refinement Matcher

Similarities between attributes and properties can be exploited in two ways: to improve the class and the instance candidates. For each class candidate, the similarity

Listing 6.1: Pseudocode to perform the duplicate-based property matching between attribute a and property p.

```
sim(a,p) <- 0
 for(entity e in E) {
   values_e <- all values of e
   topCandidates <- best candidate instances for e
   for(instance i in topCandidates) {
     values_i <- all values of i
     for(value $v_{ip}$ in values_i) {
       $v_{ea}$ = argmax(sim($v_{ip}$,values_e))
       sim(a,p) += sim(e,i) · sim($v_{ea}$,$v_{ip}$)
 }}}
```

scores of its properties are summed up. Thus, the more properties of a class are similar to attributes of a table, the more likely the class fits to the table. Since classes in a knowledge base are usually organized in a class hierarchy and all instances of a subclass also belong to its superclass, the similarities between the table and the knowledge base class computed on the data-level will not help to distinguish whether a sub- or the superclass is most suitable. Using the similarities between the attributes and the properties especially helps to overcome this issue. Assuming a web table about baseball players: based on the similarities to the instances, both classes *dbo:BaseballPlayer* and its superclass *dbo:Athlete* are suitable for the table. However, if we know that an attribute stating the batting side corresponds to the property *dbo:battingSide* and this property is exclusively used by baseball players, we can decide for the more suitable class *dbo:BaseballPlayer*.

For the instance matching task, similarity scores are computed analogously to the similarities of attributes to properties. To compute the similarity of an entity e with an instance i, the following formula is applied:

$$sim(e,i) = \sum_{v_{ea} \in values(e)} \sum_{v_{ip} \in values(i)} sim(v_{ea}, v_{ip}) \cdot sim(a,p) \tag{6.1}$$

In the formula, v_{ea} presents the value of the entity e for the attribute a. The used similarities have been computed by the value- and duplicate-based matcher.

Afterwards, all three matching tasks could be further iterated as for example proposed by [Suchanek et al., 2011] for matching ontologies. Applied to our algorithm, the updated similarity scores between entities and instances would serve as basis for the value-based matcher which builds the foundation of the duplicate-based matcher and so on. However, these iterations only very slightly improve the performance, as we verify in the evaluation (Section 6.5.2). This finding is consistent with the results of [Zhao and Ram, 2007] for matching databases. Hence, we do not include iterations but stop after the attribute-based refinement.

6.3 T2D Gold Standard

In this chapter, we introduce the publicly available T2D gold standard.[5] It's goal is two-fold: enabling the evaluation of web table matching systems considering literal columns and providing a foundation for the comparability of methods. Existing gold standards either focus on tables extracted from a particular web site, e.g., the Limaye gold standard [Limaye et al., 2010], or on tables about one topic [Hignette et al., 2007, Buche et al., 2013, Zhang, 2016]. Further, none of the gold standards contains tables that do not overlap with a knowledge base. The T2D gold standard includes cross-domain tables from more than one web site to cover a wide range of challenges, including the matching to datatype properties. By additionally comprising non-overlapping tables, the gold standard reflects the matching task more realistically.

To create the T2D gold standard, web tables from the WDC WTC 2012 have been manually annotated with correspondences to DBpedia. First, we present the requirements we demanded from the gold standard. Second, we introduce how we annotated the tables whereby the selection of the web tables is an important step. Further, the characteristics of the T2D gold standard are depicted and discussed.

6.3.1 Requirements

A gold standard to evaluate matching systems covers two data sources together with the correspondences between them [Duchateau and Bellahsene, 2014]. For the task of matching web tables to knowledge bases, gold standards provide a set of web tables, a knowledge base and the according correspondences for one or more matching tasks. One important step during the generation of a gold standard is the selection of the web tables. As stated by [Bailey et al., 2006] "Collections with highly varied data are valuable since they provide opportunities for investigation across a number of interesting axes and may reflect real-world data." Thus, we defined the following requirements:

1. the set of overlapping tables in the gold standard should present a balanced sample with respect to the
    - the number of rows and columns per table
    - the covered topics
    - the number of instance and property correspondences per table,
2. it should present a realistic set regarding tables that do not overlap with the knowledge base and
3. it should contain high precision correspondences.

The annotation process is described in the subsequent section, taking the requirements into consideration.

[5] `http://webdatacommons.org/webtables/goldstandardV2.html`

6.3.2 Annotation Process

Before the tables can be annotated, the set of according tables needs to be selected. Since we want to provide a realistic subset of web tables regarding the expected amount of web tables that overlap with the knowledge base, we need to get both: overlapping and non-overlapping tables. All tables in the gold standard have been selected from the WDC WTC 2012 (Chapter 4). As knowledge base, we use DBpedia (version 2014) since it is one of the most common knowledge bases (Section 3).

Overlapping Tables We define an overlapping table as a table for which at least one instance correspondence can be found and in turn, a class can be assigned. As detected during the profiling (Chapter 5) only about 3% of the relational tables actually overlap with DBpedia. The most realistic strategy is to randomly sample tables from the corpus. We decided against a random sampling strategy due to the increased annotating time which would be required:

- For each overlapping table, 32 non-overlapping tables need to be considered.
- With on average 22 rows per table, at most 22 instance correspondences are generated per overlapping table.
- To create 26 000 instance correspondences (amount of correspondences in T2D), 40 000 tables with 840 000 rows need to be annotated.
- With 10 seconds per row, additional 291 days (8 hours per day) are necessary to only annotate the non-overlapping tables.

Although a minority of tables can be annotated with correspondences, a lot of additional time is spent on tables that anyway do not overlap with a knowledge base. Further, a random sample is not suitable to get a balanced set of tables regarding the number of rows and columns as well as the topic [Wang et al., 2012].

Hence, we decided for a selection strategy that relies on initial entity seeds. In more detail, we query the Mannheim Search Join Engine (MSJE) [Lehmberg et al., 2015] with entity names for which we are certain that at least some of them overlap with DBpedia. We extracted these entity seeds based on the query tables used to evaluate the MSJE, e.g., the list of the 500 greatest films of all time[6]. To cover additional topics, we extract seeds from tables found on Wikipedia pages. For each query table, the MSJE returns a list of tables from the WDC WTC 2012 in which the according entity labels have been found. We annotated both, tables containing only a few seed entities and tables with many of them to get a mixture of tables with different characteristics. The annotations have been assigned manually by a team of two human annotators, requiring altogether about six person weeks. In total, the set of overlapping tables consists of 233 tables with 26 124 instance,

[6]`http://www.listchallenges.com/empire-magazines-500-greatest-films-of-all-time`

658 property, and 233 class correspondences. For each table element, at most one correspondence is created to be compliant with the one-to-one mapping.

Non-Overlapping Tables As stated by [Cafarella et al., 2008a] and confirmed by our findings, only about 1% of all web tables are relational. For all non-relational tables, we automatically assume that they do not share correspondences with a knowledge base. However, the classification into relational and non-relational tables is not perfect and about one-third of the decisions are misclassifications, cf. Section 4.2. Further, not all relational tables contain an entity label column which is in our definition required to match the table to a knowledge base. In a set of 1 000 randomly chosen tables that have been classified as relational, only about 30% of the tables contain an entity label column. Thus, when matching tables from a relational web table corpus, we have to be aware that about two-third of the tables are actually not mappable, independent of whether their topic is covered by the knowledge base. To address this issue in the T2D gold standard, we added 546 randomly chosen tables that have been determined as relational but are not useful for the matching task. This roughly corresponds to the expected amount of tables that are not valuable for this task. By adding the non-overlapping tables, the gold standard reflects the matching task more realistically compared to the existing gold standard.

Altogether, the gold standard contains 779 tables including both, overlapping and non-overlapping tables. The set of overlapping tables is also referred to as first version of the gold standard.[7] In return, the set of overlapping and non-overlapping tables is second version of the T2D gold standard.

6.3.3 Statistical Description

In this section, we provide the statistical description of the T2D gold standard. It includes basic characteristics about the table. Further, the sets of instance, property and class correspondences are depicted and analyzed.

Table Characteristics Table 6.1 provides basic statistics about the size of the tables that are contained in the T2D gold standard. The number of rows varies between 3 and 5 000 while the number of columns is between 2 and 30. Thus, small as well as large tables are included in the gold standard. For small and narrow tables, it is more difficult to decide whether a table can be matched to a knowledge base since only little evidence is available. If large and broad tables are incorrectly matched, it has a larger influence on the resulting performance of the matching system. Further, since only about one-third of the tables are overlapping, a matching method needs to find a way to not assign correspondences to non-overlapping tables to obtain acceptable results.

[7] `http://webdatacommons.orgwebtables/goldstandard.html`

Table 6.1: Statistics about row and column distributions in the T2D gold standard.

	Minimum	Maximum	Median	Average
Rows	3	5 000	17	84
Columns	2	30	4	5

For the matching itself, the characteristics of the overlapping tables are more interesting. Hence, Table 6.2 provides statistics about the set of overlapping tables. Altogether, almost 29 000 rows and 1 100 columns are covered within the tables. The average and median number of rows is about 150, for columns it is close to 5. Thus, the tables cover a larger amount of rows than arbitrary tables in WDC WTC 2012 where the average amount of rows is about 10. One reason is that overlapping tables actually contain relational information that in turn is often covered in larger tables. For example, a table about countries does usually not include just a few countries but at least the countries of a certain continent or even all countries of the world. Regarding the requirement of a balanced sample with respect to the number of rows and columns per table, we can see that with a standard deviation of 128 rows and almost 2 columns, also the set of overlapping tables shows variations in their sizes. Another interesting characteristic is that almost all rows, i.e., 92%, can be mapped to a DBpedia instance while this only holds for about half of the columns. Thus, a method needs to be more careful whether or not to create a correspondence for an attribute.

Table 6.2: Statistics about the set of overlapping tables in T2D.

	#	Average	Median	SD	#Corr.	Mapped Ratio
Rows	28 595	157.13	137	128.01	26 162	0.92
Columns	1 163	4.95	5	1.79	658	0.57

Figure 6.6 and Figure 6.7 present the associated row and column distributions. Altogether, about half of the tables cover less than 100 rows and the other half more than 100 rows. About 5% of the tables include at most 10 rows, at the same time only 6% of the tables have more than 300 rows. Regarding the distribution of columns, 97% of all columns contain less than 10 columns, almost one forth comprise exactly 4 columns.

Instance Correspondences In total, 26 124 rows cover entities that correspond to instances in DBpedia. To give an overview of the corresponding instances, we determine the DBpedia class to which an instance belongs and group them by their superclass, resulting in seven categories. For example, the category *Organisation* contains instances about companies, universities, and political parties. Figure 6.8 shows the distribution per category. About one-fourth of the entities link to populated places, followed by works. The remaining correspondences are distributed among various classes including *Organisation* or *Species*.

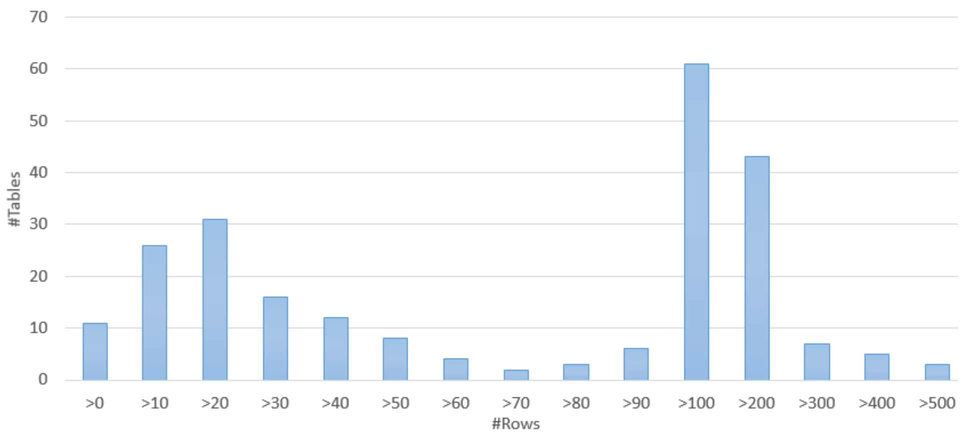


Figure 6.6: Distribution of rows in overlapping tables in T2D.

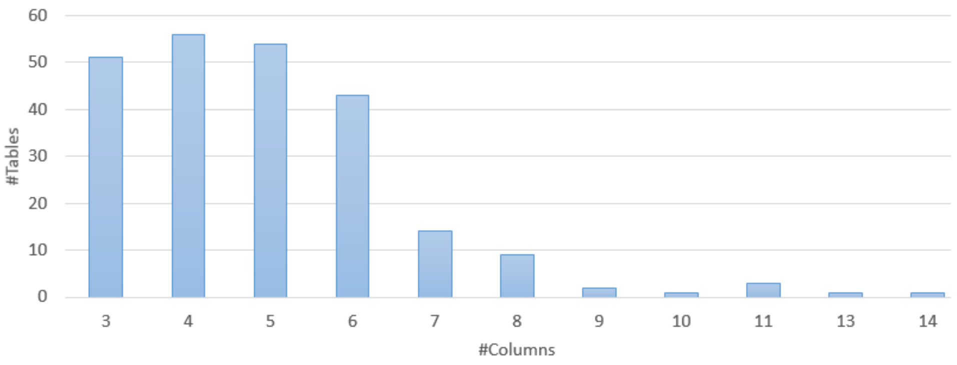


Figure 6.7: Distribution of columns in overlapping tables in T2D.

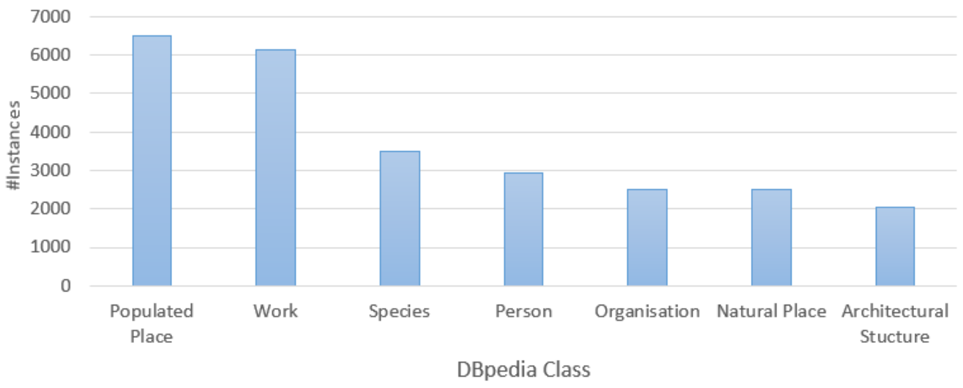


Figure 6.8: Number of instance correspondences per category in T2D.

Depending on the use case, especially the rather unknown, so called long tail entities, can be interesting. For example, to augment a knowledge base, nearly all facts for popular instances will be available in contrast to the facts of long tail ones. Further, we assume that popular instances are described in more detail in a knowledge base, thus they pose less challenges to matching systems. To provide a balanced sample, the gold standard should include both, head and tail entities. Figure 6.9 indicates how popular the instances in T2D are. The popularity is calculated based on the Wikipedia inlinks[8] of the according Wikipedia article describing the particular instance. If an article has many inlinks, we assume the instance to be a head entity. In contrast, a low amount of inlinks indicates that the instance can be found in the long tail. 50% of the instances have at least 177 inlinks in Wikipedia. In comparison, the instance with the highest score in the gold standard is *United States* with $388\,129$ inlinks which is even not depicted in the figure due space constraints. While the most popular instances are countries, smaller places like airports or mountains as well as video games can be found among the less popular instances. Altogether, T2D covers a mixture of head and tail entities.

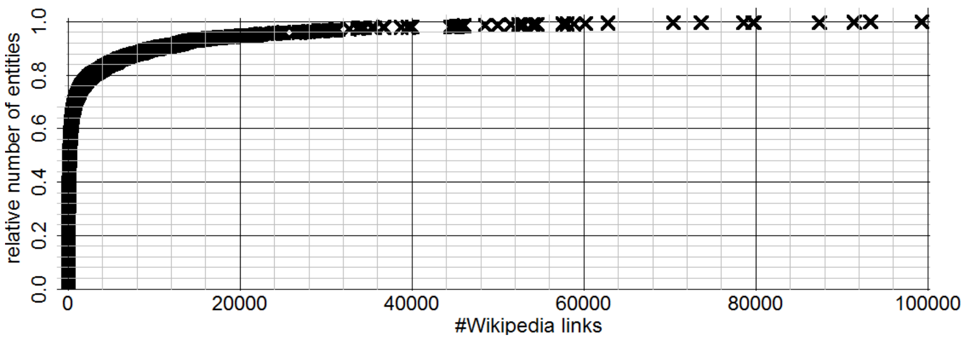


Figure 6.9: Popularity of the linked instance in T2D.

Property Correspondences Among the 658 property correspondences, 233 refer to the *rdfs:label* property since they link the entity label column. The remaining correspondences are spread over 188 tables. This implies that 20% of the tables do not have an overlapping property such that a method cannot rely on any property matching results. In total, 112 distinct properties are included in the correspondences. The most commonly referred properties are listed in Table 6.3. Most often, the release date of a work and the elevation of a place are linked. This is in line with the finding that many instances point to works and places (Chapter 5). However, only 6% of the correspondences link to the most common property *dbo:releaseDate* which indicates the variety of properties that can be found in T2D.

One aspect that influences the matching ability of the columns is its data type. In the web table corpus, about 68% of the columns are of data type string, 27% of data type numeric, and 3% of data type date. A similar distribution can be found in T2D: 63% strings, 25% numeric, 12% date. The larger amount of date columns is explainable due to the amount of tables about works since they often include at

[8]The DBpedia spotlight URI count has been used `https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Internationalization-%28DB-backed-core%29`

Table 6.3: Top 5 linked DBpedia properties in T2D.

Property	#Correspondences
dbo:releaseDate	6%
dbo:elevation	6%
dbo:populationTotal	5%
dbo:location	5%
dbo:industry	4%

least one column with a date. The average number of property correspondences ranges from 1.5 (person) to 4.1 (populated place). Hence, for tables about persons a matching algorithm cannot rely on large overlaps on the schema level.

Class Correspondences Figure 6.10 shows the number of tables per superclass. Although the picture is similar to the distribution of instances, some more characteristics can be derived. The average number of instance correspondences for tables ranges from 65 (organizations) to 250 (species) per table. Thus, when matching tables about species usually more information is provided based on the number of instances than for tables about organizations.

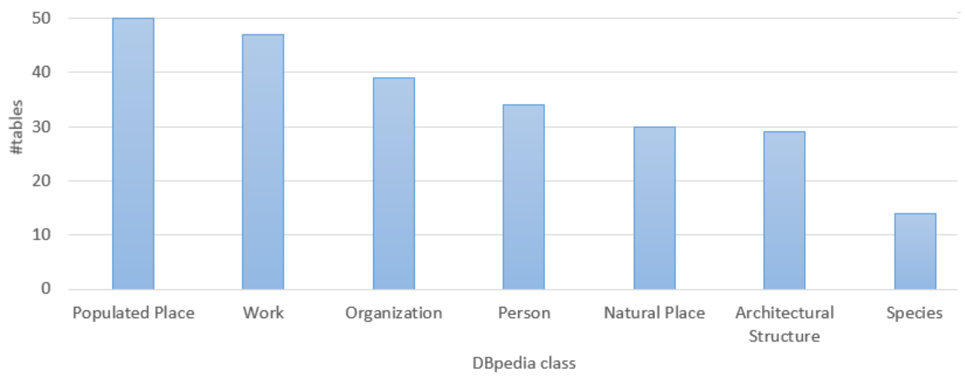


Figure 6.10: Number of class correspondences per superclass in T2D.

With the T2D gold standard, we see all requirements as fulfilled so a wide range of challenges is covered. T2D provides a balanced sample regarding the overlapping tables and presents a more realistic scenario by including a set of non-overlapping tables. Further, matching systems that include correspondences to datatype properties can now be evaluated. An overview of other gold standards together with a comparison will be presented in Section 6.4.4.

6.4 Related Work

Previously, we introduced an algorithm to match web tables to knowledge bases that tackles all matching tasks. In this section, we present strategies proposed by others for interactively handling schema and data matching. Further, we introduce existing gold standards for matching web tables to knowledge bases. The results of the matching systems are compared to T2K Match in the evaluation (Section 6.5).

6.4.1 Information Extraction

One field of research with similar objectives is information extraction which gathers information usually from unstructured text. The underlying idea is that in a sentence, entities as well as relations between the entities are mentioned. While information extraction (IE) usually focus on a specific domain, open information extraction (OIE) extracts information independent of the topic. Open information extraction system like NELL [Carlson et al., 2010] or REVERB [Fader et al., 2011] gather triples from unstructured text. Therefore, sentences found on web pages are disassembled: nouns represent entities and verbs the relations between the entities.

Since neither correspondences for the subject and object, nor for the predicate are known, both - the schema and data matching - needs to be addressed when integrating triples into a knowledge base. [Dutta et al., 2015] match triples generated by NELL and REVERB to DBpedia with the goal of knowledge base augmentation. The matching workflow starts with a data matching module that looks for instance candidates to both subject and object. Using anchor-texts of Wikipedia articles, a candidate ranking is generated for each entity and the best candidate is chosen using a Markov logic network. Afterwards, the look up module searches for properties which connect the instances that have been assigned to the subject and object. Within an optional clustering module, relational phrases having a similar meaning are clustered such that a property can be assigned to a cluster. In the property mapping module, association rule mining is applied to find frequent rule patterns. Further, the module exploits the OIE triples that overlap with the knowledge base as evidence. Altogether, the system addresses schema and data matching by first generating instance correspondences which are used to find property correspondences. In contrast to our approach, the detected property correspondences do not influence the instance matching. With the best workflow, a precision of 0.95 for the instance and 0.86 for the property matching is reported. A system with similar goals in presented by [Fossati et al., 2017] without relying on an existing OIE system. However, open information extraction systems are not able to model the interdependence among the table elements [Lu et al., 2013] which provides essential information for matching web tables.

6.4.2 Matching Databases and Ontologies

In the past, schema and data matching have often been considered as two separate problems. A first step towards connecting schema and data matching has been done by introducing the duplicate-based schema matching [Bilke and Naumann, 2005]. Thus, the data itself is taken into account and not only characteristics of the data are considered. However, the duplicate-based schema matching does not address the data matching, it solely uses duplicates for the schema matching.

As one of the first, [Zhao, 2007] presented the idea of solving schema and data matching in an integral fashion. In another paper, [Zhao and Ram, 2007] deploy an approach that actually tackles combined schema and data matching. They use two catalogs of books and two university databases as data sources, both covering partially overlapping entities and attributes. At first, the attributes are clustered using features like the attribute labels, data patterns, and statistics. Afterwards, a classifier is learned to match the entities which uses the distances on the attributes identified during the clustering together with the according values. During the classification, again insights are gained about the attribute correspondences and then used to adapt the candidates. Finally, additional iterations can be triggered but they have not shown to improve the results. In contrast to our approach, they start with the schema matching that does initially not rely on the entities but on characteristics of the attributes. These characteristics are difficult to exploit if the sources are not of the same input type as in our case. The evaluation of the method is restricted to databases from two domains. Deploying it on data sources from unknown domains would require adaptions.

The domain-independent system PARIS - Probabilistic Alignment of Relations, Instances and Schema - performs simultaneous schema and data matching focusing on ontologies as input sources [Suchanek et al., 2011]. Aligning relations represents the task of matching properties while the term schema alignment is used for the class matching. To perform the three matching tasks, a probabilistic model is applied. Logical rules are used to model equalities and subsumptions. For example, the probability of an equivalence between two classes is proportional to the number of instances that belong to both classes according to the already established instance correspondences. The algorithm starts by computing the probabilities of equivalences of instances, continues with the probabilities for relations and iterates the two steps until a fixpoint is reached. Relying on the correspondences of instances and relations, the class matching task is performed which in turn does not have an influence on the other tasks. PARIS has been evaluated on large-scale ontologies, i.e., YAGO and DBpedia, which is not the case for the only two other systems Rimom [Li et al., 2009] and Iliads [Udrea et al., 2007] that perform an integrated schema and data matching for ontologies. As result, F-measure scores of 0.81 for the instance, 0.92 for property, and $0.84 - 0.91$ (depending on the direction) for class matching task are stated. In contrast to our approach, PARIS is designed for matching ontologies with a rich class and property structure. This is a difference to web tables which can be very noisy and usually cover only a few entities and attributes. We will discuss the results of PARIS and the different challenges in more detail in Section 6.5.3.

6.4.3 Table Augmentation

Table augmentation aims at extending a query table. This can either mean to find values for a given set of entities and an attribute or to detect attributes that make relevant additions to the query table. For example, given a query table with country names, the according capitals should be added. Therefore, the entities described by the given attribute are searched in a web table corpus.
[Yakout et al., 2012] propose the InfoGather system that augments a query table using a corpus with 573 million tables. To find related entities, the query table is matched to all tables in the corpus. Due to the large amount of comparisons, the matching is restricted: only tables with overlapping entity labels having exactly the required attribute label are considered. Using the computed similarities between the tables as basis, a holistic matching is applied to also find indirect matches. We will go into detail about this step is Chapter 8.
[Bhagavatula et al., 2013] introduce WikiTables which identifies attributes that are relevant additions to the query table. As web table corpus, they focus on tables extracted from Wikipedia.
The Mannheim Search Join Engine introduced by [Lehmberg et al., 2015] can do both: finding values of a particular attribute and adding relevant attributes to a query table. Besides web tables, also semantic annotations and LOD sources are taken into account. The matching between the query table and the corpus mainly bases on the label similarity of the entities. To consolidate values, a duplicate-based schema matching is used.

Augmenting tables shares the task of matching web tables but for a different goal. Thus, similar challenges have to be overcome like finding out which entities are similar to the query table entities. The main difference between table augmentation and the matching of web tables to knowledge bases is that the augmentation focuses on the integration rather than the interpretation such that the meaning of the augmented attributes in unknown. To combine both ideas, the correspondences between web tables and knowledge bases could be used as an intermediate schema [Lehmberg et al., 2015].

6.4.4 Web Table to Knowledge Base Matching Gold Standards

Since the matching of sources like databases and ontologies has been performed for years, various gold standards exist for these tasks. Examples are the gold standards within the Ontology Alignment Evaluation Initiative. Once a year, ontology matching systems are evaluated using the gold standards to compare their performances and to assess the strengths and weaknesses of matching techniques. Since web tables have just recently gained attention, such initiatives do not exist yet.

One of the first gold standard has been introduced by [Hignette et al., 2007]. In this gold standard, 60 tables have been extracted from scientific publications found on the Web. The attributes of the tables have been manually annotated with properties of a food microbiology domain ontology. [Buche et al., 2013] extended the gold standard by adding tables about chemical risks and aeronautics. [Zhang, 2016] propose two gold standards created from well-known sources: IMDB and MusicBrainz. From IMDB, 7 000 tables describing films are considered, from MusicBrainz 1 400 tables about songs. All these gold standards have in common that they cover tables about exactly one topic.

Already knowing the topic of the tables is a great advantage but does not present a realistic scenario. The first gold standards covering multiple topics have been proposed by [Syed et al., 2010] and [Mulwad et al., 2010b] but only include 5 and 15 tables, respectively. A larger gold standard has been developed by [Limaye et al., 2010]. The Limaye gold standard consists of four subsets: WikiManual (36), WebManual (317), WebRel (30) and WikiLink (6 085), altogether resulting in 6 522 tables annotated with elements from the YAGO knowledge base. WikiManual and WikiLink cover tables from Wikipedia articles. The other two sets consist of tables from a web table corpus that has been queried with the tables from Wikipedia as seeds. While the tables of WikiLink are only annotated with instances, the WebRelations set contains solely property annotations.

The gold standard introduced by Limaye et al. has been adapted by other researchers. [Venetis et al., 2011] use a random sample of 168 tables from WikiManual and WebManual. [Mulwad et al., 2013] use a few hundreds of tables from all four subsets and annotated them with both, correspondences to DBpedia and YAGO. Recently, [Zhang, 2014a, Zhang, 2016] presented two additional subsets, Limaye112 and Limaye200 that contains 112 and 200 randomly chosen tables from the four subsets, annotated with correspondences to Freebase. The Limaye gold standard and its adaptions suffer from the following facts:

- **Variety of Sources** The majority of tables originates from Wikipedia, so characteristic variations are restricted, e.g., most tables have attribute labels.
- **Datatype Properties** Partly except for the Limaye200 gold standard, annotations to datatype properties are not included.
- **Non-Overlapping Tables** Only tables that actually overlap with the knowledge base are considered which does not reflect the distribution realistically.

In contrast to other gold standards, the Limaye gold standard covers a comprehensive amount of cross-domain tables. However, due to the mentioned restrictions, the range of challenges for web table to knowledge base matching systems is limited. The T2D gold standard provides a wider range of challenges by focusing on a larger amount of sources and including correspondences to datatype properties as well as non-overlapping tables.

6.4.5 Web Table to Knowledge Base Matching

A set of systems matching web tables to knowledge bases has been proposed over the last years. In this section, we will introduce them by focusing on how the matching is performed, which matching tasks are addressed and to which extent they influence each other. Within subsequent chapters, we concentrate on different aspects like the choice of features in Section 7.3. In Table 6.4, an overview of all methods matching web tables to knowledge bases is depicted, stating which matching tasks they consider.

Table 6.4: Overview of the tasks addressed by approaches matching web tables to knowledge bases.

Approach	Instance	Property	Class
[Hignette et al., 2007]	×	✓	✓
[Buche et al., 2013]	×	✓	✓
[Zwicklbauer et al., 2013]	×	×	✓
DSL [Pham et al., 2016]	×	✓	×
TAIPAN [Ermilov and Ngomo, 2016]	×	✓	×
[Quercini and Reynaud-Delaître, 2013]	✓	×	×
[Bhagavatula et al., 2015]	✓	×	×
[Efthymiou et al., 2017]	✓	×	×
[Venetis et al., 2011]	×	✓	✓
[Fan et al., 2014]	×	✓	✓
[Chu et al., 2015]	×	✓	✓
[Muñoz et al., 2013]	✓	✓	×
[Syed et al., 2010]	✓	✓	✓
[Mulwad et al., 2010b]	✓	✓	✓
[Limaye et al., 2010]	✓	✓	✓
[Mulwad et al., 2013]	✓	✓	✓
TableMiner+ [Zhang, 2016]	✓	✓	✓

Methods focusing on a particular domain As one of the first, [Hignette et al., 2007] and later [Buche et al., 2013] propose methods to perform the property and class matching task. Therefore, they used a domain-specific biological ontology. Since the approaches focus on a particular topic, specific knowledge encoded in the domain ontology can be exploited, e.g., the property expressing the pH value can only cover discrete values between 0 and 14. Such knowledge cannot be considering when matching web tables to cross-domain knowledge bases not focusing on a particular topic.

Methods focusing on a subset of tasks A set of approaches matches web tables to cross-domain knowledge bases but only considers a subset of matching tasks. [Zwicklbauer et al., 2013] solely performs the class matching task by applying a simple majority vote as we also include it in the algorithm: instance candidates are gathered for each entity and the instances vote for the class they belong to. Conversely [Pham et al., 2016, Ermilov and Ngomo, 2016] only consider the property matching task while [Quercini and Reynaud-Delaître, 2013, Bhagavatula et al., 2015, Efthymiou et al., 2017] take the instance matching task into account. For the property and class matching task, [Venetis et al., 2011] use an isA database that has been extracted from the Web using lexical patterns. The database contains statistics of co-occurrences like how many times does "animal such as dog" occur where "such as" presents the extracted pattern. Using the database, a maximum likelihood inference model predicts the best class for an attribute to be the one maximizing the probability of seeing all the values in the attribute given that class. Similarly, [Wang et al., 2012] utilize the probabilistic database Probase that is constructed by Hearst patterns. For each table, the set of entity labels and attribute labels is used as query for Probase to get a list with the most probable classes. Both methods heavily rely on the probabilistic statistics of the database that are not available in knowledge bases like DBpedia or YAGO. Further, [Fan et al., 2014] and [Chu et al., 2015] use crowd-sourcing for the property and class matching.

Methods focusing on a set of web tables Besides the restrictions on a particular topic, a subset of tasks and/or the usage of knowledge that is not available in all knowledge bases, also constraints to the web tables itself can be posed. [Muñoz et al., 2013] extract RDF triples from tables found in Wikipedia articles. For the instance matching task it is assumed that the cells contain internal links to other Wikipedia articles which in turn can be transferred to URIs identifying DBpedia instances. Afterwards, DBpedia is queried with the mapped instances to get all possible properties relating those instances. The work is later extended by adding a machine learning process to filter triples that are likely to be incorrect [Muñoz et al., 2014]. Having internal links simplifies the instance matching task but the method is not universally applicable.

Domain-independent methods addressing all tasks Only a few systems address all three matching tasks. Their results are depicted in Table 6.5. As mentioned during the definition of the matching tasks, all listed approaches focus on named entity attributes and do not consider datatype properties.

[Syed et al., 2010] and [Mulwad et al., 2010b] use Wikitology, a hybrid knowledge base of structured and unstructured information extracted from Wikipedia including classes from DBpedia, YAGO, WordNet, and Freebase. The method first queries the Wikitology to get instance candidates. Similar to our approach, the class is then derived based on the initial instance candidates and in turn under consideration of the class, an instance candidate refinement is performed. In the end, the same strategy as utilized by Muñoz et al. is applied which queries the knowledge base for properties that relate two entities found in the same row. Wikitology provides knowledge that is not available in other knowledge bases.

[Limaye et al., 2010] use a probabilistic graphical model with interrelated random variables. The goal is to provide an assignment for all three tasks that maximizes the joint probability. First, a model is learned that determines the best aggregation weights for a variety of features including the similarity of entity and attribute labels. These features and weights are used to define potential functions over subsets of variables, and the product of these potential provide the joint distribution over all variables. Thus, the three tasks influence each other all the time.

