Schema Matching of Relational Web Tables
- A Comparison -

Seminar Paper

presented by
Alexander Müller
Matriculation Number 1376818

submitted to the
Chair of Information Systems V
Dr. Heiko Paulheim
University Mannheim

August 2014

Contents

List of Figures

List of Tables

1 Introduction

The World Wide Web contains a huge number of structured information in form of HTML Tables. This tables contain different information, ranging from the current standing of your favorite soccer league to tables displaying information about technical details of cameras. Certainly, this big source of information has an enormous potential for the usage in areas like Data Mining or Natural Language Processing [9].

Notably more valuable are tables with an equal quality like relational tables. The fraction of this tables is relatively small in comparison to the total number of tables. However, due to the vast size of the web, there still exist many of them. Each of these relational tables can now be seen as a small relational database, which have their own predefined schema and contain various entities [10]. A project that extracts these tables is the Web Data Commons Project [3]. In this project they were able to extract 147 million relational web tables using the Common Crawl [1] as the data source.

As this number of tables suggest, a usage is only possible if an easy access to information is possible. One way of doing so that the user can query tables, which complement his own tabular information [13]. Thus, users gather new information out of a potentially high number of tables, which in an ideal case extend their present information. Certainly, to leverage the full potential of the returned tables they need to be aligned in a way that a user gets an integrated view. Integrating this table corpus yields various problems among of which the integration of schemata is one of the most dominant ones [29]. Therefore the well-known but not yet solved, problem of schema matching is tackled in this seminar paper and applied to schema mediation from relational tables from independent sources of the web. As a result, the accessibility and usefulness of the retrieved tables will be enhanced which helps users to make use of big web tables corpora more easily.

The remainder of this paper is organized as follows: In Section 2 the problem of schema matching and how it's applied to matching web tables is defined more precisely. After this in Section 3 a schema matching technique classification taxonomy is introduced [29] and multiple examples of existing approaches to schema matching are given. The core contributions of this seminar paper are presented in Section 4, where in Section 4.1, the implemented individual schema matching techniques are introduced and in Section 4.2 multiple combinations of them are proposed. These techniques are in Section5 evaluated by the use of three -newly created- gold standards for matching web tables from the web data commons table corpus. Finally in Section 6 a conclusion from the results of the conducted experiments is drawn.

2 Problem Statement

The problem of schema matching is a problem which is present since the early 1980s. The goal was to develop an global view over different independently developed schemata or ontologies.[5] This has not fundamentally changed since Euzenat and Shvaiko, define the process of ontology matching as finding the correspondence between different elements of an ontology. An ontology in this case can be everything that is expressed in a machine-readable format, like database schema, xml descriptions, web service descriptions among others [16].

Rahm & Bernstein define a match operator that takes as two schema as an input and returns so called output mappings, which describe the relation between the elements of the input schema. A schema is this case is a set of elements connected by some structure. These elements are then mapped based on their semantic correspondence, where different types of matchings are possible: Two elements S_1 and S_2 can either be directly related so S_1 same-as S_2. S_2 can be a generalization of S_1, so that S_2 is-a S_2. Furthermore S_1 can be only a part-of S_2, where for instance S_1 is a field called *first name* and S_2 is a field called *full name* ([29],[28]).

The focus in this paper lies on finding semantic correspondence between web tables. This web tables are provided by the [2] Project, which were able to extract over 147 million relational tables from the web in the web data commons *Web Table Corpus*. This tables contain a huge amount a potentially useful data, which need to be made accessible. The tables are dived into different subset of different categories like country, cities or cameras. Merging this tables to one data source, will need a step of schema integration where a global view of unique schema elements is created. A good example is the column *(*name) which occurs in over 5 million tables, if now merging all tables for instance of the country subset, the global view should contain one column describing the name of the country instead of thousands of columns containing redundant information.[3] In the remainder of this seminar paper the more specific term schema matching instead of ontology matching is used, since data set consists of relational tables, where the underlying ontology is classified as a database schema.

Because of the nature of the underlying data set the schema information for each table only consists out of column names and data type definitions as derived from a heuristic guessing mechanism.

Finally the problem investigated in this seminar paper is different from the alignment of two database schema, or two ontologies. The problem to solve is finding a set of semantic correspondences in a huge corpus of web tables, to build an integrated schema, eliminating duplicate schema elements and therefore make the data more accessible for potential future use cases.

3 Related Work

There exists various Schema Matching approaches in literature, which shows the need of a taxonomy or classification of different schema matching approaches. Therefore this seminar paper follows the classification of schema matching approaches given by Rahm & Bernstein, which is illustrated in Figure 1 [29].

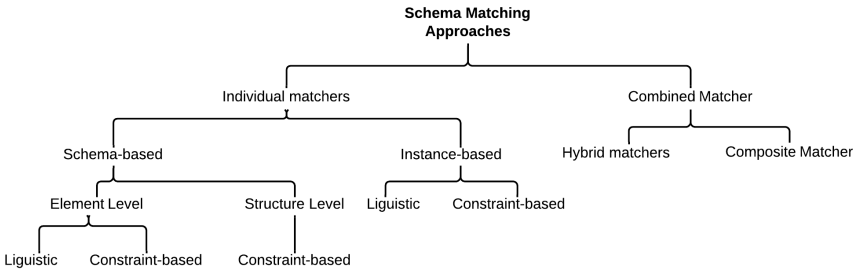

Figure 1: Classification of schema matching approaches according to Rahm & Bernstein [29]

In general the approaches are not classified by their given input, it's further generalized that each approach can deal with all input formats. The first level of distinction of schema matching systems is whether they combine different base matcher or consists only of an individuals matcher.

There exists two types of combined matcher: First to mention are Hybrid Matcher which directly combine different matching approaches by using different information sources. Second Composite Matcher are mentioned, those execute multiple individual matcher (among which might be Hybrid Matcher) and in the end combine the matching results to get one matching. In general Rahm & Bernstein suggest that often Hybrid Matcher perform better than running multiple matcher in a Composite Matcher, but with the trade-off of less flexibility.

The group of individual matcher is roughly divided into schema-based and instance-based matching techniques. For schema-based matcher there exist again a separation between different classes, first of which are element-level matcher. Which can either use the linguistic information of the element (e.g. label, column name) or take into account constraint information like database keys or data types. Another group of schema-based Matcher are structure-level matcher, they take into account the hierarchical structure of the underlying schema, like present in XML-Schemata and Ontologies and based on this graph-like information create an mapping of schema elements.

Instance-based matcher take into account the actual underlying instance data of the related schema, instead of just using meta data. But again this group can be divided into two groups. The first group are constrained-based matcher, where they use pattern of data like value distributions or more specific statistical key figures. The second groups is based on linguistic techniques: There the instance data of each column is inspected and often compared using techniques originated in the research area of information retrieval.

In addition to that rough classification of existing techniques, matching systems that have been developed focused either on a pair-wise schema matching, where always two schema are compared or a holistic matching, where an arbitrary number of schemata are matched.[28]

For schema-based matching there exists various approaches mostly relating on string distance metrics [11] or semantic relatedness ([8] , [33]). Other approaches try to use other auxiliary information to enrich schema and therefore derive better alignment ([20], [21]).

Due to the broader definition of instance-based matching, there exists various approaches. A lot of them are inspired by techniques used for Information Retrieval and try to optimize them either by pruning the search space [7] or by speed up the similarity computation [15]. Other approaches try to tackle schema-matching from a probabilistic point-of view and therefore derive matching by additionally exploiting the structure of schema [32].

Many of those matching techniques are part of more complex matching systems which use various matcher in order to overcome the vulnerability of single algorithms. An example of such a matching system is COMA ([27] [14]). It combines different base and hybrid matcher to combines them to composite matcher. Another example for a matching system is CUPID [6], which combines matchings based on names, data types and instance-based information of schema elements to robust final matching result.

The matching approaches used in this seminar paper are presented more in detail in 4.1.

4 Implemented Approach

The system implemented in this seminar paper, combines multiple individual matching approaches to a composite and partially hybrid matching system. In Figure 2 the general process for matching a set a tables is shown. First of all tables selected from the web tables corpus are parsed and transformed into an internal representation suitable for the matching task. After this the matching step is performed which output is a matching that contains all found semantic correspondences between the

input tables. This matching is in a final step used to create an integrated schema for the given input tables.

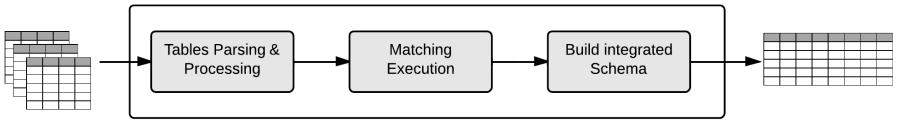

Figure 2: Overview of the Implemented Schema Matching Pipeline

The matching step is certainly the most important step in the matching pipeline. It can consists only out of one base matcher (see Section 4.1), or be a more complex pipeline execution of multiple base matcher executed sequentially or in parallel. How the execution of multiple matcher is designed is presented in Section 4.2.

4.1 Individual Matcher

In this section an overview of the approaches to enable schema matching is presented. First three different instance-based matcher are presented, following by three matcher that use string similarity and semantic relatedness in order to validate whether to match columns or not.

Not influenced by the underlying algorithm is the representation of a single match or correspondence between two columns or schema elements. This values expresses the similarity between two columns originating from different tables that is defined by the following equation:

$$sim(a, b) = x : a \in A \wedge b \in B \wedge A, B \in T \wedge A \neq B \wedge x \in [0, 1] \qquad (1)$$

So one can see that the value of similarity assessed by comparing two columns a,b from two non-equal tables A,B from a corpus T has always a value between 0 and 1, where 0 indicates a complete dissimilarity and 1 defines perfectly matching columns. Based on this similarity measure the matchings can be pruned. All of the used matching approaches follow this definition of column similarity.

4.1.1 Instance-based Approaches

Vector Space Model

The Vector Space Model (VSM) was introduced as a technique that allows automatic retrieval of unstructured data in an efficient way. Key components of the model are the usage of a bag-of-words approach that means that in a document

representation the word order is not considered, the tf-idf weighted vector representation of a document and the usage of cosine similarity to compare those vectors.[31]

The tf-idf weighted vector representation of a document, weights each word occurring in a document. Therefore it takes into account how often the word occurs in the given document (tf) and weight this with the inverse of the overall document frequency (idf), which describes the amount of documents the word is in.[31] This can be expressed as in the follow formula:

$$tfidf_{t,d} = tf_{t,d} * idf_t = tf_{t,d} * log(\frac{N}{df_t}) \tag{2}$$

Now having those vectors the similarity of two documents is given by the angle between their n-dimensional vector representations. This is often described by the length-normalized cosine similarity, which means when two vectors point in the exact same direct, they will have a similarity value of 1. Otherwise if they point into complete other directions they will have a similarity value of 0 .