[Mulwad et al., 2013] extend Limaye's work by using a more lightweight algorithm since computing the joint probability distribution is expensive. Further, the Wikitology is used to get instance candidates so that the initial selection does not only rely on a label comparison.

Table 6.5: Overview of the results achieved by matching approaches addressing all three matching tasks.

		F-measure		
System	Gold Standard	Instance	Property	Class
[Mulwad et al., 2010b]	15 tables	0.66	0.25	0.77
[Syed et al., 2010]	5 tables	0.77	1.00	0.90
TableMiner+ [Zhang, 2016]	Limaye	0.84	0.76	0.75
[Limaye et al., 2010]	Limaye	0.84	0.58	0.45
[Mulwad et al., 2013]	Limaye	0.76	0.84	0.55

According to [Zhang, 2014b], the inference used by a probabilistic graphical model is exhaustive but unnecessary. Therefore, Zhang presents TableMiner, a system with a two-phased bootstrapping strategy instead of an inference model. During the first phase, it learns an initial interpretation using partial data from the table whereas the second phase uses the initial interpretation as constraint to interpret the rest of the table. In more detail, a sample of entities is disambiguated by comparing entities to instances represented as bag-of-words. Based on the candidate instances, a majority vote decides which classes are candidates for columns. Afterwards, in the update phase, the initial candidates for instances and classes are improved by considering not only a sample of rows. As soon as no changes in the similarities are observed, TableMiner continues with the property matching task. For pairs of attributes, the knowledge base is queried for properties relating the assigned instances. Since the property matching task is applied after the other two tasks are finished, it does not have any influence on them. An extended version, TableMiner+ [Zhang, 2016], has recently been introduced and is the only systems that additionally considers literal columns. However, only string comparison methods are used such that the applicability for columns of data type numeric or date is restricted. All in all, TableMiner+ is the system closest to T2K Match.

Altogether, only a few approaches actually consider all three matching task. Further, except for TableMiner+ only columns including named entities are matched at all. Some methods heavily rely on background knowledge, e.g., the Wikitology or probabilistic databases, which is not available in every knowledge base. Other approaches apply an inference model that tries to optimize the results of the three matching tasks simultaneously. In Section 6.5, we will compare the results of the introduced matching systems with the T2K Match method.

6.5 Evaluation

In this section, we will evaluate T2K Match experimentally. First, we present the outcomes of all three tasks when applying the algorithm to tables of the T2D gold standard. We further analyze the results in more detail by considering for which table the algorithm achieves the highest results. Hence, we get an idea of the drawbacks and the improvement potential of the algorithm. Second, the results are compared to the findings for systems that also address the schema and data matching in an integrated fashion. This includes the ontology matching system PARIS and existing approaches that match web tables to knowledge bases.

6.5.1 Experimental Setup

The code of T2K Match has been released in the T2K github repository.[9] The parameters that are used during the matching process have been determined using a genetic algorithm that focuses on high precision results. Table 6.6 lists the most important parameters that have been used in all following experiments.[10]

For our experiments, we use the set of overlapping tables from the T2D gold standard (first version) and a subset of DBpedia as reference knowledge base. The DBpedia subset consists of all classes, properties, and instances from DBpedia (English version 2014) that are frequently used and hence are promising candidates to be detected in the web tables. We use all classes as long as they cover at least 1 000 instances. To exclude too specific classes, we only consider classes up to the fourth level of the class hierarchy where the class *Thing* represents the first level. All properties are discarded unless they are frequently used by instances in the subset, i.e., if at least 5% of the instances belonging to the class for which the property has been defined. The result of this selection process is a DBpedia subset covering 94 classes, $1\,393$ properties, and about 3 million instances. This corresponds to two third of all DBpedia instances although only about 14% of the

[9]`https://github.com/T2KFramework/T2K/tree/master`

[10]Full list of parameters: `http://web.informatik.uni-mannheim.de/T2K/paramtersT2KMatch.csv`

classes are taken into account. Regarding the properties, about 50% of all properties are included in the subset.

The use of a subset is reasonable since all knowledge base elements referenced in correspondences of the T2D gold standard are covered. With the evaluation on the DBpedia subset, we are able to identify which challenges cannot be overcome with the presented method without focusing on too many details like the full class hierarchy. The same holds for the set of tables: we only consider the overlapping tables from the T2D gold standard for the experiments with the T2K Match algorithm. All existing systems that match web tables to knowledge bases are evaluated in the same way since other gold standards do not include tables that do not overlap with the knowledge base.

6.5.2 Overall Results

In this section, we evaluate the overall performance of T2K Match. Table 6.7, depicts the results of matching the overlapping tables of T2D to the DBpedia subset using T2K Match. Most important, our algorithm is able to match web tables to a knowledge base without the restrictions of only considering named entity columns and achieves acceptable results. For the class matching task, the highest F-measure of 0.94 can be attained, followed by the performance for instances with 0.82 and properties with 0.70. As indicated, we favor a high precision over a high recall because the more precise our correspondences are, the less incorrect data we have to deal with during the fusion. Thus, especially for the instance and property matching tasks, the precision is at least 0.1 higher than the recall.

Error Analysis To find explanations for the results of all tasks, we analyze different steps of the process in more detail. First, we discuss the performance of the metadata recovery to know whether especially the lower outcomes of the property

Table 6.6: T2K Match parameters.

Parameter	Value
Instance Final Threshold	0.5
Property Final Threshold	0.0
Instance Candidate Selection Top K	50
Instance Candidate Refinement Top K	100
Class Candidate Selection Top K	5
Instance Candidate Selection Threshold	0.2
Instance Candidate Refinement Threshold	0.7
Instance Label Weight	5
Instance Value Weight	1
Value Threshold	0.5

matching task stem from an insufficient metadata recovery. Second, we examine the candidate selection in more detail since it presents the basis for following steps.

Metadata Recovery Starting from the beginning of the overall process, one possible source of errors are incorrectly recognized metadata like the entity label column as well as the column data type. To estimate the extent of the influence, we analyze the entity label column and the data type detection. For the entity label detection, an F-measure of 0.94 is observed which means that for almost all tables the correct entity label column is found. This is an important finding since an incorrect entity label column mostly results in incorrect correspondence for all three tasks. Similarly, the F-measure of the attribute label detection is 0.97.

Another metadata is the data type of each column which plays an essential role for the property matching task. With an incorrect data type, the values of a column will not be compared to the values of the correct property. With the following analysis we check to which extent the data type detection is responsible for the lower property matching task performance. Table 6.8 shows the confusion matrix of the data type detection. The most common misclassification is that a numeric attribute is considered to be of data type string. This error is responsible for 58% incorrectly assigned data types.

One reason why numeric columns are classified as type string is the preprocessing. The according values cover additional words like "overall" or use different terms to indicate a missing value, e.g., "nil". Mainly, this happens if units are missed. An example is a column stating the height of a person, e.g., a value is 6'9". In this case, the metadata recovery is not able to detect that the value is a length specification given in feet and inches. The misclassification of numeric columns as dates results from the fact that numeric values are in the same range as year specifications. Example are "1946" or "7.93". The most common incorrectly detected data type is date. The reason is that dates are not recognized as such since the set of date patterns does not cover the specific format. This includes for example abbreviated dates like "oct 6 08". Further, we expect that at least the year is specified, such that values like "June 25" or "Monday" are not considered as dates.

Overall, the quality of the metadata recovery on the tables from T2D is quite high, especially higher than expected based on a random sample as described in

Table 6.7: Results of T2K Match for matching the overlapping tables of T2D to the DBpedia subset.

Task	Precision	Recall	F-measure
Instance	0.90	0.76	0.82
Property	0.77	0.65	0.70
Class	0.94	0.94	0.94

Table 6.8: Confusion matrix for results of the data type detection.

	Actual	string	numeric	date
Predicted	string	646	19	4
	numeric	0	270	5
	date	0	5	70

Chapter 4. Thus, we assume the performance of the metatdata recovery to only negligibly influence the matching results.

Candidate Selection Within the candidate selection, the candidates for each table element are determined and serve as blocking keys for all subsequent steps. Hence, if the according knowledge base element is not among the candidates, the correct correspondence will not be generated. Regarding the instance matching task, for 82% of the entities in the tables, the instance it corresponds to is found among the candidates. This can also be expressed as a pair completeness of 0.82 which presents a typical measure to evaluate blocking techniques. In turn, we know that we miss 18% of the instance correspondences. One reason is the focus on the label similarity that does not necessarily hold, e.g., due to abbreviations or alternative terms. Therefore, a great potential to increase the instance performance lies in the improvement of the instance candidate selection. Regarding the candidate selections for the other two tasks, for both a pair completeness of 0.96 is determined. The only case in which the correct class is not a candidate is if the entity label column is determined incorrectly and in turn the majority of instances does not belong to the correct class. The same holds for the property matching task since all properties are excluded that are not referred to by the instance candidates.

Performing the candidate refinement helps to get more instance candidates per entity. Figure 6.11 shows the average amount of candidates for each table from the T2D gold standard. Compared to the amount of candidates created by the candidate selection, on average 1.6 additional candidates can be found for each entity. Depending on the individual table, up to 12 ancillary candidates per entity are detected. Although an increased amount of candidates does not necessarily lead to better results, with the candidate refinement an F-measure increase of 0.08 for the instance matching task is determined.

Iterative Matching After the attribute-based refinement, iterations between the instance and property matching can be performed. As we stated in the description of the algorithm, we do not apply such an iterative approach since the results hardly change. Figure 6.12 presents the differences in F-measure for the instance and property matching task when performing further iterations. The last step in the algorithm, the attribute-based refinement refers to the second iteration since the instance candidates scores are once adapted using the weights computed

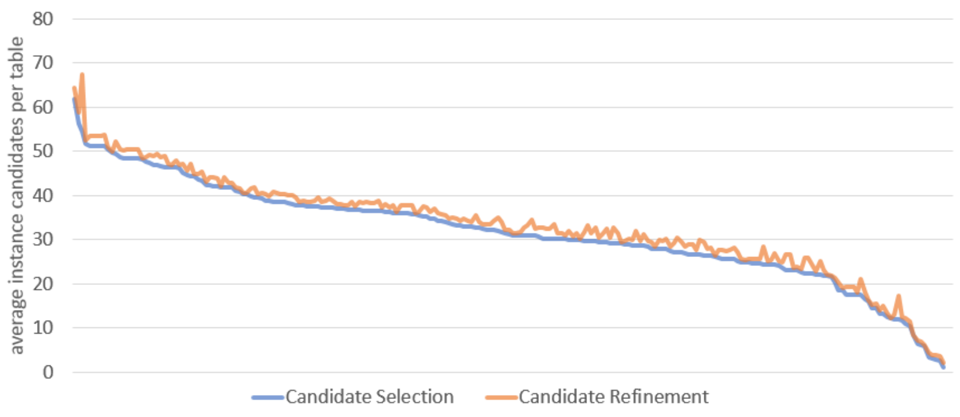


Figure 6.11: Average amount of instance candidates per entity in each table.

by the attribute scores. Altogether, the changes in F-measure are marginal, below 0.01. Hence, to dispense with iterations does not lead to deteriorated results. This is consistent with the results of [Zhao and Ram, 2007] for matching databases.

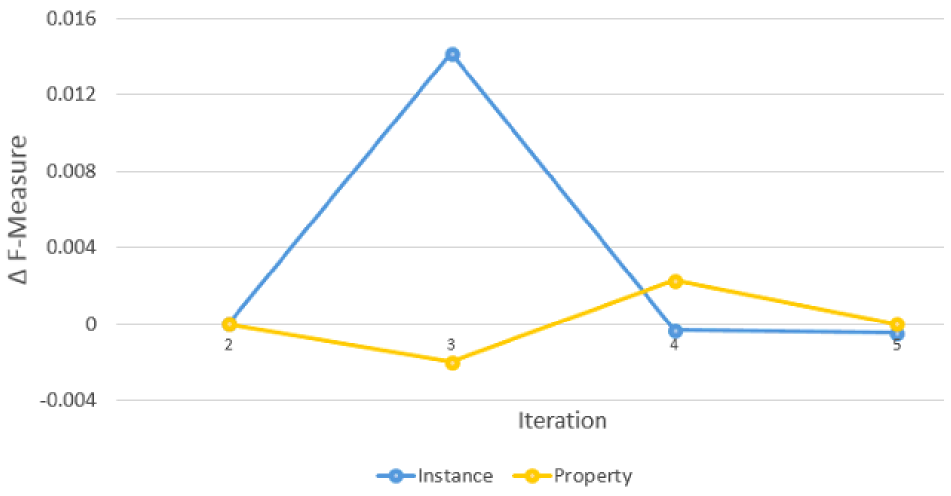


Figure 6.12: F-measure changes by applying additional iterations.

Detailed Results per Category To better discover potential issues, we analyze the results in a more fine-grained way. The achieved performance is not the same for tables describing different topics due to their individual characteristics. We analyze the differences by assigning each table to the superclass within the DBpedia class hierarchy, also referred to as category, to which the mapped class belongs. Figure 6.13 provides the overview of the achieved precision grouped by category.

The reasons for different results per category can be very specific. We present some examples which are helpful to draw general conclusions. The low prop-

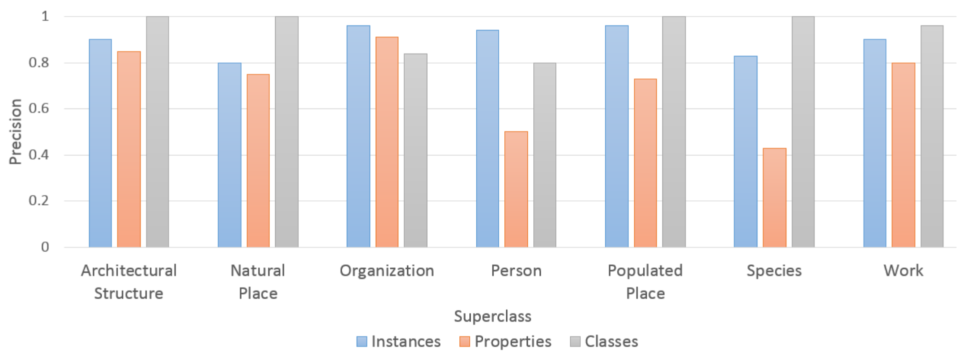


Figure 6.13: Precision of the three matching tasks according to their category.

erty precision for the person category stems from varying attribute values, e.g., the weight of an athlete might change during his/her career and the birth or death dates of historic persons are often different from source to source. Another reason which especially holds for the tables describing species is that value-based similarities can be misleading. For example, the Latin name and the genus of an animal can be similar but they do not have the same meaning, e.g., the name of the genus "melopsittacus undulatus" versus the animal name "melopsittacus" for budgerigars.
The better performance for the instance correspondences has also different reasons. In some cases, a high label similarity in combination with the restricted set of possible classes provides sufficient evidence to decide for the correct correspondences. But, for example, for companies, the precision for tables with no attribute overlap is only 0.45. For tables having one property correspondence, the precision increases to 0.97. Hence, without an overlapping property it is often not possible to assign the correct instances. Using the terminology of [Doan et al., 2004b], a schema-related evidence is not available. If only one attribute corresponds to a property, the results can be drastically improved since the schema-related evidence can be taken into account. However, for entities with ambiguous names, even an overlapping attribute might not be sufficient since it is not discriminative enough. As example, the entity "lake geneva" is mapped to the DBpedia instance "Geneva_Lake" instead of "Lake_Geneva" due to missing or insufficient schema-related evidence. Another problem is related to our concept of entity label columns: tables about species often use inconsistent entity naming, sometimes the Latin name and sometimes the English name is used as entity label, e.g., "tufted titmouse" and "baeolophus bicolor" are the English and Latin name for the same bird species. Thus, the instance index will only in some cases return the corresponding instances.
Incorrectly detected classes are due to incorrect entity label columns and discrepancies regarding the class hierarchy. For example, a table about soccer players is mapped to the more general class *Person* since for every entity in the table a person but not necessarily a soccer player can be detected. Since persons are one of the largest classes in DBpedia with many subclasses, the assignment of too general or too specific classes occur most often for tables with this topical category.

Conclusions from the error analysis With the T2D gold standard that covers tables with different characteristics, we are able to identify which challenges can only partly overcome with the T2K Match algorithm. Moreover, using a subset of DBpedia and solely overlapping tables provides a restricted, not realistic setting but helps to concentrate on the most important issues. The following conclusions can be drawn from the error analysis:

- **Metadata Recovery & Iterative Matching** Errors made during the matadata recovery and additional iterations only slightly influence the results.
- **Candidate Selection** For the schema matching tasks, property and class matching, the correct element in DBpedia is almost always among the candidates. For the instance matching task, the candidate selection is a potential for improvement but nevertheless for over 80% of all entities the according instance is a candidate.
- **Deciding for the correct correspondence** Since the correct knowledge base elements are mostly among the candidates, the main challenge is to decide for the correct candidate. Within the results per category we determined that mostly the lack of sufficient evidence leads to incorrect correspondences since terminological but also conceptual heterogeneities cannot be overcome. For example, if only evidence based on the label comparison can be used and no schema-related evidence can be gathered due to the missing attribute overlap, taking the correct decision is difficult.

All errors are aggravated if the table covers only a few entities since all steps rely in the first place on the instance candidates. Depending on the individual table characteristics, the mentioned issues have a larger influence. We use these findings to enhance the matching in the subsequent chapters.

6.5.3 Comparison with State-of-the-Art

In order to estimate the performance of T2K Match, we compare the results to the results of state-of-the-art approaches. We start with the comparison of methods for the instance matching task to see whether addressing the data matching in an integrated fashion is beneficial. Further, we highlight differences to the results of the PARIS system which has been developed for matching ontologies. Finally, we examine whether our results are in line with the reported performances of comparative systems that match web tables to knowledge bases.

Comparison with individual schema and data matching methods At first, we compare the achieved results to the results of methods that focus on either schema or data matching. Through this, we want to find out whether addressing the schema matching together with the data matching in a combined way is useful. We apply the following methods:

- **Candidate Selection** uses the best candidate for each entity that is returned by the index. As described, the similarity only relies on the labels according to the internal Lucene ranking. The results serve as a reference value for the improvements achieved by the subsequent steps of the algorithm.
- **DBpedia Lookup** We query the DBpedia Lookup service[11] for each entity label and choose the first result as correspondence. In addition to the string comparison, it considers the link in-degrees of the Wikipedia pages that describe the instances. Thus, it emulates the strategy of always choosing the most common sense of a label.
- **LogMap** is an ontology matching tool that uses lexical and structural heuristics.[12] For matching ontologies, it has been evaluated as one of the best and most efficient systems [Dragisic et al., 2014].

Table 6.9: Results of methods focusing on the instance matching task.

Method	Precision	Recall	F-measure
Candidate Selection	0.53	0.53	0.53
DBpedia Lookup	0.79	0.73	0.76
LogMap	0.57	0.89	0.70
T2K Match	0.90	0.76	0.82

Other data matching approaches like the SILK framework are more difficult to apply on the web tables since they rely on schema correspondences that are not known. Table 6.9 presents the comparison of the three methods. T2K Match outperforms all techniques concerning precision and F-measure. The candidate selection only achieves an F-measure of 0.53 which indicates that purely relying on similarities between the labels does not lead to sufficient evidence.Additionally taking the popularity of an instance into account performs better. The DBpedia Lookup achieves even better F-measure scores than LogMap. Mainly, ontology matching tools are designed to match sources with rich structures that are not given by web tables. Altogether, acceptable results are obtained if the most common sense of a label is assigned but considering values and accordingly schema-related evidence is required to achieve F-measures scores above 0.8. We will analyze the utility of different features in more detail in Chapter 7.

For the schema matching, i.e., property and class matching task, it is more difficult to receive comparable results of other methods. As most of the schema matching tools put a large emphasis on the attribute label, we tested how much correct correspondences can be detected by a method that purely relies on the comparison between the attribute and the property label. Therefore, we apply the Jaccard similarity. With the best possible threshold of 0.3, a precision of 0.14 with a recall of

[11]`http://wiki.dbpedia.org/projects/dbpedia-lookup`
[12]`http://www.cs.ox.ac.uk/isg/projects/LogMap/`

0.39 is achieved. Regarding the class matching task, since the tables do not have an explicit label, we cannot use it to employ a similar method.

In summary, individually addressing each matching task results in lower performances, if applicable. Hence, it is reasonable why systems including T2K Match handle the task of matching web tables to knowledge bases in an integrated fashion.

Comparison with PARIS PARIS performs simultaneous schema and data matching with ontologies as input sources. Among the few ontology matching systems that perform a combined approach, it is the only one that has been applied on large-scale cross-domain datasets. The PARIS system has been evaluated within four different scenarios, one is the matching of YAGO and DBpedia. For this scenario, the authors report an F-measure of 0.81 for the instance matching task with a precision of 0.92/1.0 for the property and of 0.84/0.94 for the class matching task. Two different precision scores are mentioned since the performance depends on the direction whether YAGO is matched to DBpedia or the other way around. The evaluation of the instance matching task can be performed automatically by comparing the common Wikipedia identifiers. Contrary, the property and class matching tasks have to be evaluated by sampling and manually annotating correspondences. That is also why a recall estimation is not provided.

Compared to our results, the performance of the instance matching task is pretty similar: both achieve precision of 0.9 with a recall of 0.75. The same applies for the class matching tasks. Nevertheless, we have to keep in mind that we only consider a subset of DBpedia which facilitates matching due to the restricted amount of available knowledge base elements. Contrary, the results of the property matching task significantly differ by at least 15%. One reason is that ontologies provide a formal schema, such that no errors of the metadata recovery exist and we assume more consistent formatting etc. since the values are automatically extracted. Another reason is the structural richness and the size of the ontologies. While the ontologies describe millions of instances with a variety of properties, web tables usually only contain a few entities with a few attributes. In more detail, DBpedia covers on average 11.44 properties to describe an instance, whereas the number of attributes used in web tables is around 4. Similarly, each class covers up to millions of instances, while a web table contains far less entities. Thus, more evidence can be collected during the matching. This finding is strengthened by the results reported by the authors: for instances described by more than 10 facts, the precision and recall jumps to 0.97 and 0.85, respectively. To emphasize the outcome, we run PARIS on the overlapping tables of T2D. For the instance matching task, an F-measure of 0.07 is reached due to a pretty low precision of 0.04. Both other tasks achieve similarly low performances. The results of PARIS on T2D are confirmed by [Efthymiou et al., 2017]. Although it might be possible to improve the results by tuning the parameters, it shows that out-of-the-box, a system focusing on rich ontologies is not capable to deal with web tables.

Comparison with other web table matching systems Most existing systems apply probabilistic models to match web tables to knowledge bases. Three such systems have been introduced by [Limaye et al., 2010], [Mulwad et al., 2013] and [Venetis et al., 2011]. Except for the TableMiner+ system [Zhang, 2014a], columns covering literal values are not considered. All four approaches have been evaluated on tables from the Limaye gold standard, mapped to the knowledge bases YAGO, DBpedia, or Freebase. While Limaye et al. and Venetis et al. use the same set of tables, Mulwad et al. recreated the gold standard by running their system with low threshold and verify the correctness of the returned correspondences. As stated in Section 6.4, the majority of the tables in the Limaye gold standard originates from Wikipedia where other conditions hold than for web tables coming from arbitrary website. For example, Wikipedia tables always include a header row.

Although the evaluations have been performed on different datasets, we try to draw some general conclusions. Since neither the implementations nor the results are publicly available, we can only speculate about the different outcomes.

- F-measure scores for the instance matching task vary between 0.76 and 0.84, for the property matching task between 0.55 and 0.84, and for the class matching task between 0.45 and 0.69.
- The results of the property matching are below the instance matching results.
- For the instance and property matching task, T2K Match reports F-measure scores in the same range.

The higher class matching results achieved by T2K Match mainly results from two facts: we only consider a subset of DBpedia and other knowledge bases like YAGO have in contrast to DBpedia a more fine-grained structure. Among other features, all systems exploit instance candidates in order to determine the class. Thus, if incorrect instances are assigned, the class assignment will most likely be incorrect, too. Especially due to the small sizes of tables, this happens easily. Further, the more classes exist, the more difficult it gets to identify which particular class fits best. Since each superclass covers all instances of its subclasses, taking the class to which the majority of instances belong to is not sufficient. Considering the matched properties helps but only if these properties are specific for a class. In summary, especially by having a look at the outcomes of other systems, we see that the class matching task is not as simple as our results indicate.

Altogether, independent of the dataset, the achieved performances of other web table to knowledge base matching systems mostly coincide with our results. However, we can see that the outcomes still leave room for improvement for all three matching tasks.

6.6 Summary

In this chapter, we introduced the matching of web tables to knowledge bases. First, we pointed out the challenges of using web tables as a source. Some challenges like the amount of data and its variety have to be dealt with whenever large amounts of sources are integrated. Others are more specific for web tables like the size of the tables or the lack of a formal schema.

To match web tables to a knowledge base, we described the three matching tasks that need to be addressed: instance, property, and class matching. To be able to perform all matching tasks, we presented a web-scale algorithm, T2K Match, that combines the tasks in an integrated fashion. The evidence gathered from each matching task has an influence on the other tasks. At first, candidates for each table element are detected which serve as basis for subsequent steps. In contrast to all existing works, except for [Zhang, 2016], the algorithm considers literal columns covering numeric values or dates.

As one contribution of this thesis, we proposed the T2D gold standard which covers correspondences for all three tasks. In contrast to existing gold standards, T2D shows characteristics that especially focus on providing a dataset covering a wide range of challenges:

- The gold standard is publicly available.
- It contains large amounts of correspondences for all three matching tasks, including correspondences to datatype properties.
- The tables are extracted from more than one website to incorporate tables with different characteristics.
- It includes tables that do not overlap with the knowledge base.

The T2D gold standard is used to evaluate T2K Match. In summary, F-measure scores above 0.8 for the instance and 0.94 for the class matching task are achieved. The performance of the property matching task is lower with an F-measure of 0.7. Hence, the property matching seems to be the most difficult task for T2K Match. We analyzed for which topics the tasks perform best and which differences can be observed. In general, the lack of sufficient evidence poses the greatest challenge. This is caused by heterogeneities such as the usage of different terms and by the small amount of information which is available within each individual table. To estimate the results, we discussed the performances reported by other web table to knowledge base matching systems. Although the evaluation has been performed on different datasets, the outcomes are in the same range, except for the class matching task since we restricted ourselves to a subset of the DBpedia classes.

By proposing the T2K Match method and the T2D gold standard, we laid the foundations for a systematical evaluation of the web table to knowledge base matching task. The error analysis indicated that only considering the table content not always sufficient to decide for the correct correspondence. Based on our findings in combination with the results of related approaches, we conclude that web tables often do not form self-contained units. Without considering additional features like the context of the table, e.g., the surrounding text, it is difficult, and sometimes impossible, to identify the meaning behind the data [Embley et al., 2006, Hurst, 1999].

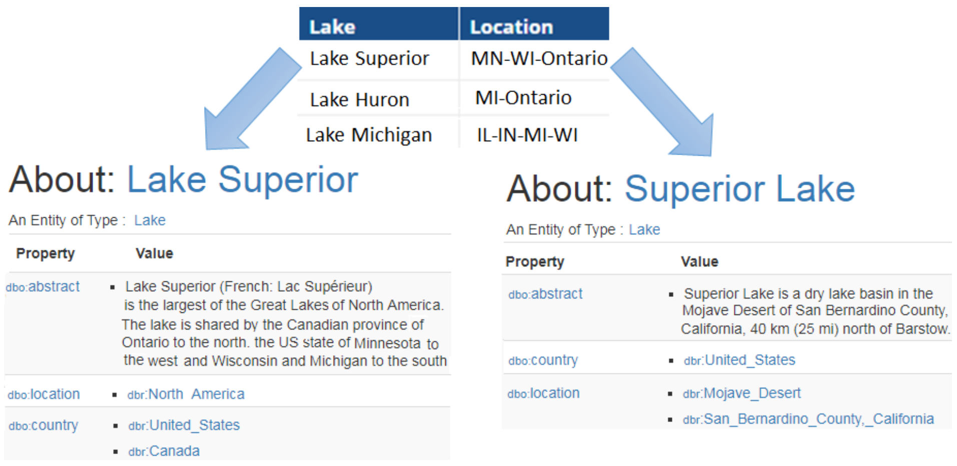
Chapter 7

Feature Utility Analysis

In the previous chapter, we introduced T2K Match, a method that performs the three matching tasks, instance, property, and class matching, in an integrated fashion. As shown on the T2D gold standard covering a wide range of challenges, the algorithm is capable of generating correspondences within a restricted scenario: only a subset of the DBpedia classes and web tables overlapping with DBpedia have been used. Even with the simpler input, we demonstrated that finding the correct correspondences is not straightforward. The main reason is the lack of evidence which leads to incorrect decisions. This lack cannot be overcome by the set of features used by T2K Match. Since tables do not always form self-contained units, features found in the table gather insufficient evidence to decide for the correct correspondence. For the instance matching task this happens if a value-based similarity cannot be computed due to non-overlapping attributes. In consequence, a matcher has to rely on the comparison of the labels which can be ambiguous.

In this chapter, we present a utility analysis of features that have been considered in state-of-the-art web table matching systems. The set of features includes features extracted from the context or gathered by external resources. By extending the T2K Match algorithm, it is possible to integrate all features into a single system and evaluate it on the T2D gold standard. Further, to be able to handle the increased amount of computed similarities in a proper way, we employ matrix predictors that adapt the similarity aggregation individually for each table. We use the full set of DBpedia classes and all web tables, including non-overlapping tables, of the T2D gold standard as input for the extended T2K Match method. Thus, we determine how the matching results vary within the more complex scenario. Moreover, we analyze the utility of every feature for the matching tasks. The analysis also includes an investigation to answer which features are most beneficial for which table characteristics. Finally, we show to which extent the additional features can overcome the lack of evidence as well as other difficulties we identified during the T2K Match error analysis in the previous chapter (Section 6.5.2). The following example illustrates how additional features can improve the matching performance.

Example 7.1 Figure 7.1 depicts an example in which the lack of evidence leads to an incorrect correspondence. For the lake with the label *Lake Superior* in the web table, two instance candidates in DBpedia have been detected: *Lake Superior* and *Superior Lake*. The label similarity to both candidates is the same. In addition, a value-based similarity cannot be determined since the value “MN-WI-Ontario” is not similar to any value of the candidates. Hence, a correspondence to one of the instances is created randomly. However, the string “Ontario” can be found in the abstract of the *Lake Superior* instance. In addition, based on popularity statistics, the *Lake Superior* candidate is much more common. By additionally considering the abstracts of the instances and their popularity, more evidence can be gathered which supports the decision that the candidate *Lake Superior* is the better choice.



Lake	Location
Lake Superior	MN-WI-Ontario
Lake Huron	MI-Ontario
Lake Michigan	IL-IN-MI-WI

About: Lake Superior

An Entity of Type : Lake

Property	Value
dbo:abstract	• Lake Superior (French: Lac Supérieur) is the largest of the Great Lakes of North America. The lake is shared by the Canadian province of Ontario to the north. the US state of Minnesota to the west and Wisconsin and Michigan to the south
dbo:location	• dbr:North America
dbo:country	• dbr:United_States • dbr:Canada

About: Superior Lake

An Entity of Type : Lake

Property	Value
dbo:abstract	• Superior Lake is a dry lake basin in the Mojave Desert of San Bernardino County, California, 40 km (25 mi) north of Barstow.
dbo:country	• dbr:United_States
dbo:location	• dbr:Mojave_Desert • dbr:San_Bernardino_County,_California

Figure 7.1: Example web table about lakes with two instance candidates for the entity with the label “Lake Superior”.

Subsequently, we start with the feature review in Section 7.1 and explain in Section 7.2 how T2K Match is adapted. Section 7.3 discusses how existing systems exploit different features. The utility of each feature is analyzed in Section 7.4 and the results of the best feature combination for each task are reported. In Section 7.5, the findings are summarized and open issues are discussed.

Parts of the feature overview and the results have already been published in [Ritze and Bizer, 2017]. This works was carried out by myself alone.

7.1 Feature Review

Within each matching process, features serve as input for matchers which apply a similarity measure to estimate the feature similarity. In this section, we introduce which features have been used for matching web tables by state-of-the-art approaches. To enable a better overview, we define a categorization scheme which enables a classification of the features according to different characteristics. Each feature category is described in more detail, including a summary of the methods that make use of these features. The matchers using the presented features as input are described in a later section of this chapter (Section 7.2).

7.1.1 Feature Categorization

To structure the types of web table features, we introduce the categorization scheme depicted in Figure 7.2. In general, a feature can either be found within the table itself (Table T) or outside the table (Context C). As context features, we consider everything that is not directly contained in the table. Context features can either be page attributes (CPA), like the page title, or free text like the words surrounding the table (CFT). We further divide table features into single features (TS), e.g., a label of an entity, and multiple features (TM), e.g., the set of all attribute labels occurring in a table. Single features refer to a value in a single cell while multiple features combine values spanning more than one cell. Other features originate from external resources (E).

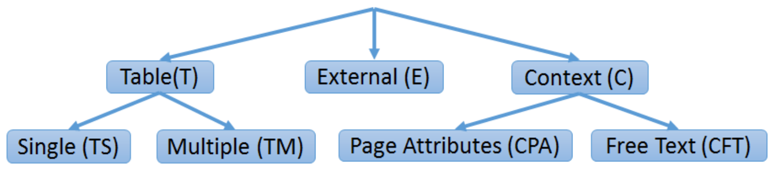


Figure 7.2: Web table feature categorization scheme.