Like mentioned by Euznat & Shaviko this techniques can be used for performing schema matching [16]. Therefore as part of this seminar paper, a straightforward approach was implemented to match two schemata based on the underlying instances, with the help of the VSM. To mention is that the reference document collection when comparing two column with each other consists only out of the compared columns, which means that the document frequency df_t can only 1 or 2. In the Table 1 and 2 example data to match columns with a VSM approach can be seen.

Country	dial code	rate
Australia	61	$0.12
China	86	$0.12
Chile	56	$0.12
United Kingdom	44	$0.11
Germany	49	$0.11
South Korea	82	$0.12

Table 1: Sample from a Table of the Web Data Commons Corpus for Countries

In order to compare two tables and therefore integrate their underlying schemata, the tf-idf vector representation of their column are compared by using cosine similarity. Based on this similarity and a user defined threshold a matching can be build and used for schema integration. For instance while comparing the *column* country of Table 1 with the column *country name* of Table 2 the following computations are necessary.

ISO 3166 code	Country name	Number of ip addresses
us	United States	1562171948
cn	China	304309986
jp	Japan	200525715
gb	United Kingdom	121789176
de	Germany	113924203
kr	Korea Republic of	106865368

Table 2: Another Sample from a Table of the Web Data Commons Corpus for Countries

$$c\_name_{tf_idf} = \begin{pmatrix} 0.447213595 \\ 0.447213595 \\ 0.447213595 \\ 0.223606798 \\ 0.223606798 \\ 0.223606798 \\ 0.447213595 \\ 0.223606798 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad c_{tf_idf} = \begin{pmatrix} 0.280938208 \\ 0 \\ 0 \\ 0.280938208 \\ 0.280938208 \\ 0.280938208 \\ 0 \\ 0.280938208 \\ 0.499445703 \\ 0.499445703 \\ 0.499445703 \end{pmatrix} \tag{3}$$

$$sim_{cos}(c_{tf_idf}, c\_name_{tf_idf}) = \frac{c_{tf_idf} \cdot c\_name_{tf_idf}}{|c_{tf_idf}| * |c\_name_{tf_idf}|} = 0.377 \tag{4}$$

DUMAS

Another pair-wise instance-based approach also making use of the VSM is DUMAS [7]. DUMAS uses duplicates among two tables to infer a schema matching. Instead of simply comparing the tables column-wise, so vertically, it performs as they call it horizontal matching, to find duplicates.

To detect duplicates among the compared table they use the tf-idf representation of the rows. Based on this representation they calculate the similarity of the rows not by computing all pair-wise similarity of all of them, but make use of the Whirl algorithm [12]. This returns the top-k duplicates, and so most expressive duplicate, with respect to their cosine similarity.

The basic idea of DUMAS is now that columns with similar content refer to the same concept and therefore should be matched. Therefore for each duplicate-pair

the pair-wise similarity of all fields is computed. In order to do so they make use of the Soft-TFID measure, which is a variation of tf-idf also considering similar terms.[11] After this the similarity values are aggregated over all duplicate-pairs resulting in a final matrix. This matrix is now used to build the final matching between the two tables.

Li & Clifton Matcher

A final instance-based matcher was consider to be evaluated. This matcher, in contrast to other instance-based matcher, does not inspect each underlying instance of a schema directly but using aggregation functions to characterize patterns of the distribution of data at instance level. Hence an approach to summarize instance data into aggregated key figures proposed by Li & Clifton is used.[25] In this paper 3 metrics for characterizing character fields and 3 metrics to summarize numeric fields are presented in Table 3.

In order to related each column or schema element to each other, Li & Clifton now use Neural Networks to train weight for this measures. Finally, by using Neural Networks, they group the schema elements into distinct clusters, where each element have the same semantic meaning. Due to the limited amount of resources for this paper a different, more simple approach was used to compare schema elements based on this aggregations.

The first step for schema matching is actually calculating the measures for each column of a given table. After this for a pair-wise comparison of the columns of two tables is performed where the values of their aggregations are compared. Instead of comparing the values by their direct equality a softer measure is introduced as defined by the following function:

$$softequals(a, b, threshold) = \begin{cases} |(a-b)| > threshold & false \\ |(a-b)| \leq threshold & true \end{cases} \tag{5}$$

Using this function a and b are the derived aggregations metrics from Table3 and the $threshold$ is a user-defined value. To compare a column with another all aggregation values are compared and it is counted for which of them the $softequals$ function returns $true$. Consequently this $true$-count is divided by the total number of metrics, to get the ratio, which is furthermore seen as the similarity value of the two columns. Clearly the threshold is important to derive meaningful results out of this approach and is therefore subject to optimization as presented in Section 5.3.