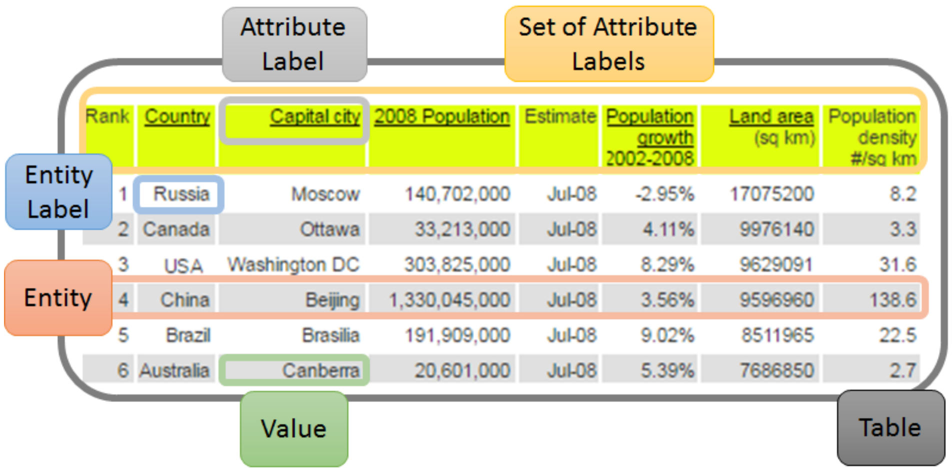
7.1.2 Table Features

Table 7.1 gives an overview of the table features used by web table matching systems. Single table features are the entity label, the attribute label, as well as the values in the cells. Multiple features are the entity with all information included in the row, the set of attribute labels, and the table as text. All multiple features are represented as bag-of-words.

Table 7.1: List of table features.

Feature	Description	Category
Entity label	The label of an entity	TS
Attribute label	The header of an attribute	TS
Value	The value that can be found in a cell	TS
Entity	The values of one row	TM
Set of attribute labels	The set of all attribute labels	TM
Table	The table content without any structure	TM

Example 7.2 In Figure 7.3, the tables features for an example table about countries are illustrated. An example of an entity label is *Russia* while *Capital city* presents an attribute label and *Canberra* is the value indicating the capital of Australia. The set of attribute labels contains for example the labels *Rank* and *Population density #sq km*. The entity given as bag-of-words for the country *China* looks like following: {4, China, Beijing, 133004500, Jul-08, 3.56%, 9596969, 138.6}. Finally, the table as bag-of-words covers all cells in the table, including the entity and attribute labels.



Rank	Country	Capital city	2008 Population	Estimate	Population growth 2002-2008	Land area (sq km)	Population density #/sq km
1	Russia	Moscow	140,702,000	Jul-08	-2.95%	17075200	8.2
2	Canada	Ottawa	33,213,000	Jul-08	4.11%	9976140	3.3
3	USA	Washington DC	303,825,000	Jul-08	8.29%	9629091	31.6
4	China	Beijing	1,330,045,000	Jul-08	3.56%	9596960	138.6
5	Brazil	Brasilia	191,909,000	Jul-08	9.02%	8511965	22.5
6	Australia	Canberra	20,601,000	Jul-08	5.39%	7686850	2.7

Figure 7.3: Table features of a web table about countries.

Since single table feature lay the foundation for the matching tasks, all state-of-the-art systems take them into account. However, most systems are limited to single table features in combination with features derived by other tasks [Zwicklbauer et al., 2013, Syed et al., 2010, Limaye et al., 2010, Muñoz et al., 2014, Venetis et al., 2011, Ling et al., 2013, Hassanzadeh et al., 2015]. Contrary, multiple table features are rarely considered. Approaches by [Hignette et al., 2007] and [Wang et al., 2012] include the set of attributes labels as features that are used to gather hints about the topic of a table. All systems using the Wikitology to gather candidates [Mulwad et al., 2010b, Mulwad et al., 2013, Syed et al., 2010] and TableMiner [Zhang, 2014b] match web tables to knowledge bases using the attribute label and the entity feature. Only InfoGather [Yakout et al., 2012] exploits all multiple table features but for matching web tables among each other.

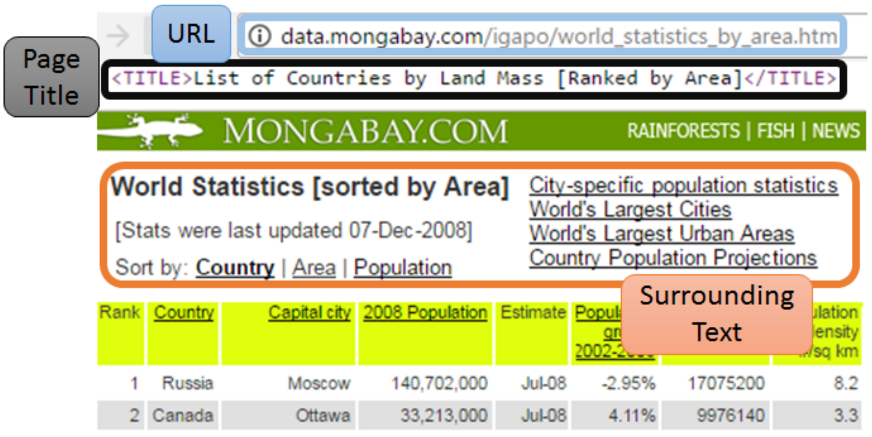
7.1.3 Context Features

The context of a table is specified as everything that is related to, but not included in, the table. Context features that are used in state-of-the-art approaches are listed in Table 7.2. Both, the page title and URL are page attributes that present general web page information. The only free text feature is the surrounding words, including all words before and after the table. In general, context features are not necessarily related to a particular table.

Table 7.2: List of context features.

Feature	Description	Category
URL	The URL of the web page	CPA
Page title	The title of the web page	CPA
Surrounding words	The 200 words before and after the table	CFT

Example 7.3 In Figure 7.4, the context features of the web table about countries are illustrated. The page title starting with "List of Countries" indicates that a table on this web page is about countries. In contrast, the URL does not directly tell something about the content. However, other URLs clearly help to understand the tables, e.g., the URL `http://airportcodes.me/us-airport-codes` expects a table found on this page to describe airports. The words surrounding the table, in this case only above the table, contain terms like "area" or "population" which give hints that the table is about populated places.



data.mongabay.com/igapo/world_statistics_by_area.htm

`<TITLE>List of Countries by Land Mass [Ranked by Area]</TITLE>`

World Statistics [sorted by Area]
[Stats were last updated 07-Dec-2008]
Sort by: Country | Area | Population

City-specific population statistics
World's Largest Cities
World's Largest Urban Areas
Country Population Projections

Rank	Country	Capital city	2008 Population	Estimate	Popul gr 2002-2		lation ensity /sq km
1	Russia	Moscow	140,702,000	Jul-08	-2.95%	17075200	8.2
2	Canada	Ottawa	33,213,000	Jul-08	4.11%	9976140	3.3

Figure 7.4: Context features of a web table about countries.

Two other imaginable context features are the title of the table and semantic markups like Microdata annotations. [Hignette et al., 2007] compute a similarity between the title of the table and property labels of a domain-specific ontology. The TableMiner system [Zhang, 2016] uses semantic markups found on the web pages. However, our preliminary investigations indicate that a table title is very rarely specified, only for about 3% of the tables. Similarly, semantic markups have only been detected on particular websites like *IMDB.com*, none of the websites from which the tables in T2D have been extracted.

As stated by [Lehmberg and Bizer, 2016], context features are crucial for high quality matching. Without considering information in the context of the table, it is sometimes even impossible to identify the meaning behind the data [Hurst, 1999]. Two different strategies have been proposed to exploit the context, either to gather more information about single attributes or about the table itself. [Braunschweig et al., 2015a] take the surrounding words to extract attribute-specific context, the CONTEXT operator of the Octopus system [Cafarella et al., 2009] uses context features to find hidden attributes which are not explicitly described in the table. To find information about the table, [Pimplikar and Sarawagi, 2012] match keywords to the surrounding text of the table. Both, the InfoGather [Yakout et al., 2012] and the TableMiner system [Zhang, 2014b] use the page attributes and the text to match web tables, either among each other or to a knowledge base. Thus, TableMiner is the only system tackling the web table to knowledge base matching task that goes beyond the information found within the table.

7.1.4 External Features

Additionally, to overcome terminological heterogeneities, a common strategy is to employ external resources, e.g., the general lexical databases WordNet [Fellbaum, 1998]. For matching web tables, systems use co-occurrences databases that leverage web text corpora [Venetis et al., 2011], natural language patterns to find relations between entities [Sekhavat et al., 2014], or exploit the anchor text of hyperlinks to find alternative surface forms of entity names [Bryl et al., 2015]. Altogether, we investigate three external resources: WordNet, a surface form catalog, and the popularity score of the instances computed based on Wikipedia inlinks.

7.1.5 Knowledge Base Features

In addition to the web table features, Table 7.3 depicts the according knowledge base (DBpedia) features. Analogously, knowledge base features can either refer to a single triple, e.g., a triple representing the information about an instance label, or to a set of triples like the set of all abstracts of instances belonging to a class. Features like the label or value are simply counterparts of table features. The abstract of an instance, provided by the *dbo:abstract* property, is the first paragraph of the according Wikipedia article. Contrary to other information provided by a knowledge base, the abstracts are unstructured texts.

7.2 Matching Components

In the following, we present the matching process that is used to determine the utility of the features. T2K Match algorithm is adapted to deal with the increased amount of features and the non-overlapping tables included in T2D. The component that has changed most is the aggregation of the similarities for which matrix prediction is employed. Further, the matchers for each task are discussed in detail.

Table 7.3: List of DBpedia features.

Feature	Description
Instance label	The name of the instance mentioned in the rdfs:label
Property label	The name of the property mentioned in the rdfs:label
Class label	The name of the class mentioned in the rdfs:label
Value	The literal/object that is found in the triple object position
Instance abstract	The abstract describing an instance
Instance classes	The classes (+superclasses) to which an instance belongs
Set of class instances	The set of instances belonging to a class
Set of class abstracts	The set of abstracts of instances belonging to a class

7.2.1 Adaptions of the Methodology

A quick recap, each matching process consists of the four steps: preprocessing, matcher execution, aggregation, and classification. The methodology introduced in the previous chapter covers straightforward methods for these components. Except for the preprocessing, all steps are improved to be able to handle the increased amount of features and non-overlapping tables.

- **Matcher Execution** First, the candidate selection is performed to find candidates for all tasks as before. Afterwards, a larger set of matchers is applied for each task will be introduced in Sections 7.2.3, 7.2.4, and 7.2.5.
- **Aggregation** With more features, more similarity matrices need to be aggregated. In contrast to fixed weights, an adapted aggregation strategy using matrix prediction is applied as it will be described in Section 7.2.2.
- **Classification** Distinguishing between overlapping and non-overlapping tables has not been necessary. To detect non-overlapping tables which lead to incorrect correspondences, an adapted classification with a prefiltering and a thresholding is applied.

Prefiltering As shown by [Cafarella et al., 2008a], only a small amount of web tables is relational. In addition, only about 3% of the relational tables actually overlap with DBpedia (see Chapter 5). Hence, we exclude all tables for which an overlap with the knowledge base is not foreseeable. Therefore, a set of heuristics is used. The heuristics follow two paradigms: overlapping tables correspond at least to a few instances and most of these instances belong to the same class since we assume a table to describe one topic. More precisely, Listing 7.1 depicts the four requirements that need to be fulfilled for a table in order to not get filtered. If less than 3 or less than 15% of the rows do not have an instance candidate at all, we assume the table to not have enough overlap with the knowledge base. Further, if the amount of classes to which the instances belong to is too large, the table either covers diverse topics or the candidates only seem similar but actually do not fit to the rows. In both cases, we do not want to generate correspondences between the table and the knowledge base because we cannot rely on the similarities. The same holds if more than 50% of the instances do not belong to the chosen class. Conditions for properties are not included since tables do not need to share properties with the knowledge base. All parameters have been determined empirically.

Listing 7.1: Pseudocode for the prefiltering of a table t as part of the classification.

```
inst <- best instance candidates of t
class <- best class candidates of t
classes <- all class candidates of t
if(|inst| < 3 or |inst|/|rows| < 0.15) {return true}
if(1 - |classes|/|inst| < 0.3) { return true}
if(|inst belong to class|/|inst| < 0.5) {return true}
return false
```

Thresholding As soon as all non-overlapping tables are filtered out, a threshold is applied on the set of possible correspondences to only return the ones that are likely to hold. If the final similarity score between two elements is too low, the system is not evident enough. In contrast to the threshold-based classifier in the algorithm, a threshold for each matching task is used instead of a threshold for all correspondences independent of the task. We determine a threshold for each matching task by a learning decision tree, trained in a 10-fold cross-validation style.

With the adaptions to the methodology, the method is able to deal with the additional features as well as with non-overlapping tables. Especially the efficient filtering of non-overlapping tables is important for matching web-scale corpora.

7.2.2 Matrix Prediction

With a rising amount of features, the number of similarity matrices generated by the matchers grows. The T2K Match algorithm used one of the most common aggregation strategies which assign weights to each matcher to indicate how much influence their matrices should have [Doan et al., 2012]. Most approaches in the field of web table matching use such a weighted aggregation to combine similarity matrices. While some of them empirically determine the weights, e.g., TableMiner [Zhang, 2014b], others employ machine learning [Yakout et al., 2012, Limaye et al., 2010]. They all have in common that the same weights are used for all tables. Due to the diversity of tables, one single set of weights is not always the best solution. Even tables with a same topic do not necessarily share the same characteristics. For example, for a table with descriptive attribute labels, the best strategy is to rely on the labels which is not the case for a table covering the same topic but uses very general attribute labels. To overcome this issue, we use a quality-driven combination strategy which adapts itself for each individual table. Such strategies have been shown as promising in the field of ontology matching [Cruz et al., 2009]. The approach tries to measure the reliability of matchers by applying so called matrix predictors on the generated similarity matrices [Sagi and Gal, 2013]. Predictors foretell the success of a matcher in identifying correct correspondences by analyzing the matchers pairwise similarity scores. The predicted reliability is then used as weight for each matrix. Since the prediction is individually performed on every individual matrix, the reliability of a matcher can differ for each table. Hence, we are able to use weights which are tailored to each table.

We propose and later evaluate three different matrix predictors: the average predictor (P_{avg}), the standard deviation predictor (P_{stdev}), and the Herfindahl predictor predictor (P_{herf}). While the first two predictors that have been proposed by [Sagi and Gal, 2013], the last one bases on the Herfindahl Index [Rhoades, 1993] and estimates the diversity of a matrix.

Average Predictor Based on the assumption that a high element in the similarity matrix leads to a correct correspondence, a matrix with many high elements is preferred over a matrix with lower elements. We compute the average of a matrix M with matrix elements e as following:

$$P_{avg}(M) = \frac{\sum_{i,j|e_{i,j}>0} e_{i,j}}{\sum_{i,j|e_{i,j}>0} 1} \tag{7.1}$$

Standard Deviation Predictor In addition to the average, the standard deviation indicates whether the elements in a matrix are close to the average. With a low standard deviation, we assume the matcher to always return roughly the same similarity independent of the input. Thus, it is difficult to distinguish whether a correspondence is correct. Formally, the standard deviation is computed as follows:

$$P_{stdev}(M) = \sqrt{\frac{\sum_{i,j|e_{i,j}>0}(e_{i,j} - \mu)^2}{N}} \tag{7.2}$$

μ is the average of all matrix elements and N is the number of non-zero elements.

Herfindahl Index Predictor The Herfindahl Index (HHI) is an economic concept which measures the size of firms in relation to the industry and serves as an indicator of the amount of competition among them. A high HHI indicates that one firm has a monopoly while a low HHI points to a lot of competition. We use this concept to determine the diversity of each matrix row and in turn of the matrix itself. Our matrix predictor based on the HHI is similar to the *Match Competitor Deviation Predictor*, which has been recently proposed by [Gal et al., 2016], that compares the elements of each matrix row with its average. We compute the normalized HHI for each matrix row which ranges between $\frac{1}{n}$ and 1.0 where n is the dimension of the matrix row.

$$\begin{bmatrix}1.0 & 0.0 & 0.0 & 0.0\end{bmatrix}$$

Figure 7.5: Matrix row with the highest possible HHI (1.0)

$$\begin{bmatrix}0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}$$

Figure 7.6: Matrix row with the lowest possible HHI (0.25)

Example 7.4 Figure 7.5 and Figure 7.6 show the highest and lowest possible case for a four-dimensional matrix row. At best, we find exactly one element larger than zero while all other elements are zero. Having this ideal case, we can perfectly see which pair fits. In contrast, a matrix row which has exactly the same element for each pair does not help at all to decide which correspondence should be created. As result, the matrix row in Figure 7.5 has a normalized HHI of 1.0 and the matrix row in Figure 7.6 of 0.25.

To get an estimation per matrix, we build the sum over all HHIs per matrix row and normalize it. Formally:

$$P_{herf}(M) = \frac{1}{V}\sum_{i}\frac{\sum_j {e_{i,j}}^2}{(\sum_j e_{i,j})^2} \tag{7.3}$$

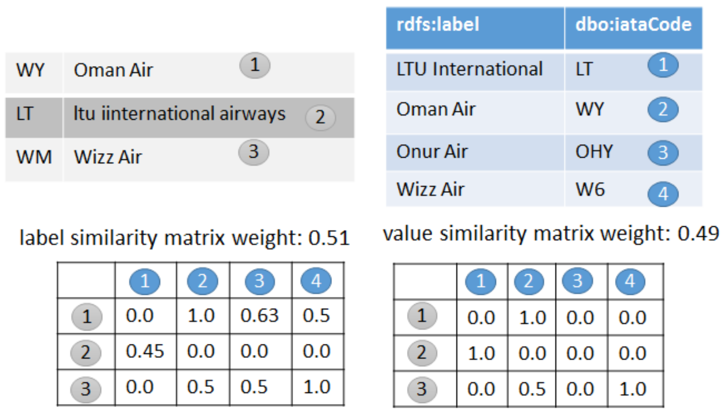
where V represents the number of matrix rows in the matrix.

All predictors predict a value between 0 and 1 where a high value indicates that the matrix is better according to the prediction criteria. A higher value can also be interpreted as a higher reliability such that a greater emphasis should be put on the according matrix during the aggregation. In Section 7.4.2, we will analyze which predictor is best suited for which matching task and to which extent the predictors correlate with the correct correspondences.

The following example motivates why individually adapted weights are useful.

Example 7.5 Figure 7.7 and Figure 7.8 depict excerpts of tables about airlines and videogames, together with parts of the corresponding DBpedia class.[1] For both tables, similarity matrices based on a label and value comparison between the entities and the instances have been computed. The weights of the matrices are determined on the full tables using the matrix predictor P_{herf}. While the weights for the table about airlines are almost the same $(0.51$ and $0.49)$, the weights for the table about videogames vary $(0.66$ and $0.33)$. Both, the name of the airline and the code, are almost unique and point to the correct instances. Hence, it makes sense to evenly rely on the label and the value, unlike it has been done using fixed weights, see Figure 6.4. Contrary, the attributes of the videogames table are not very discriminative for two reasons: the values can be different, e.g., "Namco Bandai" vs. "Bandal Namco Entertainment" and the publisher as well as the release date of other videogames can be the same. Thus, emphasizing the results of the label comparison is useful to determine the correct instance.

[1]Both tables originate from the T2D gold standard: 5873256_0_7795190905731964989 (airlines), 22864497_0_8632623712684511496 (videogames).



WY	Oman Air	1
LT	ltu iinternational airways	2
WM	Wizz Air	3

rdfs:label	dbo:iataCode	
LTU International	LT	1
Oman Air	WY	2
Onur Air	OHY	3
Wizz Air	W6	4

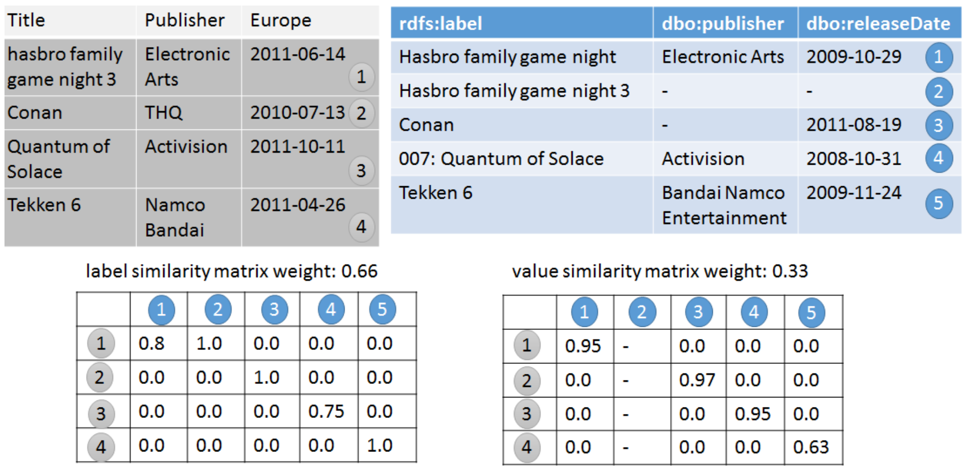
label similarity matrix weight: 0.51

	1	2	3	4
1	0.0	1.0	0.63	0.5
2	0.45	0.0	0.0	0.0
3	0.0	0.5	0.5	1.0

value similarity matrix weight: 0.49

	1	2	3	4
1	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0
3	0.0	0.5	0.0	1.0

Figure 7.7: Example table about airlines with similar label and value weights.



Title	Publisher	Europe	
hasbro family game night 3	Electronic Arts	2011-06-14	1
Conan	THQ	2010-07-13	2
Quantum of Solace	Activision	2011-10-11	3
Tekken 6	Namco Bandai	2011-04-26	4

rdfs:label	dbo:publisher	dbo:releaseDate	
Hasbro family game night	Electronic Arts	2009-10-29	1
Hasbro family game night 3	-	-	2
Conan	-	2011-08-19	3
007: Quantum of Solace	Activision	2008-10-31	4
Tekken 6	Bandai Namco Entertainment	2009-11-24	5

label similarity matrix weight: 0.66

	1	2	3	4	5
1	0.8	1.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	0.75	0.0
4	0.0	0.0	0.0	0.0	1.0

value similarity matrix weight: 0.33

	1	2	3	4	5
1	0.95	-	0.0	0.0	0.0
2	0.0	-	0.97	0.0	0.0
3	0.0	-	0.0	0.95	0.0
4	0.0	-	0.0	0.0	0.63

Figure 7.8: Example table about videogames with varying label and value weights.

7.2.3 Instance Matchers

We include five matchers for the instance matching task.

Entity Label Matcher The matcher compares the entity label with the instance label using a hybrid Jaccard with Levenshtein as inner measure. Only the top 20 instances with respect to the similarities are considered further for each entity.

Value-based Matcher We use the value-based matcher of T2K which uses data type specific similarity measures. The value similarities are weighted with the attribute similarities and are aggregated per entity. If we know that an attribute corresponds to a property, the similarities of the according values get a higher weight.

Surface Form Matcher Web tables often use synonymous names ("surface forms") which is difficult to spot for pure string-based similarity measures. The problem refers to the terminological heterogeneity which has been identified as a challenge for T2K. To understand alternative names, we use a surface form catalog that has been created from anchor-texts of intra-Wikipedia links, Wikipedia article titles, and disambiguation pages [Bryl et al., 2015]. Within the catalog, a TF-IDF score is assigned to each surface form. For each string we find in the table, either an entity label or a cell value, we build a set of terms consisting of the string itself together with according surface forms. We only consider the three surface forms with the highest TF-IDF scores if the difference of the scores between the two best surface forms is smaller than 80%, otherwise we only add the surface form with the highest score. The matcher itself compares each term found in the set of terms with the counterparts in the knowledge base by using the entity label and value-based matcher, respectively. As similarity score, the maximal similarity per set is taken.

Popularity-based Matcher The popularity-based matcher takes into account how popular an instance in the knowledge base is. For example, an instance with the label *Paris* can either refer to the capital of France or to the city in Texas. Both instances are equal regarding their labels but most of the times, the city in France will be meant. If additional evidence is missing, which is a serious issue as shown in the previous chapter, taking the most popular instance is the best strategy. To compute the popularity of an instance, we count the number of links in Wikipedia that point at the Wikipedia page describing the instance [Daiber et al., 2013]. Altogether, the matcher takes an entity and an instance as input and returns a similarity score that represents the popularity of the instance.

Abstract Matcher Comparing the entity label and the values can be insufficient if the labels differ too much or if available information about an instance is not covered by the values, e.g., the capital of a country is not mentioned in the knowledge base but it is stated in the abstract of the instance. This becomes especially relevant for the use case of filling missing values in the knowledge base. Therefore, the abstract matcher compares all values of a row with the abstracts of the instances, both represented as bag-of-words. For each entity given as bag-of-words, we create a TF-IDF vector and compare it to the TF-IDF vectors constructed from the abstracts where at least one term is required to overlap. As similarity measure we use a combination of the denormalized cosine similarity (dot product) and Jaccard to prefer vectors that contain several different terms in contrast to vectors that cover only one term a number of times:

$$sim(A,B) = A \bullet B + 1 - \left(\frac{1}{\|A \cap B\|}\right) \tag{7.4}$$

where A and B are TF-IDF vectors.

7.2.4 Property Matchers

We utilize the following four matchers for the property matching task.

Attribute Label Matcher Although the attribute label is not always available and not necessarily descriptive, it can give hints about the content of the attribute. For example, the label *capital* in a table about countries indicates that a property named *capital* is a better candidate than *largestCity* even if the similarities of the values are very close. Analogously to the similarity measures used for entity labels, we apply a hybrid Jaccard with Levenshtein as inner measure to compare the attribute and property label.

WordNet Matcher To solve alternative names for attribute labels, we consult the lexical database WordNet. WordNet is frequently applied in various research areas, e.g., in the field of ontology matching. Besides synonyms, we take hypernyms and hyponyms (also inherited, maximal five, only coming from the first synset) into account. As an example, for the attribute label *country* the terms *state*, *nation*, *land*, and *commonwealth* can be found in WordNet. For each attribute label, we generate a set containing the label itself together with additional terms. The similarity score is computed by applying the attribute label matcher on each term of the set and returning the maximal similarity score that has been determined among the set.

Dictionary Matcher While WordNet is a general source of information, we ancillary create a dictionary for attribute labels based on the results of matching the WDC WTC 2012 to DBpedia as it has been done for the profiling (Chapter 5). The correspondences are grouped by the property they refer to. For each group, the according labels of the attributes are extracted. Thus, we are able to generate a dictionary containing the property label together with the attribute labels that, based on the matching, seem to be synonymous. At this point, the dictionary includes a lot of noise, e.g., the term *name* is a synonym for almost every property. A filtering based on the number of occurrences or on the number of websites is not useful, since the rare cases are most promising. Hence, we develop a filter which excludes all attribute labels that are assigned to more than 20 different properties because they do not provide any benefit. The comparison is the same as for the WordNet matcher.

Duplicate-based Attribute Matcher The duplicate-based attribute matcher integrated in the algorithm is the counterpart of the value-based matcher: the computed value similarities are weighted with the according instance similarities and are aggregated over the attribute. Thus, if two values are similar and the associated entity-instance pair is similar, it has a positive influence on the similarity of the attribute-property pair.

7.2.5 Class Matchers

We use the matchers listed below to assign classes to tables.

Page Attribute Matcher One strategy to overcome the lack of evidence that results in incorrectly assigned classes is to consider details of the web page as the page title and URL from which the table has been extracted. We preprocess both page attributes, page title and URL, by applying stop word removal and stemming. The similarity of a page attribute to a class is the number of characters of the class label normalized by the number of characters in the page attribute.

Text Matcher Ideally, the set of abstracts belonging to instances of a class contains not only the instance labels and associated property labels but also significant clue words. We use the text matcher for the features set of attribute labels, table and surrounding words. All features are represented as bag-of-words. After removing stop words, we build TF-IDF vectors indicating the characteristic terms of the table and the classes. We apply the same similarity measure which is used by the abstract matcher.

Majority-based Matcher The majority-based matcher refers to the identification of class candidates as described in the algorithm. Based on initial instance candidates detected by comparing the entity and instance labels, it is counted how often such a candidate belongs to a particular class. As similarities, the relative amounts of candidate instances associated with a class are returned.

Frequency-based Matcher Ideally, we want to find correspondences to specific classes over general classes which is not captured by the majority-based class matcher. This issue has not been posed a challenge to the algorithm since it has been evaluated on a DBpedia subset covering less classes. Similar to [Mulwad et al., 2013], we define the specificity of a class c as following:

$$spec(c) = 1 - \frac{|c|}{\max_{d \in C} |d|} \tag{7.5}$$

where C is the set of all classes in the knowledge base.
The specificity for each class corresponds to the similarity score that will be returned by the frequency-based matcher.

Agreement Matcher The agreement matcher exploits the amount of class matchers operating on features covering different aspects. Although the matchers might not agree on the best class to choose, a class which is found by all the matchers is usually a good candidate. We propose the agreement matcher which takes the results of all other class matchers and counts how often they agree per class. In this case, all classes having a similarity score greater than zero are counted.

7.3 Related Work

In this section, we revise the state-of-the-art web table matching systems with respect to the features they use and how they exploit them. All systems have been introduced in the related work section of the previous chapter. We distinguish between approaches that only consider features extracted from the table and methods that include also information found outside the table. A comparison of the matching results reported by the systems is provided in the evaluation (Section 7.4).

7.3.1 Approaches Using Table Features

In this section, we discuss methods that rely on features found in the table itself. Some systems solely rely on table features without including any further background knowledge. A recently published approach uses word embeddings on the entity and instance labels. Thus, the similarities are computed based on the cosine distance between the entity and instance vector representation [Efthymiou et al., 2017]. Other approaches proposed by [Limaye et al., 2010] and [Muñoz et al., 2013] focus on gathering statistics about properties from the knowledge base, e.g., to determine whether a property is used in a functional way. [Limaye et al., 2010] find instance and property candidates by comparing the entity and attribute label as well as taking the values into account. Additionally, the method includes all alternative instance and property labels that can be found in the knowledge base, e.g., redirects. Based on the gathered candidates, a joint inference is applied to find the best assignments for all three matching tasks. Features that go beyond the ones that can be directly extracted from one cell (single table features) are not considered. The following approaches focus on the features found in the table but incorporate background knowledge like probabilistic databases or the Wikitology to collect further information.

Probabilistic databases as background knowledge Another group of methods use probabilistic databases with information about word co-occurrences that have been gathered from large text corpora using Hearst pattern. [Venetis et al., 2011] query the entity label in an isA database to get suitable classes for a column. By applying a maximum-likelihood model, the best classes are determined. Similarly, a relation database for properties is used to assign properties to pairs of named entity columns. If for two columns many label pairs are mentioned within the same relation, the relation is assigned. Thus, only the entity label as well as the values are considered as features. [Wang et al., 2012] utilize the isA database called Probase together with an additional database for properties. As additional feature, the set of attribute labels is taken into account to get an idea about the table's topic. Both approaches only consider a small set of features since they rely on statistics provided by the probabilistic databases. These statistics are not available in knowledge bases like DBpedia. Further, they focus on the class and property matching task without explicitly linking the entities.

Wikitology as Background Knowledge Among the approaches focusing on features found in the table, three of them [Mulwad et al., 2010b, Mulwad et al., 2013, Syed et al., 2010] additionally use the Wikitology as background knowledge. Wikitology [Syed, 2010] is a hybrid knowledge base consisting of structured and unstructured information extracted from Wikipedia, enriched with structured information from DBpedia, Freebase, WordNet, and Yago. Wikitology provides an API that allows for structured, unstructured, and combined queries. Thus, it provides a powerful source of knowledge.

Listing 7.2: Wikitology query for an entity as provided by [Mulwad et al., 2010b]

```
Input:   Entity Label, Entity, Attribute Label
Output:  Top k instances from the knowledge base
Query:   wikiTitle: entity label OR
         redirects: entity label OR
         firstSentence: entity label, attribute label OR
         types: attribute label OR
         categories: attribute label OR
         contents: entity label (boost 4), entity OR
         linkedConcepts: entity label (boost 4), entity OR
         propertyValues: entity
```

Listing 7.2 shows how Wikitology is queried to get instances candidates. As input, it takes the entity label, the entity (row content), and the attribute label. These features are mapped to various fields of the Wikitology index. Based on the mapping to the index, a query is generated which checks a set of alternative conditions and returns the top k instances. The query checks whether the entity label is contained in the title of a Wikipedia article (instance label), in a redirect or in the first sentence of the Wikipedia article. Similarly, the first sentence of the according Wikipedia article is also searched for the attribute label. Further, the attribute label is looked for in Wikipedia types or categories. The row data including the label (boosted with a weight of 4) is searched in the full Wikipedia article text (contents), within the set of linked articles (linked concept) as well as in the infobox (property values). In summary, the Wikitology enables the simple usage of single and multiple table features by providing an API that answers queries under consideration of structured and unstructured data. In knowledge bases, such an API is usually not available such that the functionality needs to be encoded in the matching algorithm.

Using the returned candidates, the approaches by [Mulwad et al., 2010b, Syed et al., 2010] compute similarities to entities by applying popularity (page rank, Wikitology index score, Wikipedia article length) and string similarity measures (Levenshtein, Dice). Besides the internal Wikitology ranking, both methods consider the instance label and popularity as features. The method proposed by [Mulwad et al., 2013] uses the Wikitology candidates as basis for a joint inference.

7.3.2 Approaches Using Context Features

Beside features found in the table itself, information from the web page, also called context, can also be used. Context features are exploited for matching web tables among each other or to assign classes to tables. Further, the context is popular to improve table search by finding query terms in the context, extracting hidden attributes, or gathering additional attribute labels. Especially for entity tables where the entity label is not mentioned in the table itself, considering the context is crucial [Yin et al., 2011].

Context features for table search To improve the table search, [Pimplikar and Sarawagi, 2012] and subsequently [Sarawagi and Chakrabarti, 2014] match keywords of the query among others to the surrounding text of the table. The TF-IDF similarity between the keywords in the query and the context is one indicator whether the table is considered as relevant for the query. Contrary to other approaches, the context is specified to consist of segments that are selected based on their position in the DOM tree of the page document. Thus, only the segments close to the table are considered as useful.