4.1.2 Schema-based Approaches

As in section 3 presented Schema-based matching approaches only use schema level information like constraint and column names to create a matching between

Name	Description	Data type
Number ratio	Describes the average ratio of the number of numerical characters to the total number of characters, of a whole column	String
Whitespace ratio	How many white-spaces occur on average in a field compared to the total number of characters	String
Statistic of the string length	Maximum, average, variance, coefficient of variance of the string length of the columns	String
Average	The average over all values of the column	Numerical
Variance	The variance over all values of the column	Numerical
Coefficient of variation	The coefficient of variation is the squareroot of the standard deviation divided by the mean of the values	Numerical

Table 3: Mesasure to compare the properties of schema elements according to Li & Clifton [25]

two schemata. Furthermore usually a distinction between label-based and structure-based methods is made. Because the structure of the schema of a web table is flat and simple the focus in this seminar paper lies on approaches that only need a label string as an input.

Further differentiation can be made in the used techniques which are used to compare two string with each other.[16] The first method class are string-based methods where the structure of the string as sequence of characters is exploited. There exist a variety of methods that can be used to compare string, where here the usage of the levenshtein distance will be evaluated. [11]

In contrast to this there exists other methods that measure the semantic relatedness or the semantic similarity of two strings. Relatedness is the more general concept than similarity. Where ”Car” and ”fuel” are clearly very semantically re-

lated but are not semantically similar, on the other side "car" and "bicyle" are semantically related and semantically similar, because both are used to get from one point to the other.[8] In this section there are two approaches presented which use WordNet [17] to determine semantic similarity and use this for schema matching.

Levenshtein based

The Levenshtein or Edit distance σbetween two string s and t is the minimal number of operations that need to be applied, to convert one string to the other. According to Levenshtein there exist three operations *insertions*, *deletions*, *substitutions*. [24] So for instance the levenshtein distance between *schema* and *ontology* is 8 where the distance between *mannheim* and *mannem* is 2. This shows this method can be used to align different column names even though there are differently spelled. To define it formally ,$\sigma(r, s)$ is the sum of the weights $w(o)$of the necessary edit operations to convert s to t, where $w(o)$ can be different for each operation o.

$$\sigma(s, t) = \sum w(o) \tag{6}$$

The implemented matcher using the levenshtein distance is comparing the column names in the schema of one table to the column names of the schema of the table it compares to. To get the similarity instead of the distance $1 - \sigma(r, s)$ is computed and consequently the column pairs whose similarity is higher than a user defined threshold are regarded as a new correspondence.

Jiang & Conrath

The Jiang & Conrath approach[22] is based on the Resnik Information Content (IC) - based approach to asses semantic similarity between two words. [30] This approach again is the WordNet Hypernym trees. A hypernym of a word is a more general term like for instance the hypernm of red is color. So one can say a hypernym express a unidirectional is-a relationship between two words. WordNet is a large English lexical database which assigns concepts to words and expresses these in a hierarchy. Over the hypernym trees each concept in WordNet is related to one of 11 general concepts like *Physical Entity*, where the maximum path length between a leaf and a root concept is 16. Important to stress is the difference between a word and a concept. A word refers to one or more concepts, and a concept can represent more than one word. For instance the word staff relates to five different concepts, like *personnel* and the *body of teachers*. Where for instance the term *work force* relates to the same concept.

The basic idea of Resnik is that the similarity of two concepts is defined by the "extent to which they share information" [30]. This can be in a hypernym tree expressed by the relative position of the least super ordinate (lso), which is the most

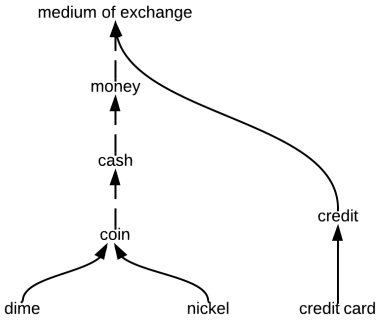

Figure 3: Fragment of a WordNet Hypernym Tree. Adpadted from Budanitsky & Hirst [8]

common subsumer of those two words. In Figure 3 a hypernym tree is given. One can see that the lso of *dime* and *nickel* is *coin* where the lso of *nickel* and *credit card* is *medium of exchange*. Furthermore it's clearly that the first pair is more similar than the second one. This is expressed by following equation:[8]

$$sim(c_1, c_2) = -log(p(lso(c_1, c_2))) \tag{7}$$

The formula expresses the idea that a word related to c_1 and word relating to c_2 are more similar when the Information Content $IC(c) = -logp(c)$ of their lso is higher. Clearly the less general a term is the higher the information content is, which underlies the same idea as in the idf-weight in the VSM. But in order to get the IC of a concept the probability $p(c)$ needs to be estimated. In practice this is done by counting the occurrence of words in corpus relating to a concept divided by the whole number of words. Which is also called Maximum Likelihood Estimation.[26] Now because $p(coin)$ is lower than the $p(medium\_of\_exchange)$ the IC *coin* is higher and therefore the terms *dime* and *nickel* are more similar. In this seminar thesis a library is used, which also contains estimation probability of word based on the Brown Corpus.[18] The weakness of this approach is that Resniks measure does not take into account the distance of the concepts to its lso. [8] Jiang & Conrath [22] adressed this issue by taking the path from the concepts to compare to it's lso in account. Therefore they use a measure for calculating the distance of a concept c to its parent $par(c_1)$:

$$dist_{JC}(c, par(c)) = IC(c) - IC(par(c)) \tag{8}$$

Now the distance between two arbitrary concepts is given by the distance along the

shortest path between them, excluding the lso.

$$dist_{JC}(c_1, c_2) = \sum_{c \in Path(c_1,c_2)\ ls(c_1,c_2)} dist_JC(c, par(c)) \quad (9)$$

$$related_{JC}(c_1, c_2) = \frac{1}{dist_{JC}(c_1, c_2)} \quad (10)$$

The relatedness measure is used to perform schema matching between semantically similar words. Therefore column headings with more than one token are tokenized and then the pair-wise normalized similarity is calculated and used to match different column.

Adapted Lesk Algorithm

Another algorithm to asses semantic relatedness is proposed by Banerjee and Pedersen [4], which is originally designed for word disambiguation. This algorithm is an adaption of the original Lesk algorithm, which asses relatedness for word disambiguation by comparing the dictionary definition, or gloss, of words.[23] Lesk uses the definition of the Oxford Advanced Learners dictionary, and defines an overlap of the gloss of two words as "the longest sequence of one or more consecutive words that occur in both glosses"[4]. This measure is now adapted by Banerjee and Pedersen to use the strength of WordNet. They compare the glosses of the concepts relating to the words and thus derive relatedness of two words. Furthermore they not only use the gloss of the concepts, but extends the comparison by the usage of the glosses of hypernyms, hyponym, holonym etc. This measure of relatedness is again used pair-wise for the tokenized labels to finally derive matchings for schemata.

4.2 Matcher Combination

The pipeline of the matching system in this seminar paper was designed to allow both executing matcher logically parallel and chaining different matching approaches to execute the pipeline in a serial way and therefore allow for instance the pruning of the search space by one matcher, to reduce the runtime and errors of a second matcher. Generally speaking the matching part of a pipeline consists out of a series of execution steps, which execute base matcher (see 4.1 Furthermore each base matcher has at least two attributes. The first one indicates whether or not the matcher runs in a chained mode. When this chained mode is true, this means that for the next matching steps only columns will be used that this matcher in the chained mode has found correspondence for. This is designed to allow data pruning steps. The second parameter is a threshold parameter. If this parameter is enabled the set of correspondences found thus matcher will be pruned so that only

those will be kept that are above the specified threshold. The execution steps can

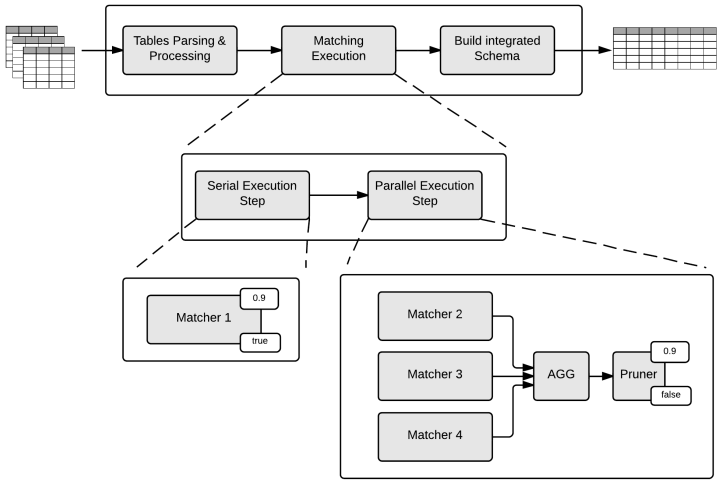

Figure 4: Possibilities to combine different base matcher

wrap one matcher (Serial Execution Step) or multiple matcher (Parallel Execution Steps). One can see in Figure 4 how the combination of the different execution steps works. First of all the pipeline executes each step after another. For a serial execution step this is straight forward as only one matcher needs to be called. Another pictures draws the execution of parallel execution steps, where multiple matcher are involved. First of all all matcher are executed logically in parallel (not in a single thread) and therefore there exits after this step multiple results, consequently a method to aggregate those is needed. There exist a variety of methods to combine multiple alignments. [16] In this seminar paper two basic methods are implemented, first of all similar correspondences are aggregated using the average. And secondly the best score for each match is kept. After the scores are aggregated each serial execution step has a optional pruning step, at this step the results are pruned by a given threshold which is again a user defined parameter for the execution step.

For the evaluation of this chaining architecture, multiple pipeline were designed and used to match data sets. Please remark that Figures 5 - 11 contain for the threshold values no constant value, but variables, which will be using a One-after another parameter optimization method optimized.

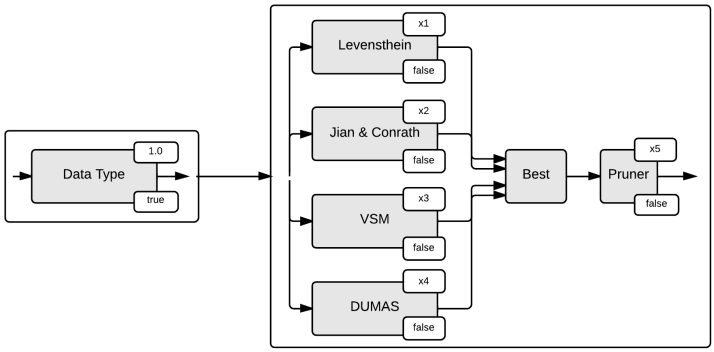

Figure 5: Matching Pipeline 1

Pipeline 1 is illustrated in Figure 5. This pipeline contains two execution steps, of which the first is a data pruning step so that only columns with the same datatype are matched in the parallel matcher. As an aggregation method was the best matching method chosen.