The CONTEXT operator of the Octopus system [Cafarella et al., 2009] uses context features to find hidden attributes which are not explicitly described in the table but projected out since they hold for every entity. For example, the VLDB 2008 conference page contains a table with the accepted publications. Since the publication year is the same for each entity, there is no attribute stating the year. However, the information is available on the web page and especially easy to understand for humans. Two proposed methods add additional attributes to the table which have been derived from the context. One method is called *SignificantTerms*: it examines the web page and returns the best k terms according to TF-IDF scores that are not included in the table itself. Our approach does not explicitly generate new attributes but use the information in the context for the comparison.

[Braunschweig et al., 2015a] use the context to gather more information about single attributes to improve table and attribute search. Although [Pimplikar and Sarawagi, 2012] stated that the context is useful but does not provide attribute-specific information, the context can for example be exploited to resolve abbreviations in the attribute labels. Different levels of the context are taken into account: the table caption, the headings before the table, the surrounding text as well as the full text. By searching the attribute label in the context, either more detailed descriptions can be extracted or acronyms and abbreviations can be resolved. However, large amounts of attribute labels are non-informative. To find more informative attribute labels, the entities of name-entity columns are matched to instances in a knowledge base (YAGO). Based on the instances, a majority vote for the classes is performed and the name of the best suitable class is used as attribute label. Additionally, WordNet is utilized to gather synonyms of the attribute labels.

Context features for table to table matching InfoGather [Yakout et al., 2012] uses context features to match web tables among each other. Altogether, the surrounding text, the table, the URL, the set of attribute names as well as the set of values within one column are considered as features. All features are represented as bag-of-words. For two web tables, the according bag-of-words are compared using the cosine similarity based on TF-IDF scores. Since they are solving the table-to-table matching task, the context features can be directly compared. Contrary, we cannot compare the surrounding text of a table with the surrounding text of the knowledge base but have to artificially generate a document that can be compared, i.e., the set of abstracts of all instances belonging to one class.

Context features for table to knowledge base matching Tableminer [Zhang, 2014a] and its extension TableMiner+ [Zhang, 2016] is the only system that exploits the context of a table for matching web tables to knowledge bases. As table features, they use the attribute label, entity (called row content), and the column content. Context features, also referred to as out-table features, are the page title, the table caption, the surrounding paragraphs, and semantic markups. The algorithm is divided into a learning and an update phase.

During the learning phase, first instance candidates are collected by searching for instance labels that overlap with the entity labels. For these candidates, two similarity scores are computed basing on the context and the name. To compute the context similarity, for each candidate all triples are queried that have the candidate in the subject position. All objects found in these triples are normalized and put together in a bag-of-words. All features, in- as well as out-table features, are compared to the created bag-of-words using a weighted Dice measure. The name similarity is the term overlap between the instance and entity label. Initial class similarities are computed for each column by checking to which classes the best candidates belong to. Further, all context features except for the row content are compared to the bag-of-words consisting of the name and URI of the class. Afterwards, the instance and class matching are iterated until no new candidates for both tasks are detected. At this point, the backward-update phase starts. The algorithm decides for the class which in turn discards all instances that do not belong to this class. As last step, the properties are assigned letting the instances vote for the best property. Beside the similarity that is based on the instances, a context score is computed using the attribute header, surrounding paragraphs, and semantic markups for which the overlap with the property name and URI is checked.

In contrast to our approach, the popularity of an instance is not considered at all and alternative names or abbreviations are only taken into account if a property in the knowledge base explicitly refers to them, e.g., redirects. Further, the entity represented as bag-of-words is compared to all objects of triples with the according instance as subject. Hence, if a value is missing in the knowledge base, it will not occur in a triple and will thus not be available for the comparison. That is the reason why we use the abstracts which often contain additional or other information about the instances. Since the context usually covers general information about the topic of the table and not about particular entities or attributes, it is not clear to which extent the context helps for the instance and property matching task.

7.4 Evaluation

In this section, we analyze the utility of the various features that serve as input for the matching of web tables to knowledge bases. First, we specify the experimental setup in Section 7.4.1. Section 7.4.2 evaluates the applicability of the matrix prediction aggregation. The aggregation weights indicate which features are in general important and for which features the influence highly depends on the characteristics of a particular table. Subsequently, Sections 7.4.3 - 7.4.5 present the results for each matching task by applying different combinations of features. The analysis comprises a discussion about improvements and a comparison to other approaches.

7.4.1 Experimental Setup

T2K has been adapted with all components mentioned in Section 7.2. The code has been released in the T2K github repository.[2] As input for the matching process, the English DBpedia (version 2014) and the T2D gold standard (version 2) including both, overlapping and non-overlapping tables, are used. In summary, 779 tables are included from which 234 overlap with DBpedia and the others are either non-relational tables or do not share entities that are represented in DBpedia.

7.4.2 Results of the Matrix Prediction

With an aggregation strategy based on matrix prediction we try to enable a the combination of similarity matrices computed by different matchers that is tailored to each individual table. The goal of this section is to analyze whether the introduced matrix predictors are applicable and useful and if so how the weights differ for tables with diverse characteristics. Following [Sagi and Gal, 2013], we measure the quality of a matrix predictor using the Pearson product-moment correlation coefficient [Pearson, 1895]. With a correlation analysis, we can verify to which extent the weights are consistent with the correct correspondences. We perform a correlation analysis with each of the matrix predictors P_{avg}, P_{stdev}, and P_{herf} to the evaluation measures precision and recall. If a predictor has a high correlation to precision and recall and we use the predictor for determining the weights, we assume the resulting correspondences to show a comparable performance.

Table 7.4 shows the Pearson correlation coefficients between the instance and property similarity matrices and precision and recall. All correlations are signif-

[2]`https://github.com/T2KFramework/T2K/tree/EDBT`

Table 7.4: Pearson correlation coefficient between different matrix predictors and precision and recall.

Matcher	Precision			Recall		
	P_{stdev}	P_{avg}	P_{herf}	P_{stdev}	P_{avg}	P_{herf}
	Instance Similarity Matrices					
Entity Label	-0.17	-0.16	**0.23**	0.09	0.05	**0.23**
Value-based	0.36	0.12	**0.38**	0.5	0.31	**0.53**
Surface Form	-0.29	-0.29	**0.24**	-0.09	-0.13	**0.24**
Popularity-based	0.14	0.11	**0.26**	-0.04	**-0.04**	-0.24
Abstract	0.05	0.13	**0.21**	0.18	**0.29**	0.15
mean	0.02	-0.02	**0.33**	0.16	0.12	**0.23**
	Property Similarity Matrices					
Attribute Label	**0.45**	0.43	0.22	0.42	**0.49**	0.21
Duplicate-based	0.05	**0.09**	-0.07	0.09	**0.11**	0.04
WordNet	**0.43**	0.32	0.12	0.34	**0.37**	0.18
Dictionary	0.36	**0.36**	0.13	0.27	**0.45**	0.15
mean	**0.33**	0.3	0.1	0.28	**0.34**	0.15

icant according to a two-sample paired t-test with significance level $\alpha = 0.001$. The analysis of the class similarity matrix predictors is not shown since the correlations are not significant. This results from the fact that only 237 tables in T2D overlap with DBpedia and in turn can be assigned to a DBpedia class. Thus, the amount of data on which the correlation can be computed is too low to determine a statistically significant effect. However, in practice the predictor P_{herf} shows the most promising results. The same holds for instance similarity matrices where P_{herf} has the highest correlation to both precision and recall. In contrast, for property similarity matrices, P_{avg} correlates most. A reason is the comparably low amount of possible property candidates for each attribute. Within each matching task, the choice of the best performing predictor is mostly consistent. One exception is the correlation of P_{herf} to the recall of the popularity-based matrix since the most popular instances do not necessarily need to be the correct candidates. Based on the results, we use the prediction computed by P_{herf} as weights for the instance and class similarity matrices and P_{avg} for the property similarity matrices.

By having a look at the weights, we can estimate the influence of a particular matcher on the overall results. Figure 7.9 depicts the variations of weights for the similarity matrices from different matchers. The median of the weights indicates the overall importance of the features across all tables for a certain matching task.

- For the instance matching, the popularity of an instance seems to play a crucial role, followed by the label.
- For the property matching, the values build the foundation.

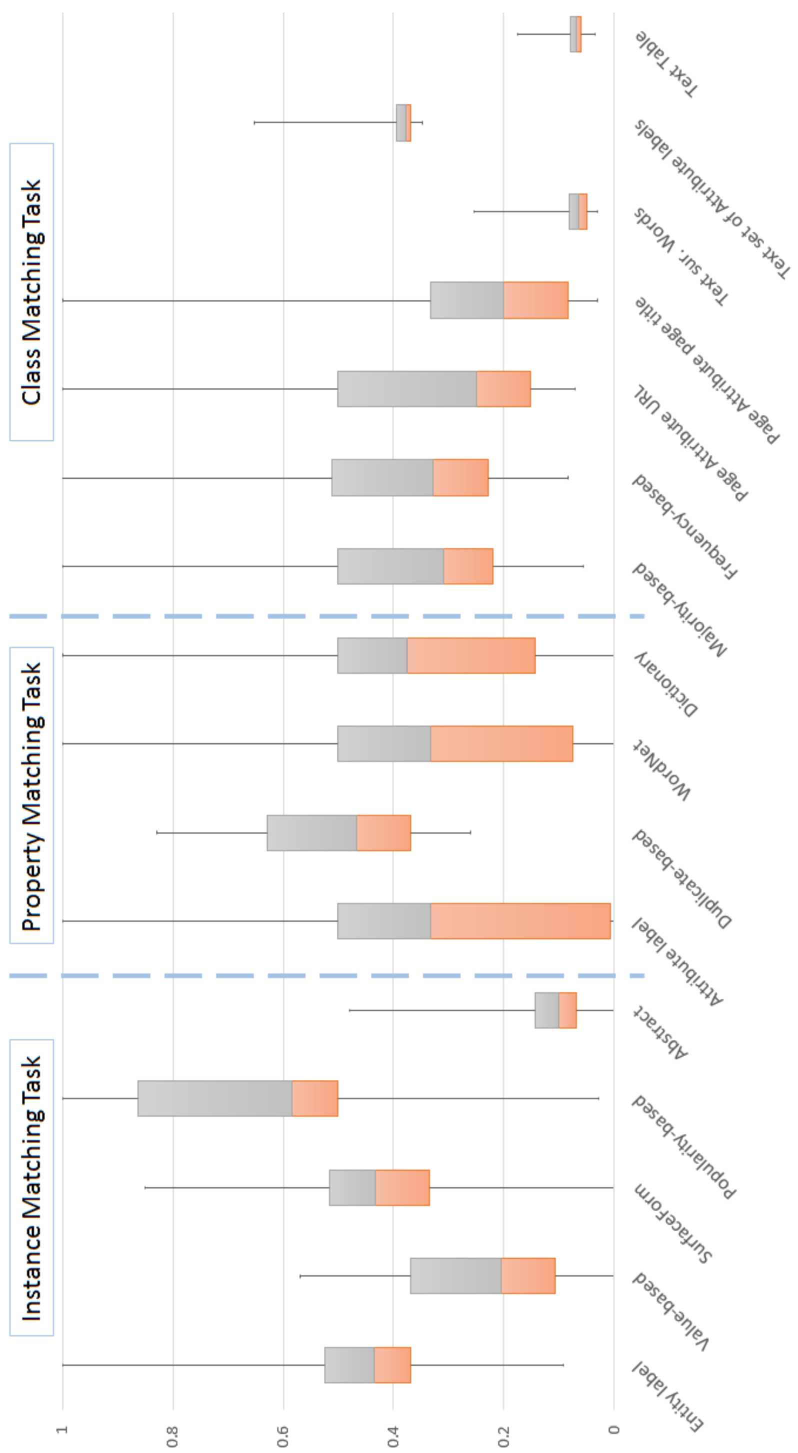


Figure 7.9: Boxplot of the matrix aggregation weights divided into quartiles for each matcher.

- For the class matching, the size of the class (frequency-based) and the amount of instance candidates (majority-based) are most important.
- Adding external resources only lead to slight changes of the weights.

In contrast to the median, the partly large variations of the weights illustrate that the actual utility of a feature depends on the individual matrix and in turn on the table. This supports our assumption that taking the same aggregation weights for all tables independent of their characteristics is not the best strategy. While the weight variations are very large for all matchers operating on attribute labels (Attribute Label-, Wordnet- and Dictionary matcher), it is the opposite for the matchers handling bag-of-words. A large variation indicates that a feature is suitable for some but not for all tables. This confirms the finding that web tables often either do not contain attribute labels or ambiguous or non-informative labels (Section 6.1.1). For matchers taking bag-of-words as input, the reliability is estimated similarly low for all tables. Comparing bag-of-words always results in large amounts of candidates since the applied similarity measures are vague.

To estimate whether a machine learning approach as for example used by [Limaye et al., 2010, Yakout et al., 2012] is also suitable, we experimented with decision trees to learn the weights. Each decision tree gets all similarities from the matchers as input and its task is to learn an aggregation model. For the instance matching task, the learned decision tree solely considers the similarities generated by the Entity Label matcher, all other similarities are not taken into account. This results from the fact that the similarity based on the label comparison is the only one producing reliable results across all tables. By applying matrix prediction, we exactly overcome this issue without the need for supervision.

7.4.3 Results of the Instance Matching Task

First of all, we present the evaluation of the instance matching task. The goal is understand which features are most important for the instance matching task and which performance can be achieved with the best combination of these features.

Table 7.5 presents the results for different combinations of matchers.[3] By only considering the entity label, a moderate result with a precision of 0.72 is achieved. Additionally taking the values into account increases the recall by 0.09 and the precision by 0.08. As expected based on the weight analysis, including the values helps to improve the performance but only using the entity label already leads to a decent amount of correct correspondences. By adding surface forms, the recall can again be raised by 0.02 which shows that alternative names for entities are used.

The popularity-based matcher slightly increases the precision and recall when additionally considered besides the label and values. Whenever the similarities for

[3]Note that we do not display all possible combinations but the most notable ones.

Table 7.5: Results of the instance matching task using different combinations of matchers.

Matcher	Precision	Recall	F-measure
Entity Label	0.72	0.65	0.68
Entity Label + Value-based	0.80	0.74	0.77
Surface Form + Value-based	0.80	0.76	0.78
Entity Label + Value-based + Popularity-based	0.81	0.76	0.79
Entity Label + Value-based + Abstract	0.93	0.68	0.79
All	0.92	0.71	0.80

candidate instances are close and no further evidence can be gathered, selecting the more common instance is in most cases the better decision. However, this assumption does especially not hold for web tables containing long-tail entities.

Including the abstract matcher, which in contrast to all other instance matchers relies on a multiple table feature, results in a precision increase of 0.13 while 0.08 recall is lost. The loss of the recall might be unexpected at first glance since a matcher comparing bag-of-words tends to generate large amounts of similarities that can be noisy. The reason for the precision increase lies in the classification. Since many potential correspondences exist, high thresholds need to be selected in order to prevent a breakdown in F-measure. Thus, comparing all values of a row with instance abstracts helps to find correct correspondences but has to be treated cautiously to not ruin the overall performance. If we use the combination of all instance matchers, the highest F-measure of 0.80 can be achieved. This illustrates that the instance candidates found by matchers exploiting different features do not necessarily overlap. Thus, the matchers can benefit from each other by complementing each other and compensating their weaknesses. Without the matrix prediction, exactly this variety of features cannot be exploited. As we explained at the end of the previous section, a decision tree only uses the entity label as feature since the similarities computed by other matchers vary to much among the tables. Based on this decision, the maximal F-measure that can be achieved is 0.68 which is the results of solely using the entity label matcher.

Comparison with other instance matching results The T2K Match algorithm including only the entity label and values as features results in an F-measure of 0.82 within the restricted scenario (Section 6.5.2). On the one hand, the F-mesaure results achieved with the same features are only 0.05 less even though non-overlapping tables are included and all DBpedia classes are used. On the other hand, including all features, only a slight loss in F-measure has to be accepted. Concerning instance matching methods introduced in related works, the only systems which compare web tables with knowledge bases without including background knowledge or considering the context are provided by [Limaye et al., 2010], reporting an F-measure of 0.84 and by [Efthymiou et al., 2017] with an F-measure between 0.82 and 0.85, depending on the gold standard. Thus, with the features found in the table a good performance can be achieved.

By exploiting the Wikitology as background knowledge, the approaches automatically include all features that we consider for the instance matching task. [Mulwad et al., 2010b] report an F-measure of 0.66 and [Syed et al., 2010] of 0.77. However, the evaluation has only been performed on 15 resp. 5 tables such that it is difficult to derive general conclusions. A subsequent approach by [Mulwad et al., 2013] results in an F-measure of 0.76. The main reason for the comparably low performance as stated by the authors is the lack of relevant data in the knowledge base: correct instances do not belong to the class or do not use the according property which is a requirement for the method to generate correspondences.

TableMiner+, the only system that considers the context, achieves an F-measure of 0.84. Hence, on similar tables, the approaches by Limaye et al. and Zhang result in almost the same performance although Limaye et al. consider the smallest set of features. Further, for the methods using the Wikitology including a broad variety of features, the performance is also below. Thus, the results from state-of-the-art systems exactly underline that the utility of features cannot be derived properly if the features are not included into a single system. However, the results are in a similar range to the best results we report such that an F-measure of 0.84 seems to be the upper bound for the instance matching task.

Summarizing, the entity labels seems to be the important feature but without including values, a performance above 0.7 is not reachable, mainly due to the missing information and terminological heterogeneities. With additional features like the popularity of the instances, the results can be slightly increased but an F-measure above 0.8 is not reachable which is in line with reported results from other systems. Thus, even with a large set of features, especially the ambiguity of the entity label and the difficulty to decide for the correct instance cannot be fully overcome. One possible reason is that even with a surface form catalog or by considering the abstracts, the terminology or abbreviations used in web tables are not represented.

7.4.4 Results of the Property Matching Task

The results of the property matching experiments using different combinations of matchers are illustrated in Table 7.6. In contrast to the instance matching task, we get a rather low recall of less than 0.5 if we only take the attribute label into account. Based on the weight analysis, we are already aware that the attribute label is not necessarily a useful feature for all tables.

Including values, covered in the duplicate-based matcher, increases the recall by 0.35 but slightly decreases the precision by 0.10. While values provide the possibility to compensate missing or non-informative attribute labels, they can also adds incorrect correspondences if they fit accidentally. This especially holds for attributes of data type numeric and date. For example, in a table describing films a

Table 7.6: Results of the property matching task using different combinations of matchers.

Matcher	Precision	Recall	F-measure
Attribute Label	0.85	0.49	0.63
Attribute Label + Duplicate-based	0.75	0.84	0.79
WordNet + Duplicate-based	0.71	0.83	0.77
Dictionary + Duplicate-based	0.76	0.86	0.81
All	0.70	0.84	0.77

numeric attribute contains values that refer to a ranking. If these values are close to a property like runtime, an incorrect correspondence will be generated since the values seem to fit. Nevertheless, the values present a valuable feature especially to achieve a decent level of recall. Taking the general lexical database WordNet into account does neither improve precision nor recall. This confirms the findings by [Braunschweig et al., 2015a], that a general dictionary is not very useful for the property matching task. In contrast, using the dictionary created from web tables increases the recall as well as the precision. Hence, with specific background knowledge that is tailored to web tables, it is possible to enhance the performance as it has also been shown by [Yakout et al., 2012]. However, the creation of the dictionary requires a lot of enhanced filtering. Without proper filtering, the dictionary would only add large amounts of noise. The result of using all matchers together is slightly lower than the best result due to the correspondences created by the WordNet matcher which the other matchers are not able to prevent.

Comparison with other property matching results With the algorithm that only relies on the duplicate-based method, an F-measure of 0.7 is determined. Similar to the instance matching task, even with non-overlapping tables and the full knowledge base, comparable results are achieved. Our results for the property matching task are difficult to compare to other methods as many of the existing systems only match attributes to object properties. Nevertheless, the results of related approaches can give insights about the usefulness of features. By only relying on the table features, [Limaye et al., 2010] report an F-measure of 0.58, [Muñoz et al., 2014] of 0.79. With Wikitology as background knowledge, the F-measure varies a lot between 0.25 [Mulwad et al., 2010a] and 0.84 [Mulwad et al., 2013], depending on the set of tables that is used for the evaluation as well as whether an inference model is applied. Using a probabilistic database as performed by [Venetis et al., 2011] does not seem to be beneficial, achieving an F-measure of 0.55. The TableMiner+ system states an F-measure of 0.76 by exploiting the context [Zhang, 2016]. For the property matching task, including the context does not lead to the highest results but using background knowledge together with an inference model seems to be a suitable method. As for the instance matching task, our results are in the same range as reported by others and close to the best performing system.

Table 7.7: Results of the class matching task using different combinations of matchers.

Matcher	Precision	Recall	F-measure
Majority-based	0.47	0.51	0.49
Majority-based + Frequency-based	0.87	0.90	0.89
Page attribute	0.97	0.37	0.53
Text	0.75	0.34	0.46
Page attribute + Text + Majority-based + Frequency-based	0.9	0.86	0.88
All	0.93	0.91	0.92

Overall, the values are most important for the property matching task together with the attribute label as long as it is informative. While a general lexical database does not improve the results, with an additional dictionary tailored to web tables, the best results can be achieved. Similarly to the instance matching task, an F-measure higher than 0.84 is not reported by any system.

7.4.5 Results of the Class Matching Task

Table 7.7 reports the results of our class matching experiments. When only considering the majority of the instance candidates to compute the class correspondences, the precision is 0.47 and the recall 0.51. Thus, for only about half of the tables the correct class is assigned. The main reason is the preferential treatment of superclasses over specific classes which are lower down in the class hierarchy. All instances that can be found in a specific class also count for its superclasses and there might be further instances candidates belonging to these superclasses. Together with the consideration of the frequency which exactly tackles the mentioned issue, an F-measure of 0.89 can be reached.

In order to see how far we get when solely considering matchers that rely on context features, we evaluate the page attribute matcher and the text matcher independently from the others. Since the differences in the performances are marginal, we do not present the results for the individual features. Whenever the page attribute matcher determines a high similarity, the according correspondence is very likely to be correct. However, since the URL and page title are compared with the label of the class, it can happen that no candidate is found at all. Regarding the recall, similar holds for the text matcher but the generated correspondences are not necessarily correct. This is not surprising because we already discovered that matchers using features represented as bag-of-words have a weak ability to differentiate between correct and incorrect candidates due to a lot of noise.

When we combine all previous matchers, an F-measure of 0.88 is obtained which is still lower than the outcome of the majority- together with the frequency-based matcher. If we make use of the number of available class matchers which is transposed by the agreement matcher, we reach an F-measure of 0.92. Thus, taking advantage of features covering the whole spectrum of available information and deciding for the class most of them agree on, is the best strategy for the class matching task. Due to the fact that the class matching task has a strong influence on the other two matching tasks, their performance can be substantially reduced with an incorrect class. For example, when only using the text matcher, the recall of the instance matching task drops to 0.52 and the property recall to 0.36.

Comparison with other class matching results Contrary to the other two tasks, when applying the same strategy (majority-based matcher) as used within the baseline algorithm, a similar performance cannot be sustained. On the one hand, more classes are available such that superclasses are preferred and on the other hand, non-overlapping tables for which a class correspondences is created worsen the results. These are also the reasons why even with all features a slightly lower (0.02) performance compared to the baseline algorithm can be achieved. As for the other tasks, it is difficult to see which features are most useful from existing works. The lowest result, an F-measure of 0.43, is reported for the method of [Limaye et al., 2010] which only considers information within the table. When taking also the specificity of the classes into account [Mulwad et al., 2013], the F-measure of 0.57 is neither higher nor lower than results of [Venetis et al., 2011] or [Zwicklbauer et al., 2013]. Same holds for considering the context with a result of 0.63 [Zhang, 2014b]. Nevertheless, considering the context seems to slightly improve the results. The only system reporting performances in the same range as ours (0.9) is the approach of [Syed et al., 2010] but is has only been evaluated on 5 tables. As we discussed during the profiling (Chapter 5) the choice of the knowledge base can have a strong influence especially on the class matching result which is also an explanation why other matching systems report way lower performances.

Altogether, for the class matching task it is important to consider to which classes the candidate instances belong to as well as to take the frequency of the class into account to not assign too generic classes. By having a look at the context, some remaining cases can be addressed which are difficult to solve otherwise. In contrast to both, the other matching tasks as well as other systems, the results for the class matching task are the highest with an F-measure above 0.9.

7.5 Summary

In this chapter, we analyzed the utility of features that are used to match web tables. First, we gave on overview of the features and proposed a classification scheme. Features can either be found in the table itself or can be extracted from the context. Further, external resources like catalogs including alternative names can be taken into account. For each feature, we introduced task-specific matchers that are integrated in T2K Match. By adding more features and in turn generating more similarity matrices, the aggregation needs to be able to combine the matrices in a useful way. Therefore, we presented the concept of matrix prediction that individually adapts the aggregation weights for each table. Using matrix predictors, we allow different web tables to favor the features that are most suitable.

We used different combinations of features for each matching task. In contrast to the evaluation of the T2K Match algorithm in Section 6.5, we included the non-overlapping tables of the T2D gold standard and all DBpedia classes which presents a more realistic scenario. As one contribution of this thesis, we proposed a utility analysis of the features which considers two dimensions. First, the weights assigned during the aggregation indicate which features are important for which tasks and how the influence of a features varies for individual web tables.

- The entity label and the popularity has the strongest influence on the instance matching results, the values contribute less but depends on the table.
- Usually the value gets the highest weight for the attribute matching while the influence of the attribute label varies greatly if the label is non-informative.
- The influence of the features for the class matching is similarly distributed among the majority-based, frequency-based and page attribute matchers.
- Features that are represented as bag-of-words like the surrounding words of a table show very small variation and often only contribute little.

Second, the evaluation of the correspondences shows which combinations of features achieve the best results for each task. In summary, taking as many features as possible into account is promising for all three tasks. Features found within tables generally lead to a higher correctness than context features. Nevertheless, taking context features into account can improve the results but particular caution is necessary since context features may also add a lot of noise. External resources proved to be useful as long as their content is closely related to the content of the web tables, i.e., the general lexical database WordNet does not improve the results for the property matching task while a more specific dictionary does.

Altogether, taking a variety of features into account leads to F-measure scores above 0.8 for the instance and property matching task and above 0.9 for the class matching task. These results are in the same range or even above as the results reported for T2K Match although a more realistic scenario without restrictions regarding the input is considered (Section 6.5). Further, also the results reported by other approaches are in the same range. Still, insufficient evidence due to the small size of the tables, the lack of discriminative content, or different kinds of heterogeneity cannot be fully overcome. Thus, the next chapter will introduce the T2K Match++ method which exploits indirect matches to web tables about the same topic and uses knowledge derived from the knowledge base.

Chapter 8

The T2K Match++ Method

In the previous chapters, we introduced the T2K match algorithm to match web tables to knowledge bases and included additional features like the context of the table. The analysis of the results shows that the method suffers from insufficient evidence due to the small size of the tables, the lack of discriminative content, or different kinds of heterogeneity that cannot be resolved by external resources. One promising approach to overcome these issues is holistic matching that has been used to match databases and ontologies [Do and Rahm, 2002, Hartung et al., 2013]. Holistic matching follows the observation that various sources contain the same data but represent it differently [Rahm, 2016]. By jointly matching all sources, additional evidence can be gathered as shown in the subsequent example.

Example 8.1 The entity *French Republic* is not similar to the instance *France* in DBpedia although both refer to the same country. If we know that *French Republic* and *Republic of France* describe the same real-world object and in turn *Republic of France* and the DBpedia instance *France*, we can use the indirect match to infer that *French Republic* and *France* also represents the same country.

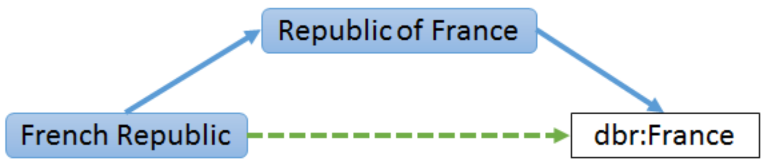


Figure 8.1: Example of an indirect match.

Applying a holistic approach to match web tables to knowledge bases gives rise to additional challenges. First, the tables need to be matched to each other for the purpose of finding out which tables represent the same data. A table to table matching potentially requires large amounts of comparisons. Second, the holistic matching can also produce spurious matches resulting in incorrect correspondences [Yakout et al., 2012]. Thus, we need a way to deal with spurious matches to not deteriorate the results. This especially holds for the property matching task. As we will show, the value comparison is not always sufficient for this task to get high precision results. One solution is to exploit domain knowledge that is gathered from the data itself. This includes the distribution of the values of a property or logical constraints which are given by the knowledge base. For example, assuming a proper unit detection, an attribute including continuous numeric values will not correspond to the property stating the population of a country since populations are given as discrete values. The following example shows how holistic matching improves the results for matching web tables to DBpedia.

Example 8.2 Figure 8.2 shows three web tables that have been matched to DBpedia using the T2K Match algorithm. Solid arrows represent correct correspondences, dashed arrows indicate errors. All tables describe countries and the country names are depicted in the first (T2,T3) and second (T1) column, respectively. For T2 and T3 all correct DBpedia countries are assigned. Contrary, for T1 only one correct instance correspondence to the country *Germany* has been found, one correspondence is missing (*France* for *french republic*) and one is incorrect (*U.S. state Georgia* instead of the country *Georgia*). The country *France* is not matched because the names are too dissimilar. Deciding for the U.S. state instead of the country is the better choice since in general this instance is referenced more often. Regarding the property matching task, two incorrect correspondences to the property *population* are generated. The reason is the incorrect assumption that the column values represent populations of countries in millions. While T2 and T3 are correctly matched to a class called *Country*, T1 is assigned to the general class *Place*. *Place* is chosen for T1 because all mapped instances are places and no more specific classes can be assigned since *Georgia* is a state and *Germany* a country.
With a holistic approach, all correct instances and in turn correct class correspondences can be determined. Since the entity with the name *French Republic* of T1 is similar to the entity *Republic of France* in T2, we can infer that the instance *France* should also be assigned to the entity in T1. Further, we determine that the majority of entities having *Georgia* as name points to the country instead of the U.S. state. The incorrect correspondences to the property *population* can be resolved by determining that populations follow a normal and not a uniform distribution.

In this chapter, we describe the T2K Match++ method that holistically matches web tables to a knowledge base. To complement the method, we utilize domain knowledge for the property matching task. We evaluate the T2K Match++ method on two gold standards: T2D and Limaye. Further, we show that the algorithm outperforms all state-of-the-art web table to knowledge base matching systems.

First, Section 8.1 presents the T2K Match++ method. Section 8.2 discusses other holistic approaches and methods exploiting domain knowledge. Afterwards, the results are presented and compared to the performances achieved by other systems (Section 8.3). The chapter is concluded by a summary (Section 8.4).

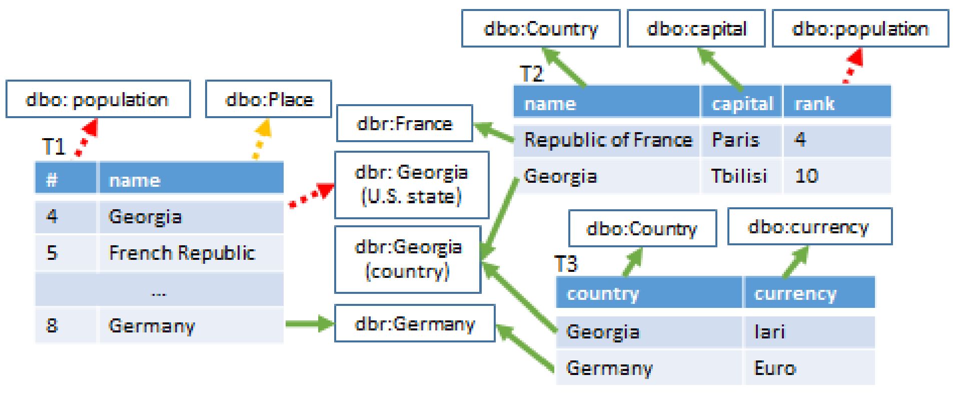


Figure 8.2: Correspondences of three web tables generated by T2K Match.

The description of the T2K Match++ algorithm and parts of the evaluation is included in the paper [Ritze and Bizer, 2018]. The paper has been submitted to the WSDM conference 2018. This works was carried out by myself alone.

8.1 Methodology

This section introduces the T2K Match++ method. At first, the overall workflow is depicted and each step of the matching process is described. Compared to T2K Match, an indirect matching component is included. In addition, the classification is improved. All other parts remain unaffected.

8.1.1 Workflow

Figure 8.3 gives an overview of the overall T2K Match++ workflow, performing the instance, property, and class matching task in an integrated fashion. It takes web tables and a knowledge base as input and returns correspondences between entities and instances, attributes and properties, and the web tables and knowledge base classes. The process consists of the following steps: **Preprocessing**, **Direct Matching**, **Prefiltering**, **Indirect Matching**, **Aggregation** and **Classification**.

- **Direct Matching & Prefiltering** The direct matching is the same as in T2K Match (Chapter 6). Further, the best combination of features per task is included as analyzed in Chapter 7. For the instance matching task, the surface form matcher, the value-based, and the popularity-based matcher are considered (Section 7.2.3). To find property correspondences, the dictionary and duplicate-based matchers are used (Section 7.2.4). Lastly, for the class matching task, all matchers introduced in Section 7.2.5 are taken into account. The prefiltering excludes web tables that do not overlap with DBpedia (Section 7.2.1). Filtering these tables is especially important for the

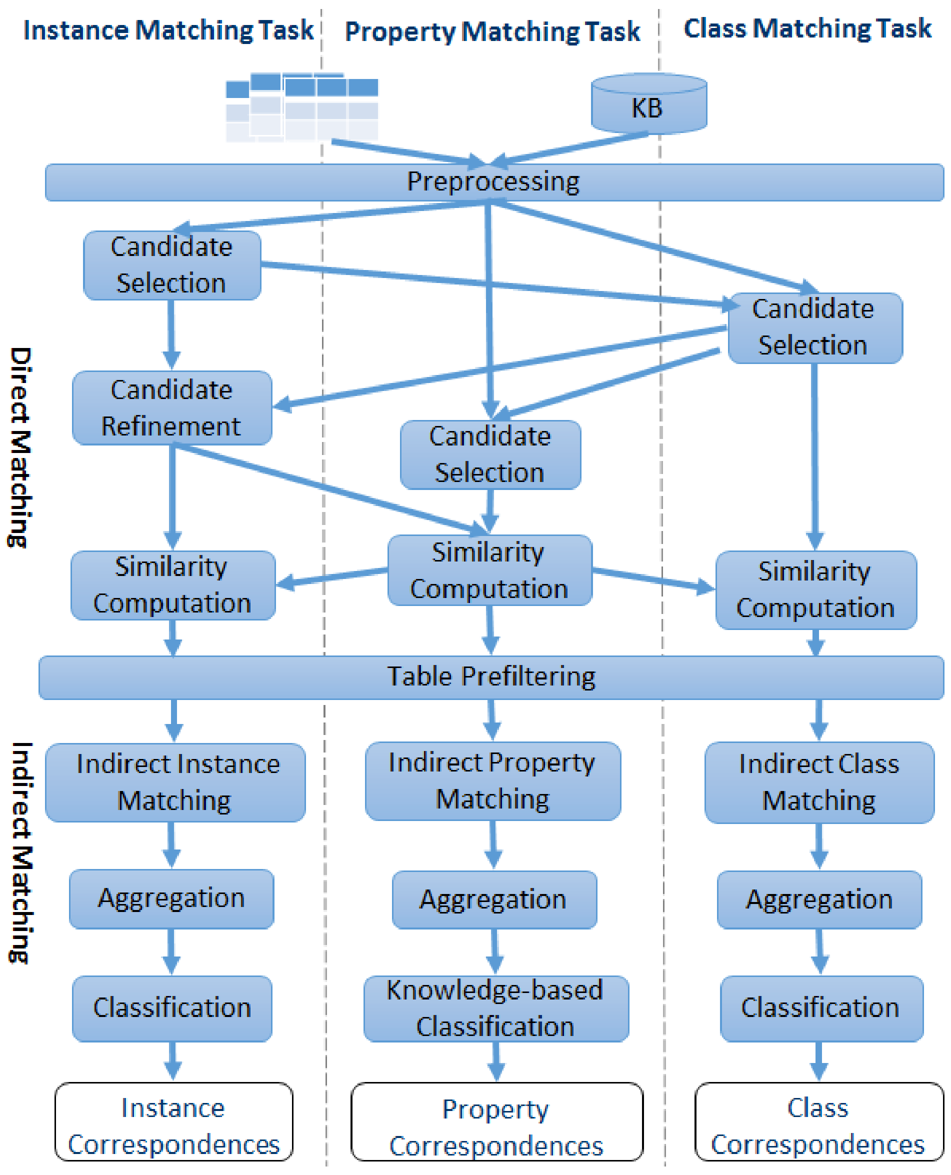


Figure 8.3: Holistic matching workflow of T2K Match++.

indirect matching. Otherwise, a large number of comparisons is required and the possibility to generate spurious matches increases.

- **Indirect Matching** The holistic component goes beyond the direct comparison between a web table and the knowledge base by exploiting that web tables about the same topics also have similarities to the knowledge base. These similarities can be used to gather more evidence or to detect similarities to knowledge base elements that have not been considered before. A detailed description is presented in Section 8.1.2.

- **Aggregation** All similarity scores coming from the direct and indirect matching are aggregated using the strategy based on matrix prediction as described in Section 7.2.2. The weights to combine the similarities base adapt themselves for each individual table. In turn, if the direct matching similarities seem to be useful in contrast to the indirect ones, these similarities will be weighted higher to avoid spurious matches.
- **Classification** During the classification, the generated correspondences are classified into matches and non-matches to only keep the ones which are likely to hold. To be robust to spurious matches that can easily occur within the indirect matching step, a proper classification strategy is needed. For the property matching task, this includes domain knowledge. Section 8.1.3 explains the classification in more detail.

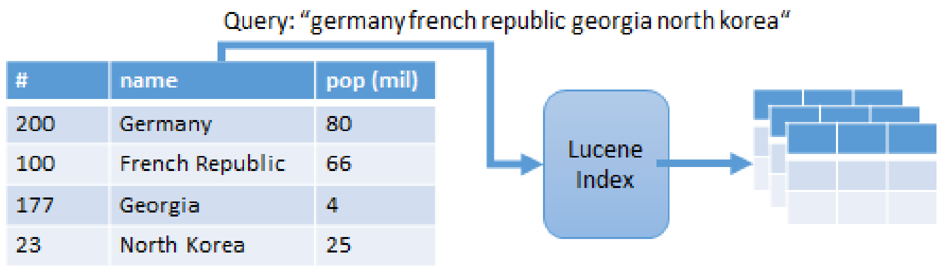
In summary, the T2K Match++ workflow includes components that have been described in the previous chapters. Additionally, an indirect matching step is added which requires a more sophisticated classification to not lose oneself in the amount of possibly inaccurate correspondences.

8.1.2 Indirect Matching

One main limitation of the direct matching is the lack of evidence that cannot be fully overcome even by including additional features. Within the indirect matching step, we exploit that multiple tables covering the same topic are potentially matched to the same knowledge base. The indirect matching consists of three subsequent steps: table search, table matching, and mapping composition.

Table Search During the table search, we identify all tables which seem to be similar to a given table t. To facilitate the search, we build a Lucene index containing the concatenated entity labels of every table. Note that the index does not include each individual entity label but only the concatenation per table to reduce the number of queries that are required and in turn to speed up the search. Figure 8.4 shows the query process. For a table t, we pose a query to the index that covers the entity labels processed the same way. In the given example, the query will be "germany french republic georgia north korea". The index returns all web tables that are relevant according to the internal Lucene ranking. The ranking sorts the tables mainly based on the cosine similarity of their TF-IDF vectors with the query vector. Among the results, we take the top k tables. For our experiments, we set $k = 1000$ which will be analyzed in Section 8.3.3.

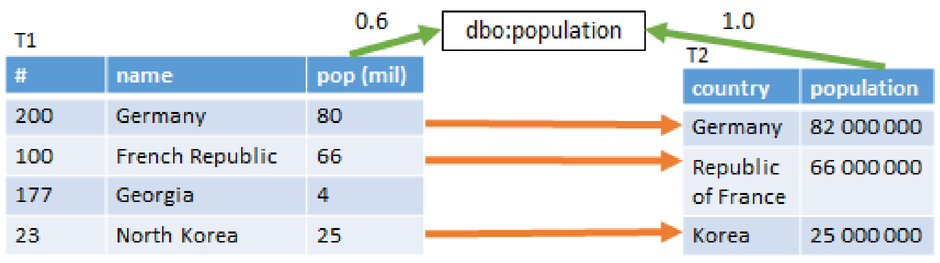
Table Matching After receiving relevant tables for t, we perform a table to table matching between t and all its relevant tables. Analogously to the instance, property and class matching task, we need to perform an entity, attribute as well as table matching. Similar to the matching of web tables to knowledge bases, we start with finding candidate entities by comparing the entity labels using a hybrid



#	name	pop (mil)
200	Germany	80
100	French Republic	66
177	Georgia	4
23	North Korea	25

Figure 8.4: Table search for related tables.

Jaccard measure. For each of the candidates, we compute a value-based similarity by applying the same set of similarity measures as the T2K algorithm. By matching all tables to the same central knowledge base, we face the specific situation to be aware of similarities and in turn correspondences between all tables and the knowledge base. Hence, we can use these correspondences to reduce the number of comparisons. This blocking strategies is called *correspondence-based blocking*. Thus, we only compare values of candidates as long as the attributes are similar to the same property in the knowledge base according to the direct matching. Figure 8.5 illustrates the matching of two tables T1 and T2.



T1

#	name	pop (mil)
200	Germany	80
100	French Republic	66
177	Georgia	4
23	North Korea	25

T2

country	population
Germany	82 000 000
Republic of France	66 000 000
Korea	25 000 000

Figure 8.5: Correspondence-based blocking for the table to table matching.

Since we know that the entity with the label *Georgia* does not have any candidate in T2, we do not need to compare its values at all. The same holds for the left-most attribute in T1 because it does not have similarities to any knowledge base property. In both tables, we find an attribute that refers to the property *population* in the knowledge base. Thus, with the correspondence-based blocking only the values of the attribute with the header *pop(mil)* of T1 are compared to values of the attribute *population* of T2. In combination with the candidate blocking, the number of comparisons is reduced to three instead of 24. Apart from any heterogeneities and under the assumption that all similarities to the knowledge base from the direct matching are correct, we only compare values which should cover the same information. Thus, if two values are not similar at all, it is a strong indication for non-corresponding entities.

The attribute matching task is performed analogously: labels are used to generate candidates and the similarities to instances in the knowledge base for blocking during the value comparison. For the table matching task, the similarity between two tables is computed by counting how often the entities point to instances from a certain class. By considering to which classes the instances belong to, we get a more fine-grained view than relying on the similarities to the classes that have been computed during the direct matching. For all three tasks, we do not include any additional features due to the increased complexity and the findings from the feature utility analysis in Chapter 7 that further features only slightly improve the results when matching web tables to knowledge bases. As last step of the table matching, we aggregate the similarity scores per task by applying the same matrix predictors as introduced in Section 7.2.2.

Mapping Composition By now we only computed similarities among the tables but not between tables and knowledge base elements. The similarity score for a table element a to the according knowledge base element b is determined by applying a weighted voting strategy:

$$sim(a, b) = \frac{\sum_{x \in Sim(b)} sim(a, x)}{|C|}$$

where C is the set of candidates of a and $Sim(b)$ the set of table elements that are similar to b. As example, for an element a in total three candidates are available, two of them point to the element e while the other one points to the element f. The resulting similarities are computed as follows: $sim(a, e) = \frac{2}{3}$ and $sim(a, f) = \frac{1}{3}$. A typical strategy for computing similarities when exploiting transitivity is to multiply similarity scores. This would require multiplying the similarity of the candidate $sim(a, x)$ with the similarity score to the element in the knowledge base $sim(b, x)$. We renounce to do that because it leads to a rapid degrading of the similarity value as shown by [Do and Rahm, 2002].

After executing the three steps, a set of similarities to knowledge base elements based on the indirect matches has been created for each table element. Note that, we do not include any additional levels of indirection, e.g., we are not looking at tables that are only indirectly similar to a table t. The reason is that already one level of indirection produces huge amounts of similarities that we have to deal with. Additionally, every level of indirection potentially contains a large amount of spurious matches that needs to be filtered.

8.1.3 Classification

The classification in T2K Match learns and applies a threshold on the final similarity score to specify whether a correspondence is likely to hold. Thus, the threshold-based classification purely relies on this score and does not take any further information into account. However, we are in possession of a large set of information that we can for example gather during the matching. To improve the classification, we use a strategy which goes beyond thresholding. For the instance task (I), property task (P), and class task (C), we build decision trees which are fed with features as described in Table 8.1. Most features are received twice, once from the direct and one from the indirect matching (IN). In the feature descriptions, we refer to other features using the number depicted in the first column.

Table 8.1: Features used for the classification.

#	Feature	Description	Task
1	(In) Direct Label Similarity	computed label similarity	I,P
2	(In) Direct Value Similarity	computed value similarity	I,P
3	Sum Similarities	$1 + (In)1 + 2 + (In)2$	I,P
4	(In) Direct Label Weight	matrix predicted label weight	I,P
5	(In) Direct Value Weight	matrix predicted value weight	I,P
6	(In) Direct Label Score	$1 \cdot 4$	I,P
7	(In) Direct Value Score	$2 \cdot 5$	I,P
8	(In) Direct Score	$6 + 7$	I,P,C
9	Final Score	$8 + (In)8$	I,P,C
10	Sum Label Scores	$6 + (In)6$	I,P
11	Sum Value Scores	$7 + (In)8$	I,P
12	(In) Direct Label Weight Deviation	$1 - 4$	I,P
13	(In) Direct Value Weight Deviation	$2 - 5$	I,P
14	(In) Direct Weight Deviation Sum	$12 + 13$	I,P
15	(In) Direct Label Mean Difference	$1 - \oslash\sum(1)$	I,P
16	(In) Direct Value Mean Difference	$2 - \oslash\sum(2)$	I,P
17	Sum (In) Direct Differences	$15 + 16$	I,P
18	Mapped Entity Ratio	$\frac{\#\text{entities}}{\#\text{entity correspondences}}$	I,P,C
19	Mapped Entities	#entity correspondences	I,P,C
20	Mapped Attribute Ratio	$\frac{\#\text{attributes}}{\#\text{attribute correspondences}}$	I,P,C
21	Mapped Attributes	#attribute correspondences	I,P,C
22	Number of indirectly found tables	#tables by table search	I,P,C

- The first set of features ($1 - 11$) covers all similarities, aggregation weights, scores (combination of similarity and weight), and sums of the scores. They present the similarity estimates provided by each matcher of the direct and indirect matching.
- Features $12 - 14$ capture if one table element shows another behavior than other elements of the same table. For example, a table describes countries with an attribute stating the capital. Most entities in this table will have a high similarity to the knowledge base instances since both, the label of the countries and the capital fit. If one entity has low label and value similarities, the mapped instance might not be correct.

- In the same way, the difference of one similarity to the similarities of all tables computed by a specific matcher provides the information if the similarity (or a combination of similarities) is above or below the average (15,16,17).
- Other features provide characteristics of the created mapping like the relative and absolute amount of entities and attributes for which a corresponding knowledge base element has been detected ($18 - 21$) and the amount of similar indirect tables (22).

Each decision tree model is trained in a 10-fold cross-validation style in order to prevent overfitting and create a generalized model. The learned decision trees are presented and analyzed in the evaluation (Section 8.3.3). As indicated in the introducing example in Figure 8.2, this classification is not sufficient for the property matching task due to its characteristics. Hence, a classification, called knowledge-based classification, with an additional set of features is applied.

Knowledge-based Classification The described classification relies on features gathered during the matching process without taking information into account which can be derived from the knowledge base itself. We refer to such information as domain knowledge. As example, all population values in the knowledge base are discrete numbers such that an attribute with continuous values cannot represent the population.[1] Considering such domain knowledge is especially useful for the property matching task. On the one hand, the domain knowledge can be straightforwardly derived for properties from the knowledge base since usually a sufficient amount of values exist. Thus, convincing characteristics can be derived which is not the case for particular instances for which only a few facts exist. On the other hand, the property matching task poses challenges that do not need to be addressed by the other tasks. One challenge is that only about half of the attributes overlap with properties from the knowledge base since web tables frequently cover information that is not represented in a knowledge base. For example, web tables about movies often include ranking attributes which are not contained in the knowledge base. In this case, relying on the similarities between the values results in incorrect correspondences. By additionally taking indirect matches into account the behavior can even be reinforced if incorrect correspondences confirm each other. To tackle this issue, we extract characteristics of the properties and use them for the classification. In the following, we list all features that are used during the knowledge-based classification. While some features are general, others are only useful for particular data types. Due to the feature division, we create an individual decision tree for each data type (Section 8.3.3). An evaluation of the knowledge-based classification will be provided in Section 8.3.

Fraction of Distinct Values (all data types) Properties can cover only distinct values, e.g., capitals of countries, or repetitive values from a fixed set, e.g., age

[1]Assuming an optimal unit detection such that values like “1.2 thousand” cannot occur.

restrictions of films. An attribute that covers many distinct values will most likely not correspond to a category property although the values might look similar. A related idea is to create recognizer which cover the characteristics of attributes [Doan et al., 2012]. For each attribute and property that are linked by a correspondence, we first determine their fraction of distinct values and then take the difference.

$$frac_{dist}(p) = \frac{|distinct(V)|}{|V|} \qquad diff_{dist}(p,a) = frac_{dist}(p) - frac_{dist}(a)$$

where V are the values of an attribute or property p and $distinct(V)$ is the set distinct values. If the fraction of distinct values is almost the same for the attribute and the property, $diff_{dist}(p,a)$ is close to 0.

Functionality (all data types) In knowledge bases, restrictions are expressed in the according ontology language. A typical restriction of the ontology language OWL is to define properties as functional (*owl:FunctionalProperty*) which indicates that there can only be exactly one value of the property for each instance, e.g., there can be only one birth date of a person. The feature we use indicates whether two attributes are mapped to the same functional property. Exploiting OWL restrictions has already been proposed by [Ehrig and Sure, 2004].

Cosine Similarity (string) For each attribute and property of data type string, we normalize all values and split them at white spaces or special characters. Further, we build TF-IDF vectors and compute their cosine similarity. If an attribute does not share any important terms with the property, a correspondence does not make sense although the values are similar. The resulting feature lies between -1 indicating the totally opposite and 1 meaning exactly the same.

Value Ranges (numeric, date) We check whether the median of the values of an attribute is between the minimum and maximum value of the property. Using percentiles instead of the minimum and maximum is not convincing because web tables often show a completely different accumulation, e.g., for a typical web table describing the highest mountains, the values of the height attribute will even be outside the 0.99 percentile of the height property of mountains. Since we only verify if the median is between the minimum and maximum, this feature is binary.

Fraction of Discrete Values (numeric) Numerical values can either be discrete or continuous. For example, the population of a country cannot be continuous since there cannot be half a person. Even if all values are similar, a continuous attribute will most likely not fit to a discrete attribute. More precisely, we take the difference of the fraction of discrete values for the attribute and property.

$$frac_{disc}(p) = \frac{|disc(V)|}{|V|} \qquad diff_{disc}(p,a) = frac_{disc}(p) - frac_{disc}(a)$$

where V are the values of an attribute resp. property p and the $disc(V)$ represents the set of discrete values in V. Whenever $diff_{d}isc(p,a)$ is close to 0, the attribute and the property share a similar behavior regarding the fraction of discrete and continuous values.

Kurtosis (numeric) The kurtosis of a distribution, the forth moment, measures the peakedness and in turn tells something about their shape. If the shapes differ essentially, the attribute and the property will most probably not contain the same information. As example, an attribute with ranks of films will not fit to the property indicating the runtime. In the first case, a union distribution is given while in the second case, a normal distribution can be assumed. More specifically, we compute the Pearson's kurtosis, also referred to as excess kurtosis, of both the attribute and the property and use their difference as feature. Including the characteristics of the distributions during the matching of numeric attributes into account is also described by [Pham et al., 2016].

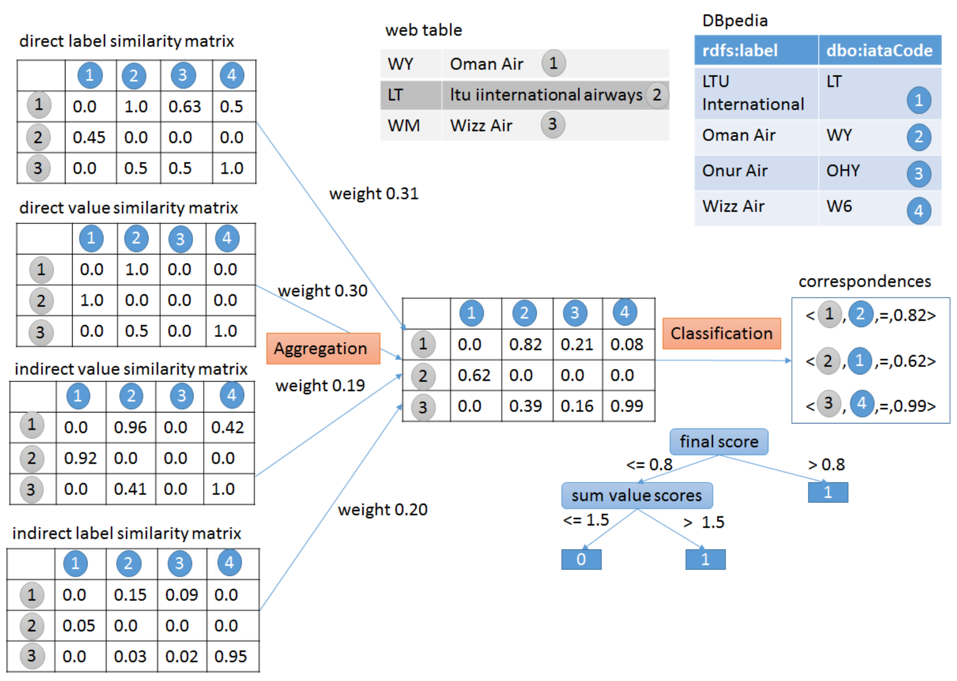
$$kurt(p) = \frac{1}{|V|}\sum_i (\frac{v_i - \overline{v}}{s})^4 - 3 \qquad diff_{kurt}(p,a) = kurt(p) - kurt(a)$$

where s is the standard deviation. While a uniform distribution has an excess kurtosis of -1.2, a normal distribution has an excess kurtosis of 0. Thus, if $diff_{kurt}(p,a)$ is close to 0, the underlying distributions are shaped similarly, the further away they are, the more diverse their peakedness is.

Example Workflow The following example depicts how the steps of the workflow work together to generate instance correspondences.

Example 8.3 Figure 8.6 presents the result of each workflow step for the instance matching task. The input is an excerpt of a table of the T2D gold standard.[2] Besides the entity label column, it covers one attribute with the airport code. The airport code corresponds to the property *dbo:iataCode* in DBpedia. Further, for all three entities in the table, a corresponding instance exist in DBpedia. During the direct and indirect matching, similarity matrices including the similarities of the label and value comparisons are generated. Matrix prediction is applied to compute the matrix weights. The weights for the direct matching matrices are about 0.1 higher than the weights for the indirect ones. This reason is that the indirect matrices cover more entries greater than 0 which leads to a lower reliability. The differences between the label and value weights are marginal since the label and the code are mostly unique. The computed weights are used during the aggregation to combine the similarities, resulting in one similarity matrix. This similarity matrix serves as input for the classification during which a learned decision tree is applied to decide whether a correspondence should hold. The simplified decision tree decides for a correspondence if the final score, presented in the aggregated similarity matrix, is above 0.8. If the final score is lower, the sum of the values scores from the direct and indirect matching are considered. With a sum greater than 1.5 the correspondence is not filtered out since the direct and indirect matching both agree on a high

[2]Table 5873256_0_7795190905731964989 in the T2D gold standard.



web table

WY	Oman Air	1
LT	ltu iinternational airways	2
WM	Wizz Air	3

DBpedia

rdfs:label	dbo:iataCode	
LTU International	LT	1
Oman Air	WY	2
Onur Air	OHY	3
Wizz Air	W6	4

direct label similarity matrix

	1	2	3	4
1	0.0	1.0	0.63	0.5
2	0.45	0.0	0.0	0.0
3	0.0	0.5	0.5	1.0

direct value similarity matrix

	1	2	3	4
1	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0
3	0.0	0.5	0.0	1.0

indirect value similarity matrix

	1	2	3	4
1	0.0	0.96	0.0	0.42
2	0.92	0.0	0.0	0.0
3	0.0	0.41	0.0	1.0

indirect label similarity matrix

	1	2	3	4
1	0.0	0.15	0.09	0.0
2	0.05	0.0	0.0	0.0
3	0.0	0.03	0.02	0.95

	1	2	3	4
1	0.0	0.82	0.21	0.08
2	0.62	0.0	0.0	0.0
3	0.0	0.39	0.16	0.99

Figure 8.6: T2K Match++ Workflow applied on an excerpt of a table about airports.

similarity during the value comparison. Hence, the label deviation of the entity *ltu iinternational airways* with its instance in DBpedia can be overcome. In contrast to the direct matching approach, see Figure 6.4, all correct correspondences are generated with confidence scores between 0.62 and 0.99

8.2 Related Work

Section 6.4 and 7.3 already discussed systems for matching web tables to knowledge bases according to different aspects. The field of related work that applies holistic matching or domain knowledge is significantly smaller. In this section, we present how holistic matching and domain knowledge is used by other methods.

8.2.1 Domain Knowledge

Approaches that exploit domain knowledge are also referred to as constraint-based or in the field of ontology matching as structure-based techniques. They have been employed for database and ontology matching [Rahm and Bernstein, 2001, Euzenat and Shvaiko, 2007]. In contrast to other methods, they do not only rely on comparisons between labels or values but infer constraints or compute similarities based on structural characteristics. For matching databases, the data types [Do and Rahm, 2002] and the cardinality [Navathe et al., 1986] are most common to be exploited. When matching ontologies, domain and range restrictions [Noy and

Musen, 2000,Lambrix and Tan, 2005] as well as definitions of the according ontology language can be utilized [Ehrig and Sure, 2004]. In addition, based on disjointness axioms defined by the ontologies, incorrect correspondences can be removed due to their inconsistency [Meilicke et al., 2007]. Besides matching, similar methods have also been applied in the field of knowledge base creation [Suchanek et al., 2007]. All mentioned methods have in common that the constraints based on information which is specified within the sources. For web tables or other types of sources such information is not available.

Domain knowledge gathered from data Instead of relying on constraints that are provided by the formal schema, domain knowledge can also be inferred from the data itself, e.g., by finding value patterns or computing value statistics. As one approach, recognizers are employed which generate a dictionary listing all possible values that occur in a schema [Doan et al., 2012]. If most of the the values appear in another schema, the recognizer concludes the similarities of the schemata. The strategy is especially beneficial if a schema can only cover a fixed set of values, e.g., the parental guidance movie classification.

One systems for matching database schemata by exploiting domain knowledge gathered from the data is SEMINT [Li and Clifton, 2000]. For each data type, SEMINT defines a set of characteristics. Among others, this includes the standard deviation for numeric values or the ratio of white space for strings. It builds vectors on these features and uses them as input for a neural network. Another field of research in which constraints are used is outlier detection. A family of approaches are statistical technique which profile the normal data, e.g., by maintaining histograms, and filter out anomalous data instances [Chandola et al., 2009]. Often, outlier approaches are mostly concerned with validating object-valued statements. In contrast, [Fleischhacker et al., 2013] detect errors in numerical properties in DBpedia by confirming facts using LOD datasets. Further, outlier detection is also used to find incorrect links between LOD datasets [Paulheim, 2014]. As mentioned during the introduction of the knowledge-based classifier, such characteristics between web tables and the knowledge base can significantly differ since web tables only presents a restricted view on a domain.

The LSD system [Doan et al., 2001] applies a set of learners. Each learner exploits a different part of information with the goal to match database schemas to one mediated schema. Some of the learners incorporate constraints like the characteristics of the value distribution. These constraints can be very domain specific, e.g., it learns for the real estate domain that the average number of rooms does not exceed 10. The same ideas are applied in the iMAP system [Dhamankar et al., 2004] to find complex mappings and by [Madhavan et al., 2005] to match schemas among each other. However, these methods focus on a set of particular domains and are not suitable for cross-domain knowledge bases.

Domain knowledge for matching web tables DSL [Pham et al., 2016] is a domain-independent system that matches the schemas of different data sources, among others also attributes of web tables, to ontologies. Besides similarities based on attribute labels and values, a distribution as well as histogram similarity is used. These similarities base on the same ideas as parts of our knowledge-based classifier. All features serve as input for a logistic regression. Another approach including domain knowledge for matching web tables to a knowledge base is implemented by [Mulwad et al., 2013]. Different modules can be included that for example identify whether an attribute contains data such as phone numbers. Using this information, more convenient similarity measures can be applied.

8.2.2 Holistic Matching

The term “holistic matching” has been used in various contexts. In general, the idea is to go beyond pairwise comparisons [Bellahsene et al., 2011]. Two main directions are considered as holistic [Rahm, 2016]:

- Build groups to decrease the amount of comparisons.
- Combine information to improve the matching quality.

Holistic matching to decrease the amount of comparisons Focusing on building groups, one strategy for schema matching is to use co-occurrences of attributes in schemata to decide whether attributes might have the same meaning [He and Chang, 2004]. Altogether, the goal is to integrate n schemas all at once and not one after another. Similarly, approaches for data matching mainly base on the idea to build cluster of entities in which all entities refer to the same real-world object [Hassanzadeh et al., 2009]. Such holistic methods exploit the amounts of sources to be matched in order to avoid pairwise comparison of each source to all other sources or to a target source. Our approach still performs pairwise comparison of each table with the knowledge base but combines information from other tables to improve the results.

Holistic matching to combine information The second direction of holistic methods gather evidence from sources that are also matched to the same target. Again, these strategies can be applied for schema [Do and Rahm, 2002] as well as data matching [Hartung et al., 2013]. If correspondences already exist, either characteristics can be learned or the transitivity can be exploited. [Madhavan et al., 2005] use existing mappings to learn typical characteristics of attributes as well as derive constraints for specific domains. A method to exploit the transitivity of correspondences, also called mapping composition, is provided by [Hartung et al., 2013]. Thus, new correspondences can be derived which leads to an improved coverage of the schemas to be matched.

Holistic methods for table to table matching The InfoGather system [Yakout et al., 2012] generates topic-specific graphs under consideration of indirect mappings in order to augment a query table with values found in a web table corpus. One augmentation method is *augmentation by attribute name* where the query table consists of an entity label column together with a second column for which an attribute label is specified but the values are missing. Hence, the task is to find the according values for the specified entity-attribute combination. First, the direct match approach (DMA) selects tables with overlapping entity labels that also share the attribute label of the augmentation attribute. Since the labels are compared using string equality, a low recall is expected (around 0.33). Depending on the domain of the query table, the precision can also be low due to the ambiguity of entity labels, e.g., names of cell phones are the same as name of camera models. Thus, additional information is gathered by considering indirectly matching tables in order to improve both, precision and recall. At first, the tables of a corpus consisting of 573 million web tables are matched among each other. For this, a combination of table and context features to compute the similarities between the tables is used. These similarities serve as weights for edges within a graph representing each table of the web table corpus as node. By using the topic sensitive pagerank, similarities between the query table and the tables in the graph are computed by considering both, direct as well as indirect paths. Finally, the similarities are aggregated to receive the value that is chosen to be augmented. To evaluate the augmentation by attribute name, six datasets covering tables about cameras, movies, baseball, albums, uk-pm, us-gov have been considered. The query tables for the experiments are generated by randomly selecting entity and attribute labels from the tables in the according datasets. The evaluation on value level shows that on average, a precision of 0.8 with a recall of 0.97 can be achieved which is 0.15 and 0.61 better than the results of direct match approach, respectively. Besides the input sources, a difference to our approach is that we restrict ourselves to indirect matches of the path length 2 and do not consider tables that only match indirect to already indirect matching tables. However, it is not clear whether further levels on indirect matches provide any benefit. Another interesting finding by the authors is that automatically generating attribute synonyms as proposed by [Cafarella et al., 2008b] is not a useful approach without manual intervention.

Holistic methods for web table to knowledge base matching For matching web tables to a knowledge base, [Cafarella et al., 2008b] follows a similar idea as [He and Chang, 2004] and create an attribute correlation statistic database based on a corpus of web tables. These statistics are used to create synonyms for attribute labels. Hence, these synonyms help to resolve terminological heterogeneities. The approach is similar to the creation of the synonym dictionary as described in the previous Chapter (Section 7.2). In contrast, we require a web table corpus that is already matched to a knowledge base such that the information are only available for subsequent matching executions.

8.3 Evaluation

In this section, we present the evaluation of the T2K Match++ method. Beside the T2D gold standard, we apply T2K Match++ on a second gold standard, the Limaye gold standard [Limaye et al., 2010], which is frequently used by other web table matching systems. We report the overall results for both gold standards, discuss the achieved performances for different steps, and analyze the results of each individual matching tasks in detail. Finally, a comparison to the results of state-of-the-art systems is given.

8.3.1 Experimental Setup

To be able to use indirect matches, correspondences between web tables and the knowledge base need to be available. Thus, for our experiments, we use the WDC WTC 2012 with 33 million tables and DBpedia (English version 2014) as input. Since we do not know all correspondences of tables in the corpus, we measure the performance of T2K Match++ on the T2D as well as on the Lim_{dbp} gold standard.

Lim_{dbp} We annotate tables from the gold standard presented by [Limaye et al., 2010] with class, property, and instance correspondences to DBpedia. In more detail, we reuse the set of 200 randomly chosen tables from the Limaye gold standard as introduced by [Zhang, 2016]. Most of the tables, around 94% have been collected from Wikipedia, the remaining ones originate from general web pages that overlap with Wikipedia tables. Table 8.2 depicts the statistics for the Lim_{dbp} gold standard. Altogether, 173 class correspondences have been created, 406 property- (with 173 label correspondences) and 3 465 instance correspondences. It is especially conspicuous that the tables cover on average a similar amount of attributes (5) as the overlapping tables in T2D while the average number of rows is considerably less. Thus, especially for the property and class matching task, a smaller amount of information is available. The lower number of rows per table obviously results in a smaller set of correspondences to instances in DBpedia. The amount of property and class correspondences are in the same order of magnitude.

Table 8.2: Statistics about entities and attributes in the Lim_{dbp} gold standard.

	#	Average	Median	SD	#Corr.	Mapped Ratio
Rows	3679	22.01	14	21.59	3465	0.94
Columns	764	4.53	4	1.55	406	0.53

Compared to the T2D gold standard, the tables especially cover considerably less rows (3 679 vs. 28 595) but with a comparable amount of attributes (764 vs. 1163). Thus, the Lim_{dbp} tables are significantly smaller but with a similar width. In contrast, the mapped ratios are almost the same for both gold standards.

The topics described by tables in the Lim$_{dbp}$ gold standard are depicted in Figure 8.7. About 40% of all tables are about persons. Together with tables describing populated places, 70% of the tables in the gold standard are covered. In contrast, in T2D the most common topics are places and works but only with a respective proportion of 20% which indicates a broader coverage of topics.