For pipeline 2 (Figure 6) a similar set-up as for pipeline 1 was chosen, in order to validate whether a search space pruning has positive implications to the performance of the system or not.

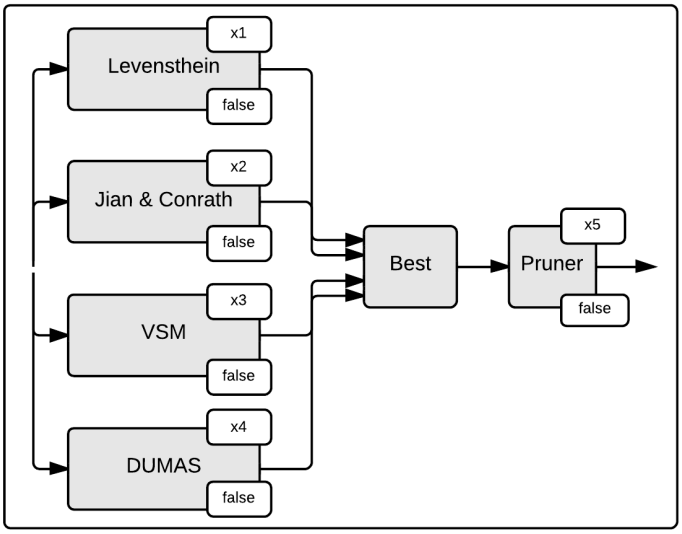

Figure 6: Matching Pipeline 2

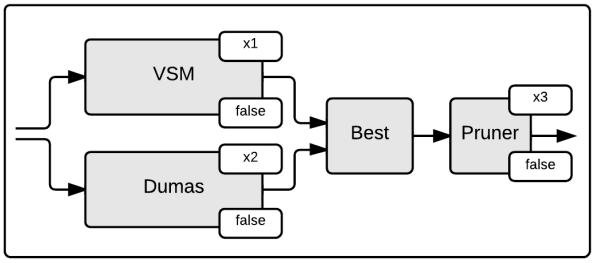

Figure 7: Matching Pipeline 3

The goal of pipeline 3 (Figure 7) is to discover whether the implemented instance-based techniques perform better together, than alone. Therefore their matchings scores are aggregates via the AVG function.

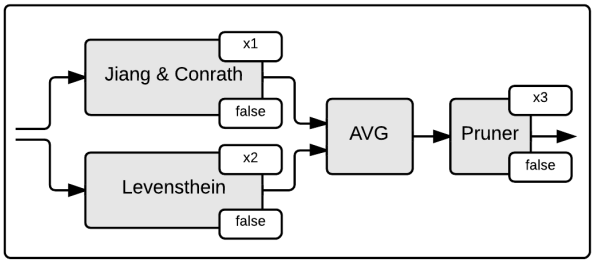

Figure 8: Matching Pipeline 4

Pipeline 4 (Figure 8) tries to eliminate the problems of string-based similarity measures and semantic relatedness, by combining them. For each matching the one is kept which has the highest similarity value.

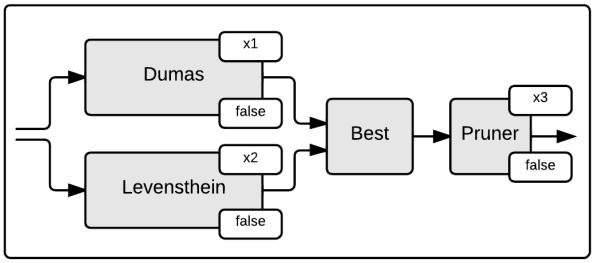

Figure 9: Matching Pipeline 5

Pipeline 5(Figure 9) and 6 (Figure 10) were implemented in order to enhance the performance of the system by combining instance-based matcher with one schema-based matcher. In the end matchings of both are aggregated via the best function.

The pipeline 7 now tries to combine all three kinds of different algorithms implemented, by combining one element-level instance-based matcher, one structure-level instance-based matcher and one schema-based matcher. At the pipeline in afterwards only the best score for each matching is kept in order to get the strength

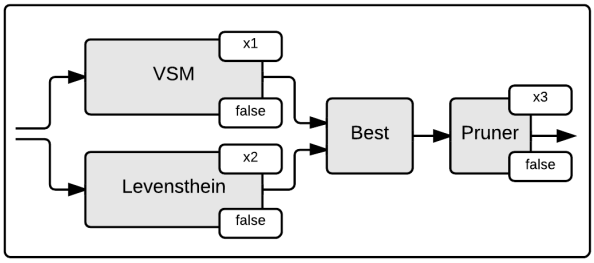

Figure 10: Matching Pipeline 6

of all algorithms.

The performance of the designed pipelines and the single base matcher in terms of result quality is going to be evaluated with three different data sets in the following section.

5 Evaluation

5.1 Used Data set

The evaluation of the implemented base matcher and matching pipelines is facilitated with the use of three datasets drawn from the Web Data Commons tables corpus [3]. For those three corpora a gold standard was manually created, to measure the performance of different implemented approaches. Table 4 now summarizes the different data sets and gold standards used. One can see that the sets fall into three different topics.