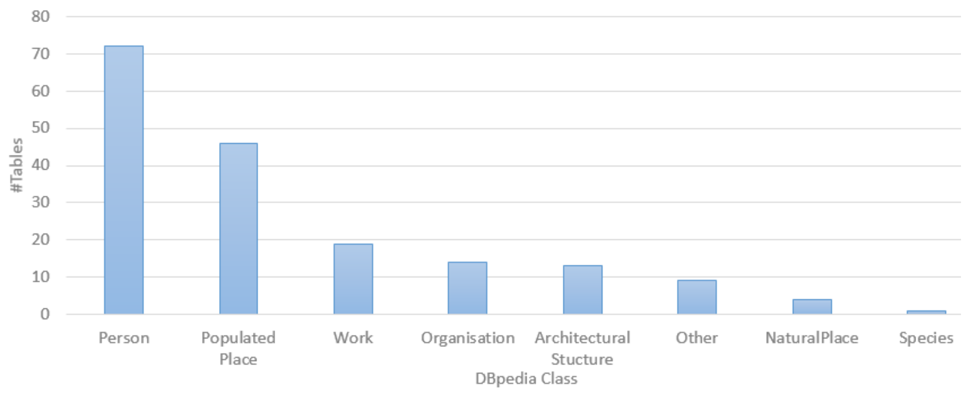


Figure 8.7: Number of instance correspondences per DBpedia category in the Lim$_{dbp}$ gold standard.

As we already discussed in Section 6.4.4, the main differences between the T2D and Limaye gold standard, and in turn the Lim$_{dbp}$ gold standard, are the variety of sources, the availability of datatype property correspondences and the incorporation of non-overlapping tables. In contrast to web tables gathered from arbitrary pages, other characteristics are provided. For example, almost every Wikipedia table has an attribute label which does not hold for other web tables.

8.3.2 Overall Results

In this section, we evaluate the overall performance of T2K Match++. Table 8.3 shows the results for all three matching tasks on T2D and Lim$_{dbp}$. The results are divided into three parts: only direct matching (dir), direct & indirect matching (ind) and including the improved classification (all).

Results on T2D Only applying the direct matching leads to decent results for all tasks as presented in the previous Chapter (Section 7.4), i.e., an F-measure of 0.79 for the instance, of 0.81 for the property, and of 0.92 for the class matching task. By additionally performing the indirect matching step, the results for the instance and class matching tasks can be improved but the precision of the property matching task drastically drops. Besides useful evidence and correct correspondences, indirect matches can also bring incorrect correspondences into play. This especially affects the property matching task due to its characteristics: while for 92% of all rows DBpedia includes an instance referring to the same real-world

Table 8.3: Results of T2K Match++ for all three matching tasks on T2D and Lim$_{dbp}$.

		Precision			Recall			F-measure		
	Task	Dir.	Ind.	All	Dir.	Ind.	All	Dir.	Ind.	All
T2D	Instance	0.81	0.90	**0.94**	0.77	0.79	**0.80**	0.79	0.84	**0.87**
	Property	0.76	0.59	**0.95**	0.86	**0.95**	0.86	0.81	0.73	**0.91**
	Class	0.93	0.95	**0.95**	0.91	0.93	**0.93**	0.92	0.94	**0.94**
Lim$_{dbp}$	Instance	0.90	0.86	**0.95**	**0.83**	0.82	0.81	0.86	0.84	**0.87**
	Property	0.59	0.53	**0.89**	0.77	**0.85**	0.73	0.66	0.65	**0.80**
	Class	0.85	0.88	**0.88**	0.85	0.88	**0.88**	0.85	0.88	**0.88**

object, for only 57% of the attributes a DBpedia property with the same meaning exists. Thus, assigning a property to an attribute is more likely to be incorrect than assigning an instance to an entity. In addition, even by solely performing the direct matching, the precision for the property task is only about 0.75 which means that 25% of all correspondence in tables with the same topic are already incorrect. By using the knowledge-based classifier exploiting domain knowledge, the incorrect property correspondences are detected and in turn the decrease of precision for the property matching task can be reversed. Additionally, the performance for the instance matching task can be improved the indirect matching and a classification that goes beyond thresholding. For the class matching task, precision and recall are accomplished by the indirect matching but to a smaller amount (0.02) since the F-measure achieved by direct matching is already above 0.9. Thus, with the indirect matching, we are able to solve some of the remaining corner cases.

Results on Lim$_{dbp}$ With the direct matching applied to the Lim$_{dbp}$ gold standard, an F-measure of 0.86 for the instance, of 0.66 for the property, and of 0.85 for the class matching task can be achieved. One reason for the high instance performance is that entities within Wikipedia tables often have the exact same label as the instance in DBpedia. To underline the evidence, we found out that 65% of all entities share exactly the same label as the according DBpedia instance. Contrary, although almost every attribute has a label, the property matching performance is comparatively low. The tables cover on average only 22 entities which leads to a smaller amount of values that can be taken into account for the duplicate-based method and in turn, little evidence is available to decide which property fits best, if any. Further, about 70% of all tables from the Lim$_{dbp}$ gold standard are about persons and populated places: two of the DBpedia classes with the highest amount of properties which in turn leads to an increased amount of candidates for each attribute. Additionally including the indirect matches slightly worsen the results for all tasks except for the class matching where an increase of 0.03 in F-measure is detected. Improvements for the class matching result from the fact that less actually overlapping tables are filtered out due to missing evidence. In contrast, the instance matching performance decreases since the indirect matches from the comparably noisy and error-prone web table corpus leads to incorrect decisions. Including the more advanced classification, the best results for all three tasks can be achieved. Especially for the property matching, the F-measure can be raised by 0.14 compared to the direct matching.

Although the gold standards vary according to different characteristics like the table size or the topical distribution, in most cases similar behavior can be observed. For all three tasks, the best results are achieved if the combination of direct and indirect matching together with the adapted classification is applied. Only considering indirect matches can, but does not need to, improve the instance matching results, depending on the table characteristics and especially on the expected quality of the correspondences that are used during the indirect matching. On both gold standards, including indirect matches decrease the results of the property matching since almost every attribute is incorrectly assigned with a property and these incorrect correspondences cannot be filtered by the threshold-based classification. Thus, the need for the knowledge-based classification becomes evident in both datasets. Overall, the performances achieved on both gold standards are in the same range, except for the property matching task where an increase in the performance between 0.17 (direct) and 0.11 (all) for T2D is determined. As mentioned, one reason is the size of the tables since tables from Lim_{dbp} are smaller which reduces the amount of evidence on which the algorithm can rely on.

Comparison to InfoGather InfoGather [Yakout et al., 2012] is the only system that includes indirect matches for matching web tables among each other, for the use case of table augmentation. On value level, they report on average a precision of 0.8 with a recall of 0.97. Thus, similar F-measures can be achieved although the results are not directly comparable. Further, the InfoGather system achieves an increase in precision of 0.15 and in recall of 0.61 in contrast to only applying the direct matching approach. For T2K Match++ we cannot detect such large recall improvements. This mainly bases on the direct matching method: InfoGather requires the equality of attribute labels in order to consider two web tables as match. Hence, the results of the direct matching approach tend to have a high precision with a low recall which is not the case for our direct approach. Apart from the discrepancies, the InfoGather outcomes support our findings that holistic matching is suitable for web tables and both, precision and recall can be improved.

8.3.3 Detailed Evaluation

While the previous section focused on the evaluation of the performance of the overall algorithm this section provides details on specific parts of our approach as well as analyzes which direct matching errors can be avoided. Thereby we focus on the T2D gold standard since on the one hand it covers a wider range of cases and on the other hand, the results for both gold standards are mostly in line.

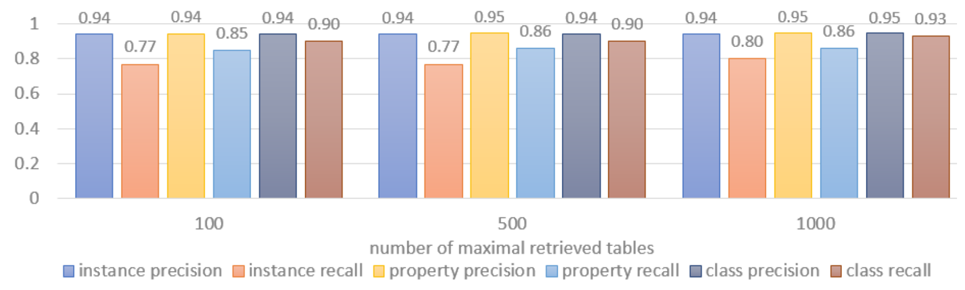


Figure 8.8: Performance changes of different maximal numbers of retrievable tables in the table search.

Table Search During the indirect matching, we search for similar tables that have also been matched to DBpedia. One important step is the table search where similar tables are retrieved by querying an index using the entity labels. Following, we analyze whether and to which extent the maximal number of tables retrieved by the index has an influence on the performance in terms of F-measure. In Figure 8.8, the results using three different numbers, 100, 500, and 1 000, of maximal retrieved tables are shown. In general, retrieving more tables leads to better results but only slight changes are noticeable. For the instance matching task, an improve in recall by 0.03 is determined when increasing the number of tables from 500 to 1 000. Thus, for some entities the top 500 tables are not enough since these tables might not contain similar entities, e.g., a particular song will not be found in every table about songs even if other songs of that table are referred. Concerning the property task, precision and recall is improved by 0.01 when taking 500 instead of 100 tables into account. In contrast to thousands of entities that can exist, usually a restricted set of attributes is used to describe a domain. Hence, it is not necessary to consider more tables but it does not harm. The results of the class matching task show again a similar behavior as the results of the instance matching task: the precision and recall can be increased by 0.01 and 0.03 when rising the number of tables from 500 to 1 000 respectively. Using more than 1 000 retrievable tables does not lead to an improvement of the results, and increases the runtime only.

Table Matching After retrieving the relevant tables from the index, the tables are matched among each other. This includes the entity, attribute, and table matching. For all three tasks, we analyze how many similar tables and elements we detect during the matching. Starting with the table matching task, Figure 8.9 presents the cumulative distribution function of the number of tables that are considered as similar. A table is considered as similar if at least one correspondence to an instance in DBpedia is shared. Since we set the maximal number of received tables to 1 000 more than 1 000 tables can never be similar. As the distribution states, for more than 90% of the tables in T2D, no more than 900 tables are similar. For 10% of the tables in T2D, we do not even find one table describing overlapping entities. Hence, the results of these tables cannot be improved by the indirect matching. Ex-

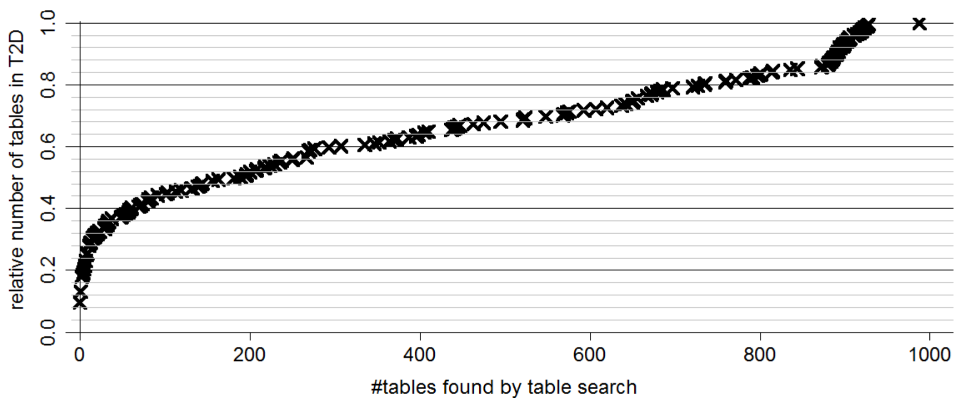


Figure 8.9: Distribution of detected similar tables per table in T2D.

amples are particularly tables containing long-tail entities, e.g., about hospitals in a very specific area. More than 40% of the tables are similar to less than 100 tables. This includes tables about topics like newspaper or museums. Only for a small amount of tables we get close to the limit of $1\,000$. Such tables are about works like video games or places. One reason for the amount of similar tables is of course the distribution of tables in the corpus. However, a large amount of tables about a certain domain in the corpus does not necessarily mean that the amount of similar tables is also high. According to the topical distribution of web tables as outlined in Chapter 5, tables about songs are frequent in the corpus. However, for a small table with specific songs, only a small amount of similar tables will be found, if any.

Besides the amount of similar tables, different amounts of similar entities and attributes can be detected. Figure 8.10 shows the distributions of similar entities and attributes that can be found during the table matching, respectively. Both distributions indicate a similar behavior: for a large portion of entities and attributes (20% and 34%) no similar elements can be detected, for 13% of the entities and 26% of the attributes only a few elements can be found but for 70% of the entities and 40% of the attributes more than 10 elements are considered as similar.

Mapping Composition To know whether the amount of similar elements has an influence on the results, we computed the correlation (Pearson correlation coefficient) between the number of similar elements to the correctness of the generated correspondence. The number of detected similar elements has a positive correlation of 0.1 for entities and of 0.29 for attributes (statistically significant, TTest $\alpha = 0.05$). Hence, if no similar attribute can be found, it is an indication that no correspondence to a DBpedia property exist. In contrast, for entities this does not necessarily hold since especially long-tail entities are not found in other tables. For example, almost $1\,000$ mountains do not have any similar entities but the correspondences to instances created by the direct matching are mostly correct.

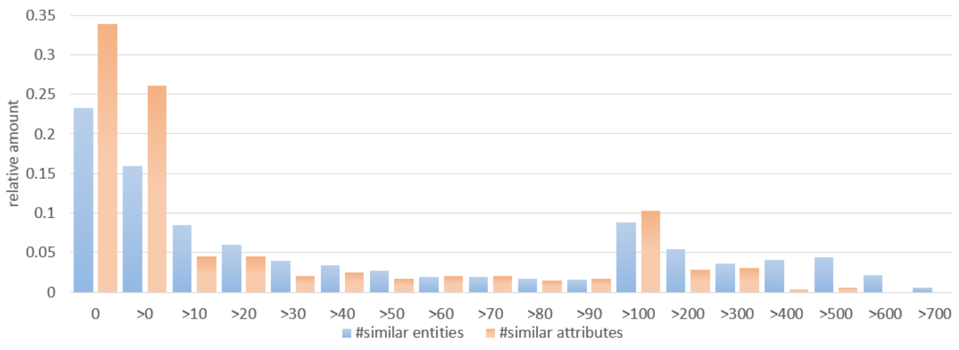


Figure 8.10: Distribution of detected similar entities per entity and attributes per attribute for tables in T2D.

Classification The classification decides whether a correspondences is considered as match or non-match based on decision trees that are learned for the matching tasks. Figure 8.11 and Figure 8.12 depict the decision trees for the instance and class matching task, given the features described in Section 8.1.3 as input. Each leaf associated with “1” is a match, “0” means a non-match.

For the instance matching task, the most upper decision is whether the final score is above 0.54. Since the final score is the combination of the direct as well as indirect matching, either both steps need to return a certain score or a sufficiently high score from one of the matching steps is required. The sum of the label scores is again a features that includes scores from both steps. Since the value in the condition of the decision tree is slightly above 1.0, it is required that a label similarity is detected by both matching steps. Based on the previous findings, we know that correspondences can be correct even if no similarities are detected by the indirect matching. This is also reflected in the decision tree: all following conditions are concerned with features originating from the direct matching step. This includes the differences to similarities of other entities in the same table, to all other entities in other tables of the gold standard and the similarities with their weights.

The decision tree learned for the class matching task covers slightly less nodes. The root condition proves whether the proportion of entities for which a correspondence to a knowledge base instance has been generated to the total amount of entities in the table is equal or less than 0.6. If additionally a low direct score is determined, the correspondence between the table and the class is not likely to hold. Hence, if the class decision only bases on a few instance correspondences for which the evidence is not even high, the table does most probably not correspond to this class. Otherwise, if more than 60% of the entities have a counterpart in the knowledge base, even a lower direct score is enough to decide for the correspondence. If more than a total of 11 entities have been matched to instances and the final score exceeds a certain value, the correspondence is not excluded.

Comparing both trees, different features seem to be important for the instance

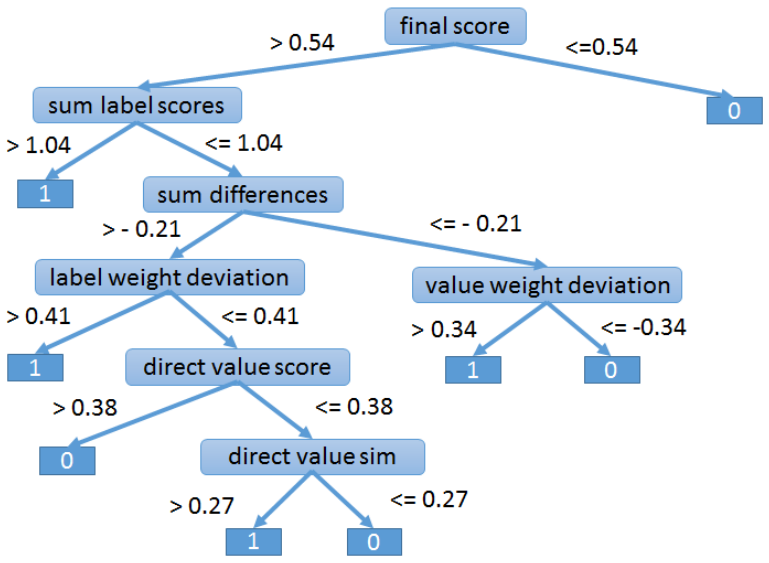


Figure 8.11: Decision tree learned to classify instance correspondences.

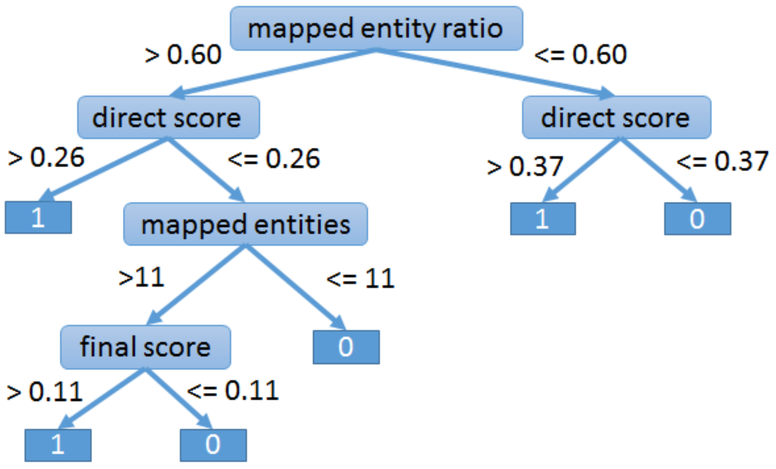


Figure 8.12: Decision tree learned to classify class correspondences.

and class matching task. Especially features combined from both, indirect and direct matching, are essential for the instance matching task. Contrary, for the class matching task, table characteristics are more helpful. One obvious reason is that the class matching task without indirect component already achieves an F-measure above 0.9 such that only slight improvements are possible. In contrast, an F-measure increase of 0.08 is determined for the instance matching task.

Knowledge-based Classification In addition to features used by the other two tasks, characteristics derived from the knowledge base are included as features to classify property correspondences. Since the features can refer to particular data types, an individual decision tree for each data type is generated. Figures 8.13, 8.14, 8.15 depict the decision trees learned for the property matching task. All trees include knowledge-based features and features generated during the direct and indirect matching. Further, features basing on both - value and label similarities - are taken into account. We will analyze the features derived from the knowledge base independently of the classifier in a correlation analysis in Section 8.3.5.

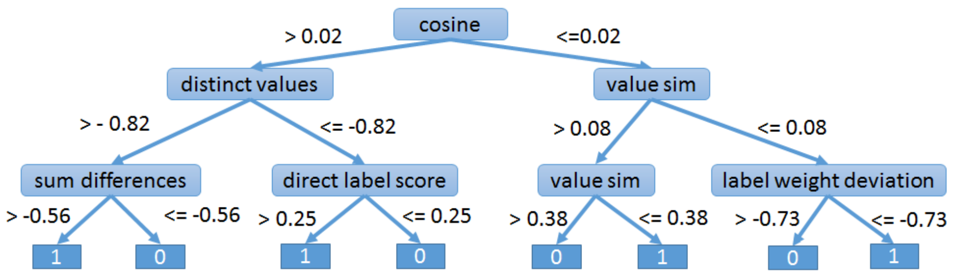


Figure 8.13: Decision tree learned to classify string property correspondences.

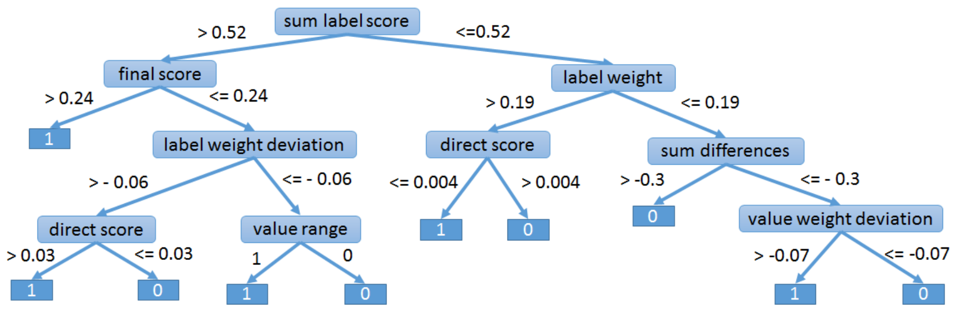


Figure 8.14: Decision tree learned to classify date property correspondences.

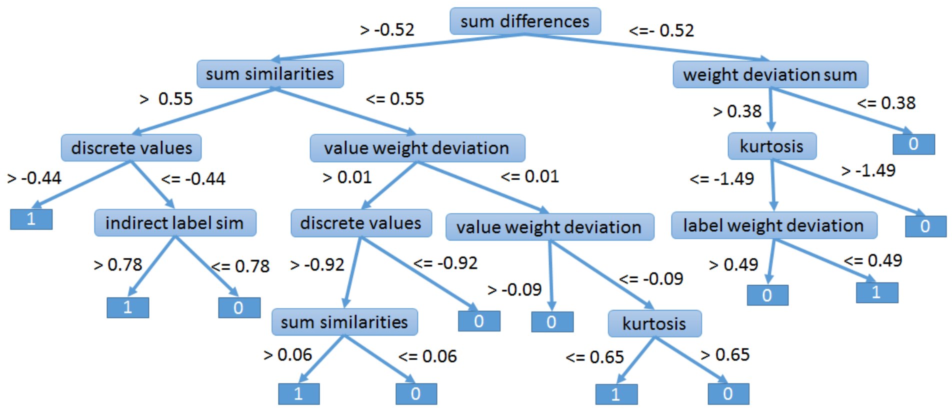


Figure 8.15: Decision tree learned to classify numeric property correspondences.

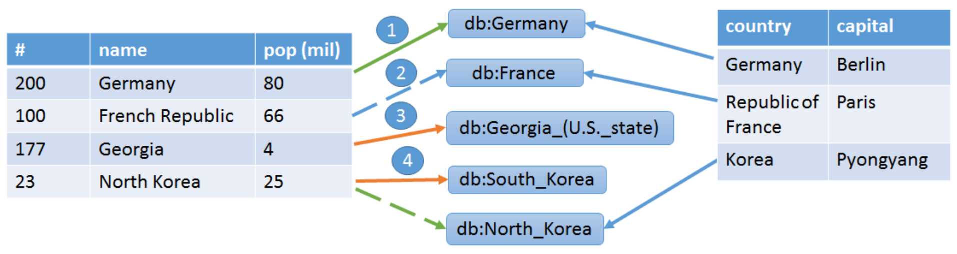


Figure 8.16: Different cases for improvement

Improvement Analysis For all three matching tasks, we know the precision and recall changes that are achieved by applying T2K Match++. However, different cases of improvements can be responsible for the overall performance increase. Figure 8.16 introduces the four cases where indirect matches can improve the matching results, in this example of the instance matching task. Green arrows indicate correct correspondences, red arrows incorrect correspondences and dashed arrows missing correct correspondences. In general, we can either gather positive or negative evidence such that the confidence of correct correspondences can increase and the confidence of incorrect ones can be decreased.

- **Case 1** In the first case, a correspondence between the entity and the correct instance can be found with direct comparison but the correspondence might be filtered out during the classification since the confidence is not high enough. Reasons for a low confidence could be that values do not overlap or no correspondences to properties can be found for the table. If we find similar entities in other tables that are also mapped to the same instance, we can be more reliable that the correspondence is indeed correct. As example, we can confirm the correspondence between the entity with the label *Germany* to the DBpedia instance called *Germany* because other entities with the same label are also matched to this instance.
- **Case 2** For entities for which no suitable instance is detected during the direct comparison, we can check to which instances similar entities have been matched. In the example, we do not find the DBpedia instance *France* as candidate for the entity *French Republic* because the labels are not similar enough. However, the corpus covers similar entities which are at the same time also similar to the preferred instance.
- **Case 3** Besides confirming a correspondence, we can also gather negative support which can lead to the exclusion of correspondences. Incorrect correspondences can for example be generated if the labels accidentally fit and we do not have any further evidence like similarities on the value level. If we rarely find this incorrect correspondence for similar entities, it is an indication that the correspondence is unlikely to hold. As example, a correspondence is generated by the direct matching between the entity *Georgia*

and the DBpedia instance named *Georgia* which describes the state in the U.S. instead of the country located in Europe. This correspondence cannot be confirmed by the correspondences detected for similar tables.

- **Case 4** A combination of case 2 and 3 is represented by this case. On the one hand, for a particular entity we find a correspondence during the indirect matching that has not been considered before and on the other hand we do not get any support for the incorrect correspondence. Due to the limited evidence, the entity *North Korea* is matched to *South Korea* which is obviously not correct. With the help of similar tables, the entity can be mapped to the correct DBpedia instance. Thus, a correspondence pointing to the wrong instance can be replaced by the correct correspondence.

To know which errors can be avoided by an additional indirect component, Table 8.4 depicts the precision and recall changes for the four cases of improvement. Regarding precision, gathering negative evidence for incorrect correspondences (case 3) is particularly important for all matching tasks. Among all tasks, an increased precision can be detected, even up to 0.18 for the instance and 0.2 for the property task. As expected, a high rise in precision is often at the expense of a loss in recall but the loss is limited. Cases 1 and 2 ensure a higher recall for all the tasks, while confirming correspondences (case 1) is more important for the instance and property matching task. For the last case, a small improvement can only be perceived for the class matching task. Thus, for the other tasks either only a missing correspondence can be found or an incorrect can be discarded but not at the same time.

Table 8.4: Precision and recall changes for the different cases of improvement.

	Precision per case				Recall per case			
Task	1	2	3	4	1	2	3	4
Instance	-2	-2	+18	-1	+3	+2	-4	+1
Property	-1	+0	+20	+0	+5	+1	-5	+0
Class	+0	-2	+2	+2	+0	+2	+1	-1

In the following sections, we provide in-depth analysis of the results of each task. Thereby we focus on the T2D gold standard since on the one hand they provide a more realistic scenario and on the other hand, the results for both gold standards are mostly in line.

8.3.4 Results of the Instance Matching Task

For the instance matching task, the direct matching step achieves an F-measure of 0.79. Considering indirect matches as well as applying the improved classification increases the F-measure by 0.05 and 0.03, respectively. The extent of the improvement is not the same for all entities but depends on their characteristics. Figure

8.17 presents the performances for the classes from the first level of the DBpedia class hierarchy to which the corresponding instances belong to. Instances that do not have one of these classes as superclass are not considered (about 4%). When we only apply direct matching, entities describing works have the lowest precision but at the same time the highest recall. A possible reason is the ambiguity of work names, e.g., the term “Love” occurs in a lot of different work names. In contrast, for entities referring to places which make up 42% of all entities, the detected correspondences are most often correct but we miss some of them.

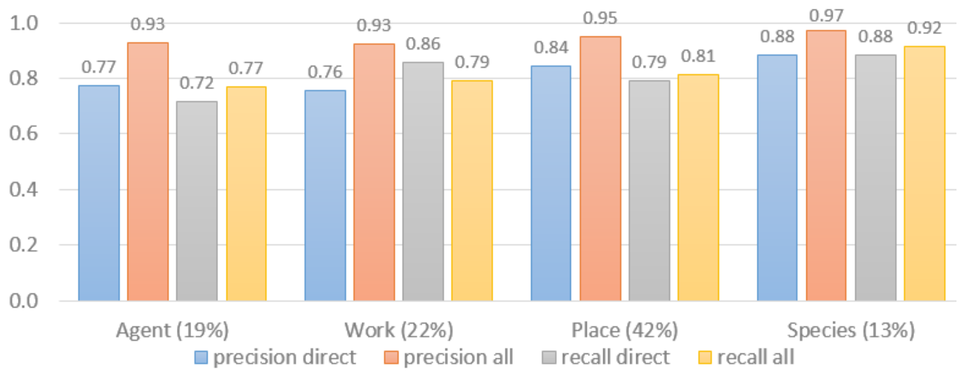


Figure 8.17: Instance matching performance differences per superclass on T2D.

If the direct matching in combination with the indirect matching and the classifier is applied, high improvements regarding precision can be detected for *Works* (0.17) followed by *Agents* (0.15). A simple reason is that the majority of tables in the web table corpus is about works and agents as introduced in Chapter 5. Thus, most evidence can be gathered during the indirect matching for instances belonging to those classes. For all other classes, relatively high rises in precision of around 10% are achieved. Also the recall is improved for most cases except for *Works* where a decline of 0.06 is attained. Obviously, if the precision of the direct matching is rather low, the similarities gained during the indirect matching can be misleading and in turn correct correspondences can be filtered due to low evidence. For other classes like *Agent* or *Species*, not only the precision but also the recall can slightly be increased by 0.05 and 0.03, respectively. Since almost half of the instances in T2D are of type *Place*, the increase in precision by 0.11 and the improvement in recall of 0.02 are conducive for the overall performance.

To get a more fine-grained view, Table 8.5 presents examples for classes of instances where significant performance changes in precision and recall compared to the results of the direct matching can be seen. For the classes *Bird* and *BaseballPlayer*, the recall increases by almost 0.4 and even comparatively large precision improvements can be achieved. These enhancements involve 9% (*Bird*) and 4% (*BaseballPlayer*) of all entities that have a correspondence to a DBpedia instance. In contrast, for *Saints*, the recall can be drastically increased but at the expense of precision. The improvements for instances of the classes *Airline* and

Table 8.5: Examples for significant instance matching task performance changes.

Class	#Rows	Precision	Recall
Bird	2304	+0.16	+0.34
BaseballPlayer	910	+0.18	+0.39
Plant	720	+0.24	+0.22
Airline	477	+0.20	+0.27
Lake	545	+0.12	-0.25
Hospital	310	-0.01	-0.40
Newspaper	65	-0.07	-0.18
Saint	455	-0.13%	+0.57

Plant are some of the largest especially regarding precision. Tables about airlines often include codes like the IATA-code which is a very distinct attribute. Decreased recall scores are found for *Hospital* and *Lake*, mainly due to the reason that we do not find these very specific long tail entities very often among the tables in the corpus and thus in contrast to other entities cannot confirm the detected correspondences. One example of a lower precision are instances of type *Newspaper*. As it is generally the case for works, newspaper names often refer to general terms, e.g., “the sun”. Since the indirect matching relies on the similarities for all tables computed during the direct matching, this issue cannot be fully overcome.

8.3.5 Results of the Property Matching Task

When matching attributes to properties, including indirect matches drastically decreases the precision by 0.17. T2K Match++ assigns a property to almost every attribute and a threshold-based classification is not able to distinguish between matches and non-matches. The lack of precision can be compensated by applying the knowledge-based classification exploiting domain knowledge. Overall, the precision can be increased to 0.95 while the recall almost remains the same, resulting in an F-measure raise by 0.1 compared to the direct matching.

Figure 8.18 depicts the performance changes per data type. The most drastic change can be seen for numeric attributes which make up 25% of all attributes in T2D. For them, an enhancement in the precision of 0.37 can be achieved. For strings, the precision improvement accounts for 0.09, for dates a raise in precision of 0.17 can be found. The reason for the higher performance change for numeric attributes is the variety of features that can be exploited during the classification. Regarding recall, an increase for numeric attributes by 0.15 but at the same time a slight decrease for string attributes is detected. Since string attributes occur more often, the in- and decrease compensate each other on the overall results.

Attributes for which correspondences are not found during the direct matching are often time- or location varying attributes, e.g., the population of a country or the release date of a film. For attributes of type string, in contrast to the di-

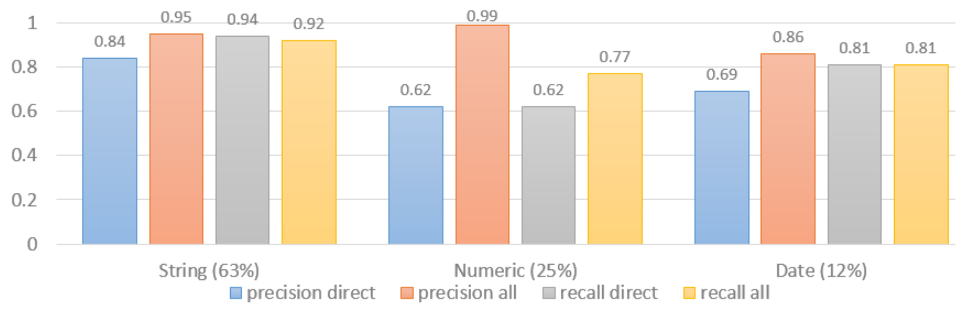


Figure 8.18: Property matching task results per data type on T2D.

rect matching, correspondences can be detected if the values include longer texts or terms that do not originate from a controlled vocabulary or use abbreviations. Thus, the values in the table are too different to the according DBpedia values. However, we find tables in the corpus whose values are similar and their attributes have been matched to a DBpedia property. This can happen for a set of reasons: more evidence is available because more entities are mapped to instances, the table contains additional/other entities or the extent of the variation between the values is slightly less.
The precision may be increased if incorrect correspondences are detected by the direct comparison but they cannot be confirmed when considering similar tables. One example is the mapping of "view date" of a film to its release date. Knowing that similar tables do not share the correspondence, it is classified as non-match.

Since the knowledge-based classifier is most important for the property matching task to improve the results, we analyze the influence of different features on the classification results. Table 8.6 depicts the Pearson correlation coefficients between the features with the largest influence and the correct classification decision. All correlations are significant (TTest with $\alpha = 0.05$), except for the final similarity score of string attributes, marked by an asterisk.

At first glance, we see that the correlations and their extent highly depend on the data type. As example, while we find correlation scores above 0.6 for numeric attributes, the highest score for string attributes is around 0.3. For attributes of data type string, the label similarity as well as the cosine similarity show the highest positive correlation, followed by the value similarity. All other features do not seem to have an influence on the result. In contrast, for numeric attributes a variety of features are even highly correlated. Besides the similarities that are computed during the matching, the distinctness seems to play an important role. If the attribute and the property do not share the same characteristics regarding the distribution of distinct values, the correspondence will probably be incorrect. Further, the features that are specifically included for numeric attributes are all positively correlated, especially the fraction of discrete values.
Regarding date attributes, the label similarity is even negatively correlated with the

Table 8.6: Pearson correlation coefficients between different features and the knowledge-based classification.

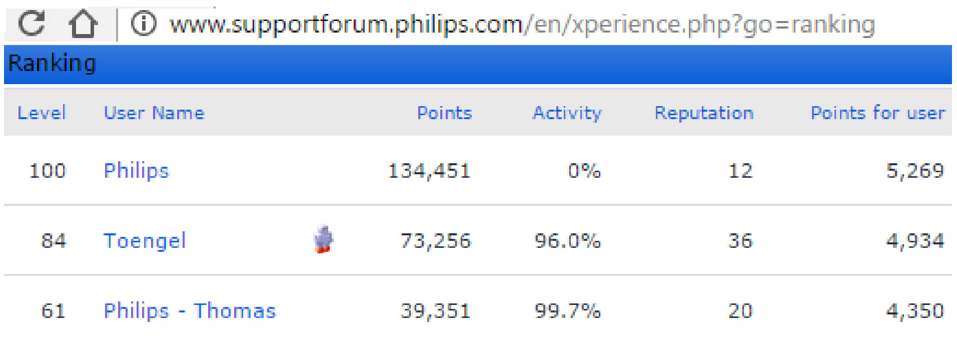
Feature	String	Numeric	Date
Direct Label Similarity	0.31	0.61	-0.17
Direct Value Similarity	0.22	0.31	0.40
Final Similarity	*0.10	0.60	0.00
Distinct Values	-0.05	-0.52	-
Functionality	0.00	0.00	-0.26
Cosine Similarity	0.30	-	-
Value Ranges	-	0.13	0.28
Discrete Values	-	0.33	-
Kurtosis	-	-0.23	-

correspondence label. This indicates that a high similarity (regarding our similarity measure) between the attribute and property labels tends to lead to an incorrect correspondence. A reason are terms like ”date“ which frequently occur in attribute and property labels. The final similarity score that is used for thresholding does neither have a positive nor a negative correlation. For filtering incorrectly matched date attributes, especially the functionality and the fact whether the median of the values lies in the range of the property is important.

In general, the domain knowledge features act as plausibility checks but they cannot be used without other features like the value similarities.

8.3.6 Results of the Class Matching Task

Regarding the class matching task, we can detect improvements of 0.02 in precision as well as recall compared to the direct matching.



Ranking

Level	User Name	Points	Activity	Reputation	Points for user
100	Philips	134,451	0%	12	5,269
84	Toengel	73,256	96.0%	36	4,934
61	Philips - Thomas	39,351	99.7%	20	4,350

Figure 8.19: Table with a correspondence to the DBpedia class *Person*.

The incorrect correspondences that can be avoided when taking indirect matches into account are mainly assignments of classes that happen due to accidentally fitting labels of instances to entites. Figure 8.19 shows an excerpt of a table that has been assigned to the class *Person* during the direct matching. For most of the entities, a person is found in DBpedia whose label is similar. However, none of the entities actually refers to a person in DBpedia. However, the incorrect instance correspondence cannot be filtered out based due to the similarity. Querying the index during the indirect table search results in an empty set of relevant tables. Thus, no table is similar which provides the crucial hint that no class should be assigned since the table does not overlap with DBpedia. Improvements in recall result from correspondences which have not been generated before due to low evidence. Among other reasons, this can happen if a table only includes a small amount of rows. With the additional evidence we get during the indirect step, we are able to assign the correct class.

8.3.7 Comparison with State-of-the-Art

All systems matching web tables to knowledge bases have been presented in previous chapters. While some of them use tables from the T2D gold standard for evaluation, most have been applied on tables from the Limaye gold standard. Firstly, we compare T2K Match++ to systems using T2D and secondly, discuss the evaluation on tables from the Limaye gold standard.

Table 8.7: Results of web table matching systems evaluated on tables from T2D.

	F-measure		
System	Instance	Property	Class
T2K Match++	**0.87**	**0.91**	**0.94**
LogMap [Efthymiou et al., 2016]	0.70	-	-
[Efthymiou et al., 2017]	0.85	-	-
TAIPAN [Ermilov and Ngomo, 2016]	-	0.51	-
DSL [Pham et al., 2016]	-	0.77	-

Approaches evaluated on tables from the T2D gold standard Table 8.7 presents the results of matching system that use tables from the T2D gold standard for their evaluation. Altogether, four approaches have been evaluated, i.e., on the set of overlapping tables in T2D. Since they directly reuse the annotated correspondences, all of them use DBpedia as knowledge base. None of the systems performs all three matching tasks, they either focus on the instance or property matching task. LogMap has been developed to match ontologies [Jiménez-Ruiz and Cuenca Grau, 2011] but has been applied on web tables to evaluate how well ontology matching tools are able to deal with web tables [Efthymiou et al., 2016]. Hence, an F-measure of 0.7 for the instance matching task is achieved without any adaptions to the specific characteristics of web tables. The result is close to the performance (0.68 F-measure) we reported in Chapter 7 when only the entity labels are used as features. Recently, [Efthymiou et al., 2017] presented a system exploiting word embeddings to compare entities with instances. The reported F-measure of 0.85 is very close to the result of T2K Match++. The other two systems, TAIPAN [Ermilov and Ngomo, 2016] and DSL [Pham et al., 2016], focus on the property matching task, resulting in an F-measure of 0.51 and 0.77, respectively. TAIPAN retrieves property candidates using the attribute label and applies a probabilistic model to find the best suitable property. Besides the tables in T2D, they include additional tables in their gold standard such that the results are difficult to compare. However, the achieved performance is slightly below the results we stated when only considering the attribute label (F-measure of 0.63). DSL uses statistical features based on the attribute and property distribution which are similar to our domain knowledge features. With this strategy, an F-measure of 0.77 is determined which is comparable to the performance of our direct matching method (0.81). T2K Match++ outperforms all approaches that have been applied on tables from the T2D gold standard. Since they all focus on a particular matching task, they cannot take advantage of the information gathered from the other tasks. Both approaches, LogMap and TAIPAN take only a small set of features into account, mainly relying on the label which is not sufficient to achieve best possible results. However, as shown by [Efthymiou et al., 2017], with a more sophisticated method including word embeddings, a high performance can be reached even if a small set of features is considered. With additional features capturing different characteristics the performance can be increased as shown by the DSL system. Nevertheless, none of the systems attains F-measure scores around 0.9.

Approaches evaluated on tables from the Limaye gold standard In the literature, we find five systems that are evaluated on tables from the Limaye gold standard. Originally, the gold standard contains correspondences to the knowledge base YAGO. Over time, the tables have also been annotated with classes, properties and instance from the knowledge bases DBpedia and Freebase. Since different knowledge bases provide different characteristics, the results are not directly comparable. Further, except for the approaches introduced by [Venetis et al., 2011], the methods are evaluated on individual subsets of tables. Additionally to the differences in the data, the systems only consider named entity attributes and do not annotate attributes including literals. Further, they assign instances to each named entity found in the table, regardless of whether they are contained in the entity label attribute. Thus, web tables that we consider as non-relational because they do not contain an entity label column are nevertheless matched by the other systems.

In Table 8.8 the achieved performances of the matching systems are presented. A weighted average is used whenever necessary, i.e., if systems are evaluated on several subsets of tables for particular tasks. By leaving the described discrepancies aside, T2K Match++ achieves the highest results for all three matching tasks. One reason for the partially large difference in F-measure for the class matching task, especially for the system proposed by [Limaye et al., 2010] and [Mulwad et al., 2013], might be caused by the choice of the knowledge base. While DBpedia cov-

Table 8.8: Results of web table matching systems evaluated on tables from the Limaye gold standard.

		F-measure		
System	Knowledge Base	Instance	Property	Class
T2K Match++	DBpedia	**0.87**	**0.84**	**0.80**
TabelMiner+ [Zhang, 2016]	Freebase	0.84	0.76	0.75
[Limaye et al., 2010]	YAGO	0.84	0.58	0.45
[Mulwad et al., 2013]	YAGO, DBpedia	0.76	**0.84**	0.55
[Venetis et al., 2011]	YAGO	-	0.55	0.69
[Efthymiou et al., 2017]	DBpedia	0.82	-	-

ers a few hundreds of classes, YAGO includes hundreds of thousands of classes which potentially make it more difficult to determine the correct class. Regarding the other two tasks, the highest reported results are similar to the performance of T2K Match++. Except for TableMiner+, every system shows high F-measure scores for one but not for all tasks. Hence, large performance differences among the tasks up to an F-measure of 0.4 are detected. On the one hand, this indicates that good results for one matching task do not entail similar performances for the other matching tasks. On the other hand, gathering as much information as possible as TableMiner+ does, leads to more consistent results for all tasks. In comparison to the results of the best performing system TableMiner+, the F-measure for the difficult property matching task is increased by 0.08, for the class nmatching task by 0.05 for the instance matching task by 0.03.

In summary, all improvements we proposed within T2K Match++ lead to high results for all three matching tasks on two gold standards, outperforming all state-of-the-art matching systems.

8.4 Summary

In this chapter, we introduced one of the main contributions of this thesis: the T2K Match++ method which holistically matches web tables to knowledge bases. Altogether, it performs all three matching tasks in an integrated fashion (Section 6.2), using various features like the context of the web table (Chapter 7) together with a component that exploits indirect mappings of similar tables. By including an indirect matching step, more evidence can be gathered which overcomes the limitations of matching web tables in isolation (direct matching). Additionally, considering domain knowledge for the property matching task which is derived from the knowledge base complements the approach. To the best of our knowledge, no system exists for matching web tables to knowledge bases which takes indirect matches into account.

On the T2D gold standard, F-measure scores around 0.9 can be achieved for all matching tasks. Further, on a second gold standard, the Limaye gold standard, comparable performances are achieved. The highest improvement compared to the direct matching is found for the property matching task with an increase of 10% in F-measure. For all three tasks, especially collecting negative support for incorrect correspondence during the indirect matching step is important to enhance the results. Altogether, T2K Match++ outperforms all state-of-the-art systems that have been evaluated either on tables from the T2D or Limaye dataset. Most systems show good results on one matching task but T2K Match++ is the only system that achieves F-measure scores above 0.8 for all tasks. Compared to results of the best performing system TableMiner+, the F-measure for the difficult property matching task is increased by 0.08, for the class matching task by 0.05, and for the instance matching task by 0.03.

Chapter 9

Conclusion

A promising source of data are web tables that have been employed in various use cases like fact search or knowledge base augmentation. For the knowledge base augmentation use case, the content of the web tables need to be understood. This can be achieved by matching web tables to a knowledge base. For the matching, three tasks need to be performed: instance, property and class matching. In this thesis, we systematically evaluated the utility of a wide range of different features for web table to knowledge base matching. Based on the results, the holistic T2K Match++ method has been developed which covers all three matching tasks. Further, we introduced a web table corpus together with its topical profile and a gold standard to facilitate research focusing on web tables. In this chapter, we summarize the key contributions, discuss known limitations, provide suggestions for future work directions, and finally depict the research impact of our work.

9.1 Summary

This thesis focuses on matching web tables to knowledge bases in order to generate correspondences for each matching task. To achieve high quality matching results, we systematically evaluated the utility of different features. Using these findings, the T2K Match++ method has been developed. Therefore, we first proposed an algorithm, T2K Match, which addresses the three matching tasks in an integrated fashion. To evaluate the performance of a matching system, we introduce the T2D gold standard which captures correspondences for all matching tasks between a set of tables and the knowledge base DBpedia. By applying T2K on the T2D gold standard, determined that relying only on features from the table is only partly sufficient. To improve the results, we included a wide range of features and analyzed their utility. Based on these findings, we extend T2K Match into T2K Match++ which exploits indirect matches to web tables about the same topic and uses knowledge derived from the knowledge base. T2K Match++ outperforms all state-of-the-art web table to knowledge base matching approaches on the T2D and Limaye gold standard.

To be able to work with web tables as data source, we created the first publicly available web table corpus, the WDC WTC 2012. In order to learn for which use cases web tables are most promising, we generated a topical profile of the web table corpus by matching it to DBpedia. Moreover, we analyzed potential of web tables for the use case of filling missing values in a knowledge base. In turn, the tables of the WDC WTC 2012 corpus together with the generated profile provided the basis for the T2D gold standard.

In the following, we discuss the contributions of this thesis.

9.1.1 Improving the Matching Evaluability and Transparency

We provide the publicly available gold standard T2D which has been generated to evaluate all tasks of matching web tables to knowledge bases. By now, most of the matching approaches for matching web tables to knowledge base have either been evaluated on topic-specific tables, a small amount of tables only, or on tables stemming from Wikipedia. Hence, topic-independent gold standards with a sufficient amount of tables are restricted regarding the choice of websites. Further, they do not include non-overlapping tables which does not follow the distribution of tables found on the Web. Only a small number of tables are relational and additionally overlap with a knowledge base. The T2D gold standard covers 779 tables including about 26 000 instance, 658 property, and 233 class correspondences to DBpedia. The distribution of the topics is balanced between tables describing for example works, agents and places. Thus, in contrast to other gold standard, e.g., introduced by [Limaye et al., 2010], a wider range of challenges is covered and the matching of web tables to knowledge bases is reflected more realistically.

By applying the algorithm T2K Match on T2D, various types of difficulties like the lack of evidence can be identified. Since state-of-the-art web table matching systems are evaluated on gold standards with a restricted range of challenges and their results are neither transparent nor verifiable, the influence and extent of the difficulties have not been known before. To find ways how to overcome these difficulties, we analyzed the utility of features that are used in state-of-the-art web table matching systems. Beside features found in the table itself, the set of features includes the web table context as well as external resources like a surface form catalog providing alternative names. In previous works, only a subset of the tasks and the features is considered so a conclusion of the usefulness of individual features cannot be drawn. We integrate all features into T2K Match and proof their utility on the T2D gold standard. For the instance matching task, the entity label as well as the popularity of the instances has the largest influence on the results, considering the values contribute less. Contrary, the values show the largest utility for the property matching task while the influence of the attribute label varies largely, depending on the label informativeness. The results of the class matching task depend on a variety of features: the majority of the instances, the frequency as well as the page title and URL. In general, features found within tables lead to more correct results than context features. Nevertheless, taking context features into account can improve the results but particular caution is necessary since context features may also add a lot of noise. External resources show to be useful as long as their content is closely related to the content of the web tables.

Since not only the T2D gold standard but also the code of the matching method is publicly available, the results are transparent, can be reproduced, and additional features can be included.

9.1.2 Improving the Matching Quality

T2K Match++ is a web-scale system for holistically matching web tables to knowledge bases resulting in a high matching quality with respect to correctness and completeness. It covers all three matching tasks - instance, property, and class matching - in an integrated fashion. Compared to previous works, we do not only focus on relation extraction between named entity columns but also consider datatype properties which requires to include literal values. One of the main challenges that are posed by web tables is the lack of information, e.g., due to the size of tables or the non-existence of headers. To overcome the hetereogeneities, the method exploits the context of a web table as well as knowledge gained by matching web tables with the same topic. In order to properly handle additional features and indirect mappings, we adapted all steps of the matching process. First, an aggregation which estimates the matcher reliability ensures that features we cannot rely on for an individual table do not have a strong influence on the matching result. Second, a classification that goes beyond threshold-based filtering by exploiting domain knowledge is able to determine which correspondences are likely to hold, even if incorrect information is added due to indirect mappings. We evaluated T2K Match++ on the T2D gold standard which covers a wider range of challenges and reflects the matching task more realistically than other gold standards. On this gold standard F-measure scores around 0.9 are accomplished for all tasks which is higher than other systems report.

Most of the systems proposed in literature neither include information that can be gathered from the context nor take into account that tables with similar topics are also matched to the knowledge base. Almost all systems have been evaluated on tables from the Limaye gold standard. This gold standard focuses on tables originating from Wikipedia which restricts the range of challenges and the representativity. To be able to compare the results, we applied T2K Match++ on the Limaye gold standard. All web table to knowledge base systems are outperformed by T2K Match++ on all three matching tasks. This includes systems that incorporate background knowledge like an isA database [Venetis et al., 2011], apply probabilistic models [Limaye et al., 2010, Mulwad et al., 2013] or use an iterative matching approach [Zhang, 2016]. Most systems show good results on one matching task but T2K Match++ is the only system that achieves F-measure scores above 0.8 for all tasks. Compared to results of the best performing system TableMiner+, the F-measure for the difficult property matching task is increased by 0.08, for the class and instance matching task by 0.05 and 0.03, respectively. However, it needs to be mentioned that the majority of systems uses different knowledge bases as input which somehow limits the comparability.

Overall, we presented the T2K Match++ method that achieves high matching performances on both - a commonly used dataset including tables from Wikipedia and a dataset covering a wider range of challenges and incorporating tables from more than one website.

9.1.3 Increasing the Data Availability and Applicability

Until 2009, only big search engines have been in possession of large web crawls and in turn have been able to extract structured data from the Web in large scales. With the incorporation of non-profit organizations like the Common Crawl, researchers outside the companies can use large publicly available web crawls. In this thesis, we presented the first publicly available web table corpus: WDC Web Table Corpus 2012. Based on the WDC extraction framework, we were able to extract large amounts of relational web tables. Further, we implemented each step of the relational web table extraction process. The process covers the classification into relational tables and the metadata recovery, including methods for entity label column or data type detection. The WDC Web Table Corpus comprises 147 million relational tables that have been extracted from 3.5 billion pages. In total, about 1.1% of all tables are relational which is in line with the results of other extractions, e.g., by [Cafarella et al., 2008a]. With on average 12.4 rows and 3.5 columns, the web tables show similar characteristics as reported for other corpora.

Since web table corpora have not been publicly available, one can only speculate which topics are covered by web tables. With the WDC Web Table Corpus, we are able to profile the tables. We provide a topical analysis of the web tables by matching them to DBpedia. Out of the relational tables only about 3% could be matched to DBpedia. Thus, the majority of tables do not overlap with DBpedia since they cover topics like products which are rarely represented in DBpedia. However, about 1 million tables actually overlap with DBpedia. Most commonly detected topics are persons, especially athletes, and works. With the topical analysis, it is possible to estimate for a use case if web tables are a suitable source. Based on tables for which correspondences have been generated, we analyze the potential for one use case, the knowledge base augmentation. Since knowledge bases are most useful when they are complete and correct as possible, finding facts in web tables that are not included in the knowledge base are important. To estimate the slot filling potential for DBpedia, we first have to group the facts and then to resolve inconsistencies. Grouping all facts with the same instance-property combination results in about 1 million groups which presents the maximal amount of new facts. Of them, about 75% are included in DBpedia while the remaining 25% are missing. By applying the knowledge-base trust fusion strategy that estimates the quality of the created facts by comparing them to the according facts in the knowledge base, an F-measure of 0.7 is determined. The same quality is assumed for the missing facts. Overall, the slot filling potential highly depends on the individual property. For example, the largest amount of missing facts can be found for release dates of works with 15 836 new facts.

In brief, with the first publicly available web table corpus, researchers are able to work with web tables. Based on the topical profiling, researchers get an impression for which use cases web tables are beneficial.

9.2 Limitations and Future Work

In the following, we discuss current limitations of the presented approaches and which directions of feature work help to overcome these limitations. Considering the whole data integration pipeline, starting from the extraction and ending with the fusion, each single component could be improved or restriction could be relaxed.

This already starts with the assumptions that are defined by the underlying entity-attribute table model. In this model, exactly one concept is mentioned per table, each attribute is said to be binary and describe an entity whose name is depicted in the entity label column. Tables that describe more than one concept, have compound entity label columns or n-ary attributes are not interpreted correctly [Adelfio and Samet, 2013, Lehmberg and Bizer, 2016, Oulabi et al., 2016]. Therefore, semantic normalization tries to detect and separate individual semantic concepts. For tables, [Braunschweig et al., 2015b] apply a semantic normalization that uses functional dependencies to identify concepts. All subsequent steps can then be performed on the normalized tables such that it is possible to correctly match them to the knowledge base.

The web table extraction, including the classification and metadata recovery, builds the foundation of the further process and hence has a crucial influence on the integration results. The classification of relational tables that has been applied to generate the WDC Web Table Corpus 2012 results in a precision of about 0.6 which is improvable. Further, all tables are excluded for which the entity label column detection is not able to recover an entity label column. One possible enhancement of the entity column detection is to use the knowledge base to get an impression if values of a column correspond to instance labels and the remaining attributes potentially fit to properties. Such an entity label column detection is for example performed by [Wang et al., 2012] using the knowledge base Probase.

Regarding the matching, if tables only consist of a few entities, they can either get filtered due to missing evidence or incorrect correspondences can be generated. Even by considering additional features like the context or similar tables, not all tables will get matched correctly. As recently shown by [Lehmberg and Bizer, 2017], stitching web tables from a website together improves the matching quality. Thus, stitched tables together before matching them can be beneficial especially to improve the matching recall and increase the amount of generated correspondences which in turn potentially increases the amount of missing facts that can be found for augmenting a knowledge base. However, the currently available gold standards rarely cover such tables. This results from the table selection strategy that uses seed entities to find tables that overlap with a knowledge base. Without a gold standard that explicitly includes small tables, the improvement in the matching quality using stitched tables is hardly measurable.

Another direction of future work is to introduce a feedback loop between matching and fusion. During the fusion, we compare the generated facts to the knowledge base and in turn can derive which correspondences are correct. Thus, the knowledge base can be used as distant-supervision for learning better matching rules and data fusion policies. A typical approach for matching ontologies is to let an expert verify correspondences and learn from the answers how to adapt the matching to improve the quality [Ritze et al., 2013]. By using the knowledge base to verify the generated facts, the feedback can be gathered automatically. However, the feedback loop can also involve the user especially if the knowledge base does not have enough coverage [Chu et al., 2015, Fan et al., 2014].

9.3 Research Impact

In this section, we describe the impact of our contributions for other researchers. First, the data including the generated web table corpora as well as the T2D gold standard as has already been reused by the research community. Second, our approaches are either applied or used as state-of-the-art reference to which other methods are compared to.

Data Reuse The WDC Web Table corpora are used as large scale data sources that provide real-world scenarios. Both, [Zhu et al., 2016] and [Rao and Zhu, 2016] apply their blocking strategy, local sensitive hashing, on the corpus to show that the approaches are able to decrease the tremendous amount of required comparisons. [Tschirschnitz et al., 2017] use the corpus to detect inclusion dependencies. By now, three table matching systems consider the T2D gold standard for their evaluation. [Efthymiou et al., 2016, Efthymiou et al., 2017] use it to compare the performance of different instance matching methods, the TAIPAN system [Ermilov and Ngomo, 2016] and the system by [Pham et al., 2016] are evaluated on T2D.

Method Reuse The four mentioned matching systems are not only evaluated on the T2D gold standard but also compare their results to the results of T2K Match which is referred to as state-of-the-art system for matching web tables to knowledge bases. Further, parts of the algorithm have been integrated in the data science platform RapidMiner[1] to enable a table search [Gentile et al., 2016]. Further, T2K Match has been adapted to match web tables to the Yahoo knowledge graph and to Wikidata [Oulabi and Bizer, 2017]. Besides the matching, also the methods within the web table extraction have been reused. While [Eberius et al., 2015] build on our extraction framework to create the Dresden Web Table Corpus consisting of 125 million relational tables, [Saleiro et al., 2017] integrate our entity label column detection into their system to generate a set of 600 queries from Wikipedia tables that is used to evaluate a table search algorithm.

In summary, web tables have become a source that is gaining more and more attention. Companies like Google already provide yet an experimental interface that answers queries by showing tables that include the according information. With publicly available web table corpora, we enable the usage of web tables for the entire research community. A gold standard covering tables from more than one source helps the researchers to find shortcomings of their approaches and provide an ability to easily compare the results. By presenting the feature utility study, we highlighted which features are crucial to achieve acceptable matching results and which external sources are beneficial to consider. T2K Match and its successor T2K Match++ provide methods that are already referred to as state-of-the-art such that newly introduced approaches try to outperform them which in turn advances the research. Finally, we believe that our findings together with the discussed limitations lay a foundation for future research.

[1] `https://rapidminer.com/`

List of Figures

List of Tables

Listings

Bibliography

[Abadi et al., 2014] Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P. A., Carey, M. J., Chaudhuri, S., Dean, J., Doan, A., Franklin, M. J., Gehrke, J., Haas, L. M., Halevy, A. Y., Hellerstein, J. M., Ioannidis, Y. E., Jagadish, H. V., Kossmann, D., Madden, S., Mehrotra, S., Milo, T., Naughton, J. F., Ramakrishnan, R., Markl, V., Olston, C., Ooi, B. C., Ré, C., Suciu, D., Stonebraker, M., Walter, T., and Widom, J. (2014). The beckman report on database research. *SIGMOD Rec.*, 43(3):61–70.

[Abiteboul et al., 1995] Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[Adelfio and Samet, 2013] Adelfio, M. D. and Samet, H. (2013). Schema Extraction for Tabular Data on the Web. *Proc. of the VLDB Endow.*, 6:421–432.

[Aggrawal, 2015] Aggrawal, A. (2015). *Managing big data integration in the public sector*. Information Science Reference - Imprint of: IGI Publishing.

[Aprosio et al., 2013] Aprosio, A. P., Giuliano, C., and Lavelli, A. (2013). Extending the coverage of dbpedia properties using distant supervision over wikipedia. In *Proc. of the 2013th Int. Conference on NLP-DBpedia*, pages 20–31, Aachen, Germany, Germany. CEUR-WS.org.

[Assaf et al., 2015] Assaf, A., Troncy, R., and Senart, A. (2015). *Roomba: An Extensible Framework to Validate and Build Dataset Profiles*, pages 325–339. Springer International Publishing, Cham.

[Aumueller et al., 2005] Aumueller, D., Do, H.-H., Massmann, S., and Rahm, E. (2005). Schema and ontology matching with coma++. In *Proc. of the 2005 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '05, pages 906–908, New York, NY, USA. ACM.

[Bailey et al., 2006] Bailey, P., Hawking, D., and Krumpholz, A. (2006). Toward meaningful test collections for information integration benchmarking. In *Proc. of the 15th Int. Conference on the World Wide Web : Workshop on Information Integration on the Web*, IIWeb '06.

[Balakrishnan et al., 2015] Balakrishnan, S., Halevy, A., Harb, B., Lee, H., Madhavan, J., Rostamizadeh, A., Shen, W., Wilder, K., Wu, F., and Yu, C. (2015). Applying webtables in practice. In *Proc. of the 7th Biennial Conference on Innovative Data Systems Research*, CIDR '15.

[Barbosa et al., 2014] Barbosa, L., Kien, P., Silva, C., Vieira, M., and Freire, J. (2014). Structured open urban data: Understanding the landscape. *Big Data*, 2(3).

[Bellahsene et al., 2011] Bellahsene, Z., Bonifati, A., and Rahm, E. (2011). *Schema Matching and Mapping*. Springer.

[Berners-Lee, 2006] Berners-Lee, T. (2006). Linked data.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, (5):28–37.

[Bhagavatula et al., 2013] Bhagavatula, C. S., Noraset, T., and Downey, D. (2013). Methods for exploring and mining tables on wikipedia. In *Proc. of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, pages 18–26, New York, NY, USA. ACM.

[Bhagavatula et al., 2015] Bhagavatula, C. S., Noraset, T., and Downey, D. (2015). *TabEL: Entity Linking in Web Tables*, pages 425–441. Springer International Publishing, Cham.

[Bilenko and Mooney, 2003] Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proc. of the 9th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 39–48, New York, NY, USA. ACM.

[Bilke, 2006] Bilke, A. (2006). *Duplicate-based Schema Matching*. PhD thesis.

[Bilke and Naumann, 2005] Bilke, A. and Naumann, F. (2005). Schema matching using duplicates. In *Proc. of the 21st Int. Conference on Data Engineering*, pages 69–80, Washington, DC, USA. IEEE Computer Society.

[Bizer, 2014] Bizer, C. (2014). Search Joins with the Web. In *Proc. of the 17th Int. Conference on Database Theory*, ICDT '14, page 3.

[Bizer et al., 2009a] Bizer, C., Heath, T., and Berners-Lee, T. (2009a). Linked Data - The Story So Far. *Int. Journal on Semantic Web and Information Systems*, 5:1–22.

[Bizer et al., 2009b] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009b). Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165. The Web of Data.

[Bleiholder and Naumann, 2009] Bleiholder, J. and Naumann, F. (2009). Data fusion. *ACM Comput. Surv.*, 41(1):1—41.

[Bollacker et al., 2008] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proc. of the 2008 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

[Braunschweig et al., 2015a] Braunschweig, K., Thiele, M., Eberius, J., and Lehner, W. (2015a). Column-specific Context Extraction for Web Tables. In *Proc. of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 1072–1077.

[Braunschweig et al., 2015b] Braunschweig, K., Thiele, M., and Lehner, W. (2015b). *From Web Tables to Concepts: A Semantic Normalization Approach*, pages 247–260. Springer International Publishing, Cham.

[Bryl et al., 2015] Bryl, V., Bizer, C., and Paulheim, H. (2015). Gathering alternative surface forms for dbpedia entities. In *Proc. of the 3rd NLP & DBpedia Workshop*, pages 13–24.

[Buche et al., 2013] Buche, P., Dibie-Barthelemy, J., Ibanescu, L., and Soler, L. (2013). Fuzzy web data tables integration guided by an ontological and terminological resource. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):805–819.

[Bühmann and Lehmann, 2013] Bühmann, L. and Lehmann, J. (2013). Pattern based knowledge base enrichment. In Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J. X., Aroyo, L., Noy, N., Welty, C., and Janowicz, K., editors, *Proc. of the 12th International Semantic Web Conference*, ISWC '13, pages 33–48, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Busse et al., 1999] Busse, S., Kutsche, R.-D., Leser, U., and Weber, H. (1999). Federated information systems : concepts, terminology, and architectures. Technical report, TU Berlin.

[Cafarella et al., 2008a] Cafarella, M., Halevy, Alonand Zhang, Y., Wang, D. Z., and Wu, E. (2008a). Uncovering the Relational Web. In *Proc. of the 11th Int. Workshop on the Web and Databases*, WebDB '08.

[Cafarella et al., 2009] Cafarella, M. J., Halevy, A., and Khoussainova, N. (2009). Data Integration for the Relational Web. *Proc. of the VLDB Endow.*, 2:1090–1101.

[Cafarella et al., 2008b] Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., and Zhang, Y. (2008b). WebTables: Exploring the Power of Tables on the Web. *Proc. of the VLDB Endow.*, 1:538–549.

[Cambazoglu and Baeza-Yates, 2015] Cambazoglu, B. B. and Baeza-Yates, R. (2015). Scalability challenges in web search engines. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7:1–138.

[Carlson et al., 2010] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, Jr., E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1306–1313. AAAI Press.

[Chakrabarti et al., 2002] Chakrabarti, S., Joshi, M. M., Punera, K., and Pennock, D. M. (2002). The structure of broad topics on the web. In *Proc. of the 11th Int. Conference on World Wide Web*, WWW '02, pages 251–262, New York, NY, USA. ACM.

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58.

[Chen et al., 2000] Chen, H.-H., Tsai, S.-C., and Tsai, J.-H. (2000). Mining tables from large scale html texts. In *Proc. of the 18th Conference on Computational Linguistics*, COLING '00, pages 166–172, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Chen et al., 2009] Chen, Z., Kalashnikov, D. V., and Mehrotra, S. (2009). Exploiting context analysis for combining multiple entity resolution systems. In *Proc. of the 2009 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '09, pages 207–218, New York, NY, USA. ACM.

[Christen, 2008] Christen, P. (2008). Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *Proc. of the 14th Int. Conference on Knowledge Discovery and Data Mining*, KDD '08.

[Christen, 2012] Christen, P. (2012). *Data Matching*. Springer.

[Christophides et al., 2015] Christophides, V., Efthymiou, V., and Stefanidis, K. (2015). *Entity Resolution in the Web of Data*. Morgan & Claypool.

[Chu et al., 2015] Chu, X., Morcos, J., Ilyas, I. F., Ouzzani, M., Papotti, P., Tang, N., and Ye, Y. (2015). Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proc. of the 2015 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '15, pages 1247–1261, New York, NY, USA. ACM.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P., and Fienberg, S. (2003). A comparison of string distance metrics for name-maching tasks. In *Proc. of the 8th IJCAI Workshop on Information Integration*, pages 73–78.

[Crestan and Pantel, 2011] Crestan, E. and Pantel, P. (2011). Web-scale Table Census and Classification. In *Proc. of the 4th Int. Conf. on Web Search and Data Mining*, WSDM '11, pages 545–554. ACM.

[Cruz et al., 2009] Cruz, I. F., Antonelli, F. P., and Stroe, C. (2009). Efficient selection of mappings and automatic quality-driven combination of matching methods. In *Proc. of the 4th Int. Workshop on Ontology Matching*, OM '09.

[Daiber et al., 2013] Daiber, J., Jakob, M., Hokamp, C., and Mendes, P. N. (2013). Improving efficiency and accuracy in multilingual entity extraction. In *Proc. of the 9th Int. Conference on Semantic Systems*, I-Semantics '13.