Dataset	Countries	Cities	Cameras
No. of Tables	57	41	41
No. Columns	334	255	204
No. of Unique Columns	266	61	93
No. Of Matchings	1298	2844	750

Table 4: Information relating the used Evaluation Data sets

The first set only contains tables relating countries, typical schema elements are field like *Name*, *Currency* or *Dial Code*. In total the corpus consists out of 57 tables, with a total 334 columns, from which are 266 containing unique infor-

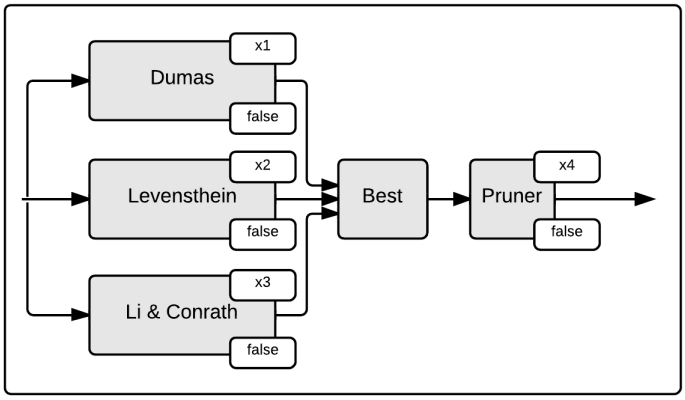

Figure 11: Matching Pipeline 7

mation. In order to match those columns with each other a total number of 1298 were necessary, please remark that this high number of matching is possible due to the fact that each correspondence is counted as one matching. So for instance if a column A has two matches, with column B and C, this counted as two matchings and not one for A.

The second data set contains tables with entities relating to Cities, with elements like *Name*, *Population* and *State*. For this corpus a gold standard was created consisting out of 41 tables, with a total number of 255 columns, from which only 61 are unique, which results in the number of 2844 matchings.

In addition another table corpus containing entities relating to cameras is used. This data set is made up 41 tables, with a total number of 204 columns, for which 750 matchings were created, resulting in 93 unique columns. A challenge of this corpus is that 2 table headings are in German, one in French and another one in Italian. Most probably these columns will not be possible to match using simple string metrics.

The three created gold standard were eventually used to evaluation the implemented pipelines and all single matcher using the evaluation metric, present in the following section.

5.2 Evaluation Metric

To compare the result of the implementing matchings systems with each other, in this paper the standard measures as used by the OAEI are used, to measure

the quality of the produced alignment. [19] Precision, Recall and the F-Measure, which are defined as the following:

$$Precision = \frac{tp}{tp + fp} \tag{11}$$

$$Recall = \frac{tp}{tp + fn} \tag{12}$$

$$F_1 = \frac{2 * P * R}{P + R} \tag{13}$$

5.3 Evaluation Results

All matcher and matching combinations presented in Section 4.1 and Section 4.2 were now applied to all three Goldstandard data sets presented Section5.1. In order to optimize the results a simple one after another parameter optimization was executed, where the thresholds whether to keep a column correspondence or not was optimized. The result of the experiments are illustrated in Table 5.

	Dataset	Countries			Cities			Cameras		
	Measure	Precision	Recall	F1-Measure	Precision	Recall	F1-Measure	Precision	Recall	F1-Measure
Schema	Levenshtein	0.60	0.27	0.37	0.81	0.76	0.78	0.60	0.39	0.47
	Jiang & Conrath	0.210	0.418	0.279	0.723	0.773	0.747	0.762	0.423	0.544
	Lesk	0.210	0.418	0.279	0.723	0.773	0.747	0.762	0.423	0.544
Instance	VSM	0.503	0.609	0.551	0.761	0.308	0.439	0.215	0.651	0.323
	Dumas	0.514	0.735	0.605	0.729	0.245	0.367	0.664	0.556	0.605
	Li & Clifton	0.216	0.029	0.052	0.798	0.222	0.347	0.804	0.169	0.280
Pipeline	Pipeline 1	0.301	0.782	0.434	0.801	0.766	0.783	0.644	0.693	0.668
	Pipeline 2	0.301	0.782	0.434	0.801	0.766	0.783	0.644	0.693	0.668
	Pipeline 3	0.528	0.689	0.598	0.734	0.322	0.447	0.637	0.555	0.593
	Pipeline 4	0.040	0.840	0.077	0.160	0.866	0.271	0.070	0.985	0.131
	Pipeline 5	0.501	0.749	0.601	0.801	0.766	0.783	0.682	0.633	0.657
	Pipeline 6	0.485	0.682	0.567	0.809	0.758	0.782	0.455	0.437	0.446
	Pipeline 7	0.518	0.710	0.599	0.797	0.766	0.781	0.718	0.505	0.593

Table 5: Evaluation Results for all executed Schema Matching runs, with 3 Gold Standards

The table shows for each data set the results for each single matcher and the pipeline respectively, the evaluation measures Precision, Recall and F1-Measure. By inspecting the table the most prominent observation, which can be made is the fact that different matching techniques perform different on the data sets. Especially a clear distinction between schema-based and instance-based techniques with the respect to the given data set can be made. Schema-based matching algorithms need a high quality schema, for which columns with equal meaning are having similar or at least related names. On the other hand instance-based matching system rely on a high overlap between the different data set. This explains for instance the difference for instance-based matcher for the data set cities. This data set contains a lot of schemata having the same layout but having different instance data. This table schemata can only weakly be matched by algorithms relying on instance data without aggregating it. Furthermore it was observed that the DUMAS algorithm performs better than the VSM matcher, when the data contains duplicates, if not the VSM method delivers the higher quality of results.