[Das Sarma et al., 2012] Das Sarma, A., Fang, L., Gupta, N., Halevy, A., Lee, H., Wu, F., Xin, R., and Yu, C. (2012). Finding Related Tables. In *Proc. of the 2012 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '12.

[Dhamankar et al., 2004] Dhamankar, R., Lee, Y., Doan, A., Halevy, A., and Domingos, P. (2004). imap: Discovering complex semantic matches between database schemas. In *Proc. of the 2004 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '04, pages 383–394, New York, NY, USA. ACM.

[Dhenakaran and Sambanthan, 2011] Dhenakaran, S. and Sambanthan, K. T. (2011). Web crawler - an overview. *International Journal of Computer Science and Communication*, 2(1).

[Do and Rahm, 2002] Do, H.-H. and Rahm, E. (2002). COMA: A System for Flexible Combination of Schema Matching Approaches. In *Proc. of the 28th Int. Conference on Very Large Data Bases*, VLDB '02, pages 610–621.

[Doan et al., 2001] Doan, A., Domingos, P., and Halevy, A. Y. (2001). Reconciling schemas of disparate data sources: A machine-learning approach. *SIGMOD Rec.*, 30(2):509–520.

[Doan et al., 2012] Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of Data Integration*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

[Doan et al., 2004a] Doan, A., Madhavan, J., Domingos, P., and Halevy, A. (2004a). *Ontology Matching: A Machine Learning Approach*, pages 385–403. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Doan et al., 2004b] Doan, A., Noy, N., and Halevy, A. (2004b). Introduction to the Special Issue on Semantic Integration. *SIGMOD Rec.*, 33(4).

[Dong et al., 2014] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *Proc. of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610.

[Dong et al., 2015] Dong, X. L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., and Zhang, W. (2015). Knowledge-based Trust: Estimating the Trustworthiness of Web Sources. *Proc. of the VLDB Endow.*, 8(9):938–949.

[Dong and Srivastava, 2015] Dong, X. L. and Srivastava, D. (2015). *Big Data Integration*. Morgen & Claypool.

[Dragisic et al., 2014] Dragisic, Z., Eckert, K., Euzenat, J., Faria, D., Ferrara, A., Granada, R., Ivanova, V., Jiménez-Ruiz, E., Kempf, A. O., Lambrix, P., Montanelli, S., Paulheim, H., Ritze, D., Shvaiko, P., Solimando, A., Trojahn, C., Zamazal, O., and Grau, B. C. (2014). Results of the ontology alignment evaluation initiative 2014. In *Proc. of the 9th Int. Workshop on Ontology Matching*, OM'14, pages 61–104, Aachen, Germany, Germany. CEUR-WS.org.

[Duchateau and Bellahsene, 2014] Duchateau, F. and Bellahsene, Z. (2014). Designing a benchmark for the assessment of schema matching tools. *Open Journal of Databases*, 1(1):3–25.

[Dutta et al., 2015] Dutta, A., Meilicke, C., and Stuckenschmidt, H. (2015). Enriching structured knowledge with open information. In *Proc. of the 24th Int. Conference on World Wide Web*, WWW '15, pages 267–277, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Eberius et al., 2015] Eberius, J., Braunschweig, K., Hentsch, M., Thiele, M., Ahmadov, A., and Lehner, W. (2015). Building the Dresden Web Table Corpus: A Classification Approach. In *Proc.of the 2nd Int. Symposium on Big Data Computing*, BDC '15.

[Efthymiou et al., 2017] Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., and Christophides, V. (2017). Matching web tables with knowledge base entities: From entity lookups to entity embeddings. In *Proc. of the 16th Int. Semantic Web Conference*, ISWC '17.

[Efthymiou et al., 2016] Efthymiou, V., Hassanzadeh, O., Sadoghi, M., and Rodriguez-Muro, M. (2016). Annotating web tables through ontology matching. In *Proc. of the 15th Int. Workshop on Ontology Matching*, OM '16.

[Ehrig and Sure, 2004] Ehrig, M. and Sure, Y. (2004). *Ontology Mapping – An Integrated Approach*, pages 76–91. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Elfeky et al., 2002] Elfeky, M. G., Verykios, V. S., and Elmagarmid, A. K. (2002). Tailor: a record linkage toolbox. In *Proc. 18th Int. Conference on Data Engineering*, pages 17–28.

[Elmagarmid et al., 2007] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16.

[Embley et al., 2006] Embley, D. W., Hurst, M., Lopresti, D., and Nagy, G. (2006). Table-processing paradigms: a research survey. *International Journal of Document Analysis and Recognition*, 8(2):66–86.

[Ermilov and Ngomo, 2016] Ermilov, I. and Ngomo, A.-C. N. (2016). *TAIPAN: Automatic Property Mapping for Tabular Data*, pages 163–179. Springer International Publishing, Cham.

[Euzenat and Shvaiko, 2007] Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer.

[Fader et al., 2011] Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *Proc. of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Fan et al., 2014] Fan, J., Meiyu, L., Beng Chin, O., Wang-Chiew, T., and Zhang, M. (2014). A Hybrid Machine-Crowdsourcing System for Matching Web Tables. In *Proc. of the 30th IEEE Int. Conference on Data Engineering*, pages 976–987.

[Färber et al., 2016] Färber, M., Ell, B., Menne, C., and Rettinger, A. (2016). A Comparative Survey of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web*, 1-5.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

[Fellegi and Sunter, 1969] Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210.

[Fleischhacker et al., 2013] Fleischhacker, D., Paulheim, H., Bryl, V., Vker, J., and Bizer, C. (2013). Detecting Errors in Numerical Linked Data using Cross-Checked Outlier Detection . In *Proc. of the 13th Int. Semantic Web Conference*, ISWC '13.

[Fossati et al., 2017] Fossati, M., Dorigatti, E., and c, C. G. (2017). N-ary Relation Extraction for Joint T-Box and A-Box Knowledge Base Augmentation. *Semantic Web Journal*, pages 1–27.

[Gal et al., 2016] Gal, A., Roitman, H., and Sagi, T. (2016). From Diversity-based Prediction to Better Ontology & Schema Matching. In *Proc. of the 25th Int. Conference on World Wide Web*, WWW '16.

[Galkin et al., 2015] Galkin, M., Mouromtsev, D., and Auer, S. (2015). Identifying Web Tables: Supporting a Neglected Type of Content on the Web. In Klinov, P. and Mouromtsev, D., editors, *Proc. of the 6th Int. Conference on Knowledge Engineering and Semantic Web*, pages 48–62, Cham. Springer International Publishing.

[Gatterbauer et al., 2007] Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., and Pollak, B. (2007). Towards Domain-independent Information Extraction from Web Tables. In *Proc. of the 16th Int. Conference on World Wide Web*, WWW '07, pages 71–80, New York, NY, USA. ACM.

[Gentile et al., 2016] Gentile, A. L., Kirstein, S., Paulheim, H., and Bizer, C. (2016). *Extending RapidMiner with Data Search and Integration Capabilities*, pages 167–171. Springer International Publishing, Cham.

[Göbel et al., 2012] Göbel, M., Hassan, T., Oro, E., and Orsi, G. (2012). A Methodology for Evaluating Algorithms for Table Understanding in PDF Documents. In *Proc. of the 2012 ACM Symposium on Document Engineering*, DocEng '12, pages 45–48, New York, NY, USA. ACM.

[Halevy et al., 2006] Halevy, A., Rajaraman, A., and Ordille, J. (2006). Data integration: The teenage years. In *Proc. of the 32nd Int. Conference on Very Large Data Bases*, VLDB '06, pages 9–16. VLDB Endowment.

[Halevy et al., 2005] Halevy, A. Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., and Sikka, V. (2005). Enterprise Information Integration: Successes, Challenges and Controversies. In *Proc. of the 2005 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '05, pages 778–787, New York, NY, USA. ACM.

[Hartman and Ackermann, 2010] Hartman, K. and Ackermann, E. (2010). *Searching and Researching on the Internet and the World Wide Web*. Franklin, Beedle & Associates Inc.

[Hartung et al., 2013] Hartung, M., Gro, A., and Rahm, E. (2013). Composition Methods for Link Discovery. In *Proc. of 15th GI-Fachtagung fr Datenbanksysteme in Business, Technologie und Web*.

[Hassanzadeh et al., 2009] Hassanzadeh, O., Chiang, F., Lee, H. C., and Miller, R. J. (2009). Framework for Evaluating Clustering Algorithms in Duplicate Detection. *Proc. of the VLDB Endow.*, 2(1):1282–1293.

[Hassanzadeh et al., 2015] Hassanzadeh, O., Ward, M. J., Rodriguez-Muro, M., and Srinivas, K. (2015). Understanding a large corpus of web tables through matching with knowledge bases: an empirical study. In *Proc. of the 10th Int. Workshop on Ontology Matching*, OM '15.

[He and Chang, 2004] He, B. and Chang, K. C.-C. (2004). A Holistic Paradigm for Large Scale Schema Matching. *SIGMOD Rec.*, 33:20–25.

[Hernández and Stolfo, 1995] Hernández, M. A. and Stolfo, S. J. (1995). The Merge/Purge Problem for Large Databases. In *Proc. of the 1995 ACM SIGMOD Int. Conference on Management of Data*, pages 127–138, New York, NY, USA. ACM.

[Hernández and Stolfo, 1998] Hernández, M. A. and Stolfo, S. J. (1998). Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Mining and Knowledge Discovery*, 2(1):9–37.

[Herschel and Naumann, 2010] Herschel, M. and Naumann, F. (2010). *An Introduction to Duplicate Detection*. Morgan & Claypool.

[Hignette et al., 2007] Hignette, G., Buche, P., Dibie-Barthélemy, J., and Haemmerlé, O. (2007). An Ontology-Driven Annotation of Data Tables. In Weske, M., Hacid, M.-S., and Godart, C., editors, *Proc. of the 8th Int. Conference on Web Information Systems Engineering*, WISE '07, pages 29–40, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Hurst, 1999] Hurst, M. (1999). Layout and language: Beyond simple text for information interaction - modelling the table. In *Proc. of the 2nd Int. Conference on Multimodel Interfaces*.

[Hurst, 2001] Hurst, M. (2001). Layout and Language: Challenges for Table Understanding on the Web. In *Proc. of the Int. Workshop on Web Document Analysis*, pages 27–30.

[Isele and Bizer, 2012] Isele, R. and Bizer, C. (2012). Learning Expressive Linkage Rules Using Genetic Programming. *Proc. of the VLDB Endow.*, 5(11):1638–1649.

[Jaccard, 2006] Jaccard, P. (2006). The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50.

[Jiménez-Ruiz and Cuenca Grau, 2011] Jiménez-Ruiz, E. and Cuenca Grau, B. (2011). LogMap: Logic-Based and Scalable Ontology Matching. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., and Blomqvist, E., editors, *Proc. of the 10th Int. Semantic Web Conference*, ISWC '11, pages 273–288, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Jung and Kwon, 2006] Jung, S.-W. and Kwon, H.-C. (2006). A scalable hybrid approach for extracting head components from web tables. *IEEE Transactions on Knowledge and Data Engineering*, 18(2):174–187.

[Kim and Lee, 2005] Kim, Y.-S. and Lee, K.-H. (2005). Detecting tables in Web documents. *Engineering Applications of Artificial Intelligence*, 18(6):745 – 757.

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax - W3C Recommendation.

[Kobilarov et al., 2009] Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., and Lee, R. (2009). Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections. In Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., and Simperl, E., editors, *Prof. of the 6th European Semantic Web Conference*, ESWC '09, pages 723–737, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Köpcke and Rahm, 2010] Köpcke, H. and Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197 – 210.

[Lambrix and Tan, 2005] Lambrix, P. and Tan, H. (2005). *A Framework for Aligning Ontologies*, pages 17–31. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Lautert et al., 2013] Lautert, L. R., Scheidt, M. M., and Dorneles, C. F. (2013). Web Table Taxonomy and Formalization. *SIGMOD Rec.*, 42(3):28–33.

[Lehmann et al., 2015] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, 6(2):167–195.

[Lehmberg and Bizer, 2016] Lehmberg, O. and Bizer, C. (2016). Web Table Column Categorisation and Profiling. In *Proc. of the 19th Int. Workshop on Web and Databases*, WebDB '16. ACM.

[Lehmberg and Bizer, 2017] Lehmberg, O. and Bizer, C. (2017). Stitching Web Tables for Improving Matching Quality. *Proc. of the VLDB Endow.*

[Lehmberg et al., 2016] Lehmberg, O., Ritze, D., Meusel, R., and Bizer, C. (2016). A Large Public Corpus of Web Tables containing Time and Context Metadata. In *Proc. of the 25th Int. Conference on World Wide Web*, WWW '16.

[Lehmberg et al., 2015] Lehmberg, O., Ritze, D., Ristoski, P., Meusel, R., Paulheim, H., and Bizer, C. (2015). The Mannheim Search Join Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:159–166.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

[Li et al., 2009] Li, J., Tang, J., Li, Y., and Luo, Q. (2009). RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Knowledge and Data Engineering*, 21(8):1218–1232.

[Li and Clifton, 2000] Li, W.-S. and Clifton, C. (2000). SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49 – 84.

[Limaye et al., 2010] Limaye, G., Sarawagi, S., and Chakrabarti, S. (2010). Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc.of the VLDB Endow.*, 3:1338–1347.

[Lindahl, 2012] Lindahl, G. (2012). Blekko donates search data to common crawl.

[Ling et al., 2013] Ling, X., Halevy, A., Wu, F., and Yu, C. (2013). Synthesizing union tables from the web. In *Proc. of the 23rd Int. Joint Conference on Artificial Intelligence*, pages 2677–2683.

[Lu et al., 2013] Lu, C., Bing, L., Lam, W., Chan, K., and Gu, Y. (2013). Web entity detection for semi-structured text data records with unlabeled data. *International Journal of Computational Linguistics and Applications*, 4(2):135–150.

[Madhavan et al., 2005] Madhavan, J., Bernstein, P. A., Doan, A., and Halevy, A. (2005). Corpus-Based Schema Matching. In *Proc. of the. 21st Int. Conference on Data Engineering*, pages 57–68.

[Madhavan et al., 2001] Madhavan, J., Bernstein, P. A., and Rahm, E. (2001). Generic Schema Matching with Cupid. In *Proc. of the VLDB Endow.*, pages 49–58.

[Mahdisoltani et al., 2015] Mahdisoltani, F., Biega, J., and Suchanek, F. M. (2015). YAGO3: A Knowledge Base from Multilingual Wikipedias. In *Proc. of the 7th Biennial Conference on Innovative Data Systems Research*, CIDR '15.

[Mannino et al., 1988] Mannino, M. V., Chu, P., and Sager, T. (1988). Statistical Profile Estimation in Database Systems. *ACM Comput. Surv.*, 20(3):191–221.

[McCallum et al., 2000] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163.

[Meilicke et al., 2007] Meilicke, C., Stuckenschmidt, H., and Tamilin, A. (2007). Repairing Ontology Mappings. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence*.

[Meusel, 2017] Meusel, R. (2017). *Web-scale profiling of semantic annotations in HTML pages*. PhD thesis, Mannheim, Germany.

[Meusel et al., 2016] Meusel, R., Ritze, D., and Paulheim, H. (2016). Towards More Accurate Statistical Profiling of Deployed schema.org Microdata. *Special Issue on Web Data Quality of the ACM Journal on Data and Information Quality*, 8(1):1–31.

[Mühleisen and Bizer, 2012] Mühleisen, H. and Bizer, C. (2012). Web Data Commons - Extracting Structured Data from Two Large Web Corpora. In *Proc. of the 5th WWW Workshop Linked Data on the Web*, LDOW '12.

[Mulwad et al., 2013] Mulwad, V., Finin, T., and Joshi, A. (2013). Semantic message passing for generating linked data from tables. In *Proc. of the 12th Int. Semantic Web Conference*, ISWC '13.

[Mulwad et al., 2010a] Mulwad, V., Finin, T., Syed, Z., and Joshi, A. (2010a). T2LD: Interpreting and Representing Tables as Linked Data . In *Proc. of the 9th Int. Semantic Web Conference*, ISWC '10.

[Mulwad et al., 2010b] Mulwad, V., Finin, T., Syed, Z., and Joshi, A. (2010b). Using linked data to interpret tables. In *Proc. of the 1st Int. Workshop on Consuming Linked Data*, COLD '10.

[Muñoz et al., 2013] Muñoz, E., Hogan, A., and Mileo, A. (2013). Triplifying Wikipedia's Tables. In *Proc. of the 1st Int. Conference on Linked Data for Information Extraction*, LD4IE '13.

[Muñoz et al., 2014] Muñoz, E., Hogan, A., and Mileo, A. (2014). Using Linked Data to Mine RDF from Wikipedia's Tables. In *Proc. of the 7th ACM Int. Conference on Web Search and Data Mining*, WSDM '14, pages 533–542, New York, NY, USA. ACM.

[Naumann, 2014] Naumann, F. (2014). Data profiling revisited. *ACM SIGMOD Record*, 42(4):40–49.

[Navathe et al., 1986] Navathe, S., Elmasri, R., and Larson, J. (1986). Integrating User Views in Database Design. *Computer*, 19(1):50–62.

[Neumaier et al., 2017] Neumaier, S., Polleres, A., Steyskal, S., and Umbrich, J. (2017). Data Integration for Open Data on the Web. In *Reasoning Web. Semantic Interoperability on the Web*. Springer, Lodon, United Kingdom. to appear.

[Ngomo and Auer, 2011] Ngomo, A.-C. N. and Auer, S. (2011). LIMES: A Time-efficient Approach for Large-scale Link Discovery on the Web of Data. In *Proc. of the 22nd Int. Joint Conference on Artificial Intelligence*, IJCAI'11, pages 2312–2317. AAAI Press.

[Ngomo et al., 2011] Ngomo, A.-C. N., Lehmann, J., Auer, S., and Höffner, K. (2011). RAVEN - Active Learning of Link Specifications. In *Proc of the 6th Int. Workshop on Ontology Matching*, OM '11, pages 25–36, Aachen, Germany, Germany. CEUR-WS.org.

[Noy and Musen, 2000] Noy, N. F. and Musen, M. A. (2000). Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. of the 15th AAAI Conference on Artificial Intelligence*.

[Oulabi and Bizer, 2017] Oulabi, Y. and Bizer, C. (2017). Estimating Missing Temporal Meta-Information using Knowledge-Based-Trust. In *Proc. of the 3rd Int. Workshop on Knowledge Discovery on the WEB*, KDWeb '17.

[Oulabi et al., 2016] Oulabi, Y., Meusel, R., and Bizer, C. (2016). Fusing time-dependent web table data. In *Proc. of the 19th Int. Workshop on Web and Databases*, WebDB '16, pages 3:1–3:7, New York, NY, USA. ACM.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: bringing order to the Web. Technical report, Stanford InfoLab.

[Pasternack and Roth, 2010] Pasternack, J. and Roth, D. (2010). Knowing What to Believe (when You Already Know Something). In *Proc. of the 23rd Int. Conference on Computational Linguistics*, pages 877–885.

[Paulheim, 2014] Paulheim, H. (2014). Identifying Wrong Links between Datasets by Multi-dimensional Outlier Detection. In *Proc. of the 3rd Int. Workshop on Debugging Ontologies and Ontology Mappings*, volume 1162, pages 27–38, Aachen. RWTH. Online Ressource.

[Paulheim and Ponzetto, 2013] Paulheim, H. and Ponzetto, S. (2013). Extending DBpedia with Wikipedia List Pages. In *Proc.of 1st Int. Workshop on NLP & DBpedia*.

[Pearson, 1895] Pearson, K. (1895). Notes on regression and inheritance in the case of two parents. *Proc. of the Royal Society of London*, 58:240–242.

[Penn et al., 2001] Penn, G., Hu, J., Luo, H., and McDonald, R. (2001). Flexible Web document analysis for delivery to narrow-bandwidth devices. In *Proc. of the 6th Int. Conference on Document Analysis and Recognition*, pages 1074–1078.

[Petrovski et al., 2014] Petrovski, P., Bryl, V., and Bizer, C. (2014). Learning Regular Expressions for the Extraction of Product Attributes from E-commerce Microdata. In *Proc. of the 2nd Int. Conference on Linked Data for Information Extraction*, LD4IE '14, pages 45–54, Aachen, Germany, Germany. CEUR-WS.org.

[Pham et al., 2016] Pham, M., Alse, S., Knoblock, C. A., and Szekely, P. (2016). Semantic Labeling: A Domain-Independent Approach. In *Proc. of the 15th Int. Semantic Web Conference*, ISWC '16, pages 446–462. Springer.

[Pimplikar and Sarawagi, 2012] Pimplikar, R. and Sarawagi, S. (2012). Answering Table Queries on the Web Using Column Keywords. *Proc.of the VLDB Endow.*, 5(10):908–919.

[Pinto et al., 2002] Pinto, D., Branstein, M., Coleman, R., Croft, W. B., King, M., Li, W., and Wei, X. (2002). QuASM: A System for Question Answering Using Semi-structured Data. In *Proc. of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, pages 46–55, New York, NY, USA. ACM.

[Pinto et al., 2003] Pinto, D., McCallum, A., Wei, X., and Croft, W. B. (2003). Table Extraction Using Conditional Random Fields. In *Proc. of the 26th Annual Int. ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 235–242, New York, NY, USA. ACM.

[Qian, 2013] Qian, R. (2013). Understand Your World with Bing.

[Quercini and Reynaud-Delaître, 2013] Quercini, G. and Reynaud-Delaître, C. (2013). Entity Discovery and Annotation in Tables. In *Proc. of the 16th Int. Conference on Extending Database Technology*, EDBT '13, Genoa, Italy.

[Rahm, 2016] Rahm, E. (2016). The Case for Holistic Data Integration. In *Proc. of the 20th Advances in Databases and Information Systems Conference*, pages 11–27. Springer.

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

[Rahm and Do, 2000] Rahm, E. and Do, H. H. (2000). Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13.

[Rao and Zhu, 2016] Rao, B. and Zhu, E. (2016). Searching Web Data Using MinHash LSH. In *Proc. of the 2016 Int. Conference on Management of Data*, SIGMOD '16, pages 2257–2258, New York, NY, USA. ACM.

[Rhoades, 1993] Rhoades, S. (1993). The Herfindahl-Herschman Index. *Federal Reserve Bulletin*, 79:188–189.

[Rigden et al., 2016] Rigden, D., Fernandez-Suarez, X., and Galperin, M. (2016). The 2016 database issue of Nucleic Acids Research and an updated molecular biology database collection. *Nucleic Acids Research*, 44:D1 – D6.

[Rinser et al., 2013] Rinser, D., Lange, D., and Naumann, F. (2013). Cross-Lingual Entity Matching and Infobox Alignment in Wikipedia. *Information Systems*, 38:887–907.

[Ritze and Bizer, 2017] Ritze, D. and Bizer, C. (2017). Matching Web Tables To DBpedia - A Feature Utility Study. In *Proc. of the 20th Extended Database Technology Conference*, EDBT '17.

[Ritze and Bizer, 2018] Ritze, D. and Bizer, C. (2018). T2K Match++: Restricting Holistic Matching with Domain Knowledge. In *Paper submitted at the 11th ACM Int. Conference on Web Search and Data Mining*, WSDM '18.

[Ritze et al., 2015] Ritze, D., Lehmberg, O., and Bizer, C. (2015). Matching HTML Tables to DBpedia. In *Proc. of the 5th Int. Conference on Web Intelligence, Mining and Semantics*, WIMS '15.

[Ritze et al., 2016] Ritze, D., Lehmberg, O., Oulabi, Y., and Bizer, C. (2016). Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In *Proc. of the 25th Int. Conference on World Wide Web*, WWW '16.

[Ritze and Paulheim, 2011] Ritze, D. and Paulheim, H. (2011). Towards an Automatic Parameterization of Ontology Matching Tools Based on Example Mappings. In *Proc. of the 6th Int. Workshop on Ontology Matching*, OM '11, pages 37–48, Aachen, Germany, Germany. CEUR-WS.org.

[Ritze et al., 2013] Ritze, D., Paulheim, H., and Eckert, K. (2013). Evaluation Measures for Ontology Matchers in Supervised Matching Scenarios. In Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J. X., Aroyo, L., Noy, N., Welty, C., and Janowicz, K., editors, *Proc. of the 12th Int. Semantic Web Conference*, ISWC '13, pages 392–407, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Ritze et al., 2010] Ritze, D., Völker, J., Meilicke, C., and Šváb Zamazal, O. (2010). Linguistic Analysis for Complex Ontology Matching. In *Proc. of the 5th Int. Ontology Matching Workshop*, OM '10, pages 1–12, Aachen, Germany, Germany. CEUR-WS.org.

[Sagi and Gal, 2013] Sagi, T. and Gal, A. (2013). Schema matching prediction with applications to data source discovery and dynamic ensembling. *VLDB Journal*, 22:689–710.

[Saleiro et al., 2017] Saleiro, P., Milic-Frayling, N., Rodrigues, E. M., and Soares, C. (2017). RELink: A Research Framework and Test Collection for Entity-Relationship Retrieval. In *Proc. of the 40th Int. ACM SIGIR Conference*, page to be publsihed.

[Salton and McGill, 1983] Salton, G. and McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill.

[Sandhaus, 2010] Sandhaus, E. (2010). Semantic Technology at The New York Times: Lessons Learned and Future Directions. In Patel-Schneider, P. F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J. Z., Horrocks, I., and Glimm, B., editors, *Proc. of the 9th Int. Semantic Web Conference*, ISWC '10, pages 355–355, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Sarawagi and Chakrabarti, 2014] Sarawagi, S. and Chakrabarti, S. (2014). Open-domain Quantity Queries on Web Tables: Annotation, Response, and Consensus Models. In *Proc. of the 20th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 711–720, New York, NY, USA. ACM.

[Schilder and Habel, 2001] Schilder, F. and Habel, C. (2001). From Temporal Expressions to Temporal Information: Semantic Tagging of News Messages. In *Proc. of the 13th Workshop on Temporal and Spatial Information Processing*, TASIP '01, pages 9:1–9:8, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Schmachtenberg et al., 2014] Schmachtenberg, M., Bizer, C., and Paulheim, H. (2014). Adoption of the Linked Data Best Practices in Different Topical Domains. In *Proc. of the 13th Int. Semantic Web Conference*, volume 8796 of *ISWC '14*, pages 245–260. Springer International Publishing.

[Seitner et al., 2016] Seitner, J., Bizer, C., Eckert, K., Faralli, S., Meusel, R., Paulheim, H., and Ponzetto, S. (2016). A large database of hypernymy relations extracted from the web. In *Proc. of the 10th Edition of the Language Resources and Evaluation Conference*.

[Sekhavat et al., 2014] Sekhavat, Y. A., di Paolo, F., Barbosa, D., and Merialdo, P. (2014). Knowledge Base Augmentation using Tabular Data. In *Proc. of the 7th Workshop on Linked Data on the Web*, LDOW '14.

[Seth et al., 2010] Seth, S., Jandhyala, R., Krishnamoorthy, M., and Nagy, G. (2010). Analysis and Taxonomy of Column Header Categories for Web Tables. In *Proc. of the 9th IAPR Int. Workshop on Document Analysis Systems*, DAS '10, pages 81–88, New York, NY, USA. ACM.

[Singhal, 2012] Singhal, A. (2012). Introducing the Knowledge Graph: Things, Not String. Blog. Retrieved June 07, 2017.

[Spiegler, 2013] Spiegler, S. (2013). Statistcs of the common crawl corpus 2012. Technical report, Technical report, SwiftKey.

[Staab and Studer, 2009] Staab, S. and Studer, R. (2009). *Handbook on Ontologies*. Springer.

[Suchanek et al., 2011] Suchanek, F., Abiteboul, S., and Senellart, P. (2011). Paris: Probabilistic alignment of Relations, Instances, and Schema. *Proc.of the VLDB Endowment*, 5:157–168.

[Suchanek et al., 2007] Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *Proc. of the 16th Int. Conference on World Wide Web*, WWW '07, NY. ACM Press.

[Surdeanu and Ji, 2014] Surdeanu, M. and Ji, H. (2014). Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population Evaluation. http://nlp.cs.rpi.edu/paper/sf2014overview.pdf.

[Syed et al., 2010] Syed, Z., Finin, T., Mulwad, V., and Joshi, A. (2010). Exploiting a Web of Semantic Data for Interpreting Tables. In *Proc. of the 2nd Web Science Conference*.

[Syed, 2010] Syed, Z. S. (2010). *Wikitology: A Novel Hybrid Knowledge Base Derived from Wikipedia*. PhD thesis, Catonsville, MD, USA. AAI3422868.

[Tejada et al., 2001] Tejada, S., Knoblock, C. A., and Minton, S. (2001). Learning object identification rules for information integration. *Information Systems*, 26(8):607 – 633. Data Extraction,Cleaning and Reconciliation.

[Torzec, 2014] Torzec, N. (2014). Yahoo's Knowledge Graph. Retrieved June 07, 2017.

[Tschirschnitz et al., 2017] Tschirschnitz, F., Papenbrock, T., and Naumann, F. (2017). Detecting inclusion dependencies on very many tables. *ACM Trans. Database Syst.*, 42(3):18:1–18:29.

[Udrea et al., 2007] Udrea, O., Getoor, L., and Miller, R. J. (2007). Leveraging Data and Structure in Ontology Integration. In *Proc. of the 2007 ACM SIGMOD Int. Conference on Management of Data*, SIGMOD '07, pages 449–460, New York, NY, USA. ACM.

[Venetis et al., 2011] Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering Semantics of Tables on the Web. *Proc. of the VLDB Endow.*, pages 528–538.

[Verykios et al., 2000] Verykios, V. S., Elmagarmid, A. K., and Houstis, E. N. (2000). Automating the approximate record-matching process. *Information Sciences*, 126(14):83 – 98.

[Vrandečić and Krötzsch, 2014] Vrandečić, D. and Krötzsch, M. (2014). Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM*, 57(10):78–85.

[Wang et al., 2012] Wang, J., Wang, H., Wang, Z., and Zhu, K. Q. (2012). Understanding Tables on the Web. In *Proc. of the 31st Int. Conference on Conceptual Modeling*, pages 141–155.

[Wang and Cohen, 2008] Wang, R. C. and Cohen, W. W. (2008). Iterative Set Expansion of Named Entities Using the Web. In *Proc. of the 8th IEEE Int. Conference on Data Mining*, ICDM '08, pages 1091–1096.

[Wang and Hu, 2002a] Wang, Y. and Hu, J. (2002a). A Machine Learning Based Approach for Table Detection on the Web. In *Proc. of the 11th Int. Conference on World Wide Web*, WWW '02, pages 242–250, New York, NY, USA. ACM.

[Wang and Hu, 2002b] Wang, Y. and Hu, J. (2002b). Detecting Tables in HTML Documents. In *Proc. of the 5th Int. Workshop on Document Analysis Systems*, pages 249–260.

[Yakout et al., 2012] Yakout, M., Ganjam, K., Chakrabarti, K., and Chaudhuri, S. (2012). InfoGather: Entity Augmentation and Attribute Discovery by Holistic

Matching with Web Tables. In *Proc. of the 2012 ACM SIGMOD Int. Conference on Management of Data*, pages 97–108.

[Yin and Tan, 2011] Yin, X. and Tan, W. (2011). Semi-supervised Truth Discovery. In *Proc. of the 20th Int. Conference on World Wide Web*, WWW '11, pages 217–226. AC.

[Yin et al., 2011] Yin, X., Tan, W., and Liu, C. (2011). FACTO: A Fact Lookup Engine Based on Web Tables. In *Proc. of the 20th Int. Conference on World Wide Web*, WWW '11, pages 507–516, New York, NY, USA. ACM.

[Yoshida and Torisawa, 2001] Yoshida, M. and Torisawa, K. (2001). A method to integrate tables of the world wide web. In *Proc. of the 1st Int. Workshop on Web Document Analysis*, pages 31–34.

[Zanibbi et al., 2004] Zanibbi, R., Blostein, D., and Cordy, J. (2004). A survey of table recognition. *Int. Journal on Document Analysis and Recognition*, 7:1–16.

[Zhang and Chakrabarti, 2013] Zhang, M. and Chakrabarti, K. (2013). InfoGather+: Semantic Matching and Annotation of Numeric and Time-varying Attributes in Web Tables. In *Proc. of the 2013 ACM SIGMOD Int. Conference on Management of Data*, pages 145–156.

[Zhang, 2014a] Zhang, Z. (2014a). Learning with Partial Data for Semantic Table Interpretation. In Janowicz, K., Schlobach, S., Lambrix, P., and Hyvönen, E., editors, *Proc. of the 19th Int. Conference on Knowledge Engineering and Knowledge Management*, EKAW '14, pages 607–618, Cham. Springer International Publishing.

[Zhang, 2014b] Zhang, Z. (2014b). Towards efficient and effective semantic table interpretation. In *Proc. of the 12th Int. Semantic Web Conference*, ISWC '14, pages 487–502. Springer.

[Zhang, 2016] Zhang, Z. (2016). Effective and Efficient Semantic Table Interpretation using TableMiner+. *The Semantic Web Journal.*

[Zhao, 2007] Zhao, H. (2007). Semantic Matching Across Heterogeneous Data Sources. *Commun. ACM*, 50(1):45–50.

[Zhao and Ram, 2007] Zhao, H. and Ram, S. (2007). Combining schema and instance information for integrating heterogeneous data sources. *Data & Knowledge Engineering*, 61(2):281 – 303.

[Zhu et al., 2016] Zhu, E., Nargesian, F., Pu, K. Q., and Miller, R. J. (2016). LSH Ensemble: Internet-scale Domain Search. *Proc. of the VLDB Endow.*, 9(12):1185–1196.

[Zwicklbauer et al., 2013] Zwicklbauer, S., Einsiedler, C., Granitzer, M., and Seifert., C. (2013). Towards disambiguating Web tables. In *Proc. of the 12th Int. Semantic Web Conference*, ISWC '13, pages 205–208.