For the implemented combination mechanisms a mixed impact can be observed. Combining different algorithms of the same group does not have a remarkable positive impact. Pipeline 3 & 4 they still are tending to be sensitive to the the same kind of data set. But combining different matcher from different categories enhance the robustness of the system. The most stable and best performance throughout the data sets have pipeline 5, 6, 7 which all combine matcher of different data sets. The best performance of all algorithms for all data sets used for evaluation has the pipeline 5 which combines DUMAS and Levenshtein matcher.

6 Conclusion and Future Work

In this seminar paper a validation of existing schema matching techniques was conducted. Therefore different matching techniques were implemented and afterwards with a combination mechanism integrated to pipelines which are less susceptible to characteristics of the data set for which the matching was performed. In general the techniques combined in the pipelines arise two categories: schema-based and instance-based. Both the categories have different strength and weaknesses. So it was observed that schema-based matching techniques perform good, when a higher quality schema level was available. On the other side instance-based are not vulnerable to such kind of data quality issues. Because they rather inspect the underlying instance the need a relatively high overlap between the different tables. It was observed that for dataset which have the same schema information but a disjunctive set of instances, the quality of results drops. Due to combining the best performing algorithm of both categories in one composite matcher a robust system was created, which performs on average the best over all three datasets with balanced Precision and Recall values.

Despite this first very promising results the problem of schema matching on large web table corpora is far from being solved. In order to further improve the results, more base matcher can be implemented using other techniques. Especially in the area of comparing schema elements based on their values, creating more advanced techniques should be considered in order increase their performance.

In addition to that the combination mechanisms for different matchings presented in this seminar paper are very basic. Therefore a big improvement potential lies in this are as well. For instance a weight-based voting mechanism or other Meta learning techniques, could be used in order get better results out of ensemble of individual matching algorithms. Another aspect that needs to be tackled is the analysis of the runtime of the implemented system. Since especially a usage in a live scenario is considered a high performance is crucial for user acceptance.

Although this improvable facts, the system designed in this seminar paper allows robust and qualitative matching of different table schemata originated from web tables in order to build an integrated schema. This will help potential users to yield as much as information as possible out of relational tables originating from the web.

References

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

[2] http://webdatacommons.org/.

[3] http://webdatacommons.org/webtables/index.html.

[4] Satanjeev Banerjee and Ted Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 136–145. Springer Berlin Heidelberg, 2002.

[5] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, December 1986.

[6] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. Generic schema matching with cupid. Technical Report MSR-TR-2001-58, Microsoft Research, August 2001.

[7] Alexander Bilke and Felix Naumann. Schema matching using duplicates. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 69–80. IEEE, 2005.

[8] Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

[9] Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. Data integration for the relational web. *Proc. VLDB Endow.*, 2(1):1090–1101, August 2009.

[10] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, August 2008.

[11] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, volume 3, pages 73–78, 2003.

[12] William W Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *ACM SIGMOD Record*, volume 27, pages 201–212. ACM, 1998.

[13] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. Finding related tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 817–828, New York, NY, USA, 2012. ACM.

[14] Hong-Hai Do and Erhard Rahm. Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 610–621. VLDB Endowment, 2002.

[15] Songyun Duan, Achille Fokoue, Oktie Hassanzadeh, Anastasios Kementsietsidis, Kavitha Srinivas, and Michael J Ward. Instance-based matching of large ontologies using locality-sensitive hashing. In *The Semantic Web–ISWC 2012*, pages 49–64. Springer, 2012.

[16] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2nd edition, 2013.

[17] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Language, speech, and communication. MIT Press, 1998.

[18] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

[19] Bernardo Cuenca Grau, Zlatan Dragisic, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Roger Granada, Valentina Ivanova, Ernesto Jiménez-Ruiz, Andreas Oskar Kempf, Patrick Lambrix, et al. Results of the ontology alignment evaluation initiative 2013. In *Proc. 8th ISWC workshop on ontology matching (OM)*, pages 61–100, 2013.

[20] Toni Gruetze, Christoph Böhm, and Felix Naumann. Holistic and scalable ontology alignment for linked open data. In *LDOW*, 2012.

[21] Prateek Jain, Pascal Hitzler, Amit P Sheth, Kunal Verma, and Peter Z Yeh. Ontology alignment for linked open data. In *The Semantic Web–ISWC 2010*, pages 402–417. Springer, 2010.

[22] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[23] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings*

of the 5th Annual International Conference on Systems Documentation, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[24] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[25] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *VLDB*, volume 94, pages 12–15, 1994.

[26] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[27] Sabine Maßmann, Salvatore Raunich, David Aumüller, Patrick Arnold, and Erhard Rahm. Evolution of the coma match system. In *OM*, 2011.

[28] Erhard Rahm. Towards large-scale schema and ontology matching. In Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, editors, *Schema Matching and Mapping*, Data-Centric Systems and Applications, pages 3–27. Springer Berlin Heidelberg, 2011.

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

[30] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[31] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[32] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *Proc. VLDB Endow.*, 5(3):157–168, November 2011.

[33] Mikalai Yatskevich and Fausto Giunchiglia. Element level semantic matching using wordnet. In *Meaning Coordination and Negotiation Workshop, ISWC*, 2004.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den 5.08.2014 Unterschrift