Generalizing Matching Knowledge Using Transfer Learning

Master Thesis

presented by
Alexander Brinkmann
Matriculation Number 1540241

submitted to the
Data and Web Science Group
Prof. Dr. Christian Bizer
University of Mannheim

January 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Setting

Due to the constantly growing amount of data, organisations drown in data. Naturally, many sources for this data are disjoint, because the data originates from several different sources in the organisation. Integrating these different data sources is inevitable to gain new business value or scientific insights [73]. Hence, industry and academia strive for solutions to improve data integration. A key step of the data integration process is finding the same real-world entities in different separated data sets, a task called Identity Resolution [81]. As approaches like rule-based systems do not scale with growing data complexity, supervised ML becomes more popular in academia and industry to solve the Identity Resolution task [64, 70]. The main issue of supervised ML in the context of Identity Resolution is the lack of training data [70].

In this thesis it is examined if it is possible to learn a generalizing matching rule for a knowledge base class, which enhances a class of an existing knowledge base like DBPedia [40]. Thus, whenever a new record from any data source is matched to a specific class of a knowledge base, this previously trained ML model is reused and transferred for Identity Resolution. Thereby, training a new supervised matching ML model for each class in an unknown data set becomes obsolete if a trained model for this class is attached to the corresponding knowledge base class. Consequently, domain experts do not have to annotate training data, which contains numerous examples for matches and non-matches between records, to learn a corresponding ML model for each class in a new data set. Hence, their efforts in a data integration process are reduced, which is crucial as time of domain experts is usually short and expensive.

1.2 Goals & Contributions

The main question of this thesis is, whether it is possible to learn a general matching rule for a Knowledge Base (KB) class. To obtain requirements for a general matching rule, different semantic topics are created. These topics contain data sets that shall be matched to a central KB. To solve the matching tasks different matching rules are learned. If these matching rules are transferable within the topic, it is possible to derive requirements for a generalising matching rule from the transferred matching rules. Along this process, three detailed research question (RQ) are asked to find requirements for a general matching rule:

- **RQ1**: What is the performance of the learned matching rules?
- **RQ2**: To which extent can entire matching rules be transferred?
- **RQ3**: Is it more beneficial to transfer only parts of the matching rule instead of the complete rule?

This thesis contributes a standardised experimental setup to analyse the performance of learned and transferred matching rules. This includes a framework to asses the importance of different parts of a matching rule. Using this standardised setup, it is observed that the naive transfer of matching rules and matching rule parts can be beneficial. In one case, the transferred matching rule outperforms the baseline's F score by 0.131. Furthermore, an algorithm to generate configurable silver standards for Identity Resolution tasks is presented. Hence, this algorithms influences the difficulty of a training set to improve the quality of a matching rule.

All data sets, conducted experiments and implementations are published on GitHub[1].

1.3 Thesis Structure

First, Chapter 2 introduces the fundamentals of Identity Resolution. This introduction includes learning matching rules, because these matching rules are transferred later on. Transfer Learning is described in Chapter 3. In Chapter 4 the experimental set up is introduced. This introduction in particular consists of explanations of the silver standard creation process, the used learning algorithms and the metrics to evaluate the research questions. The topics used in the experiments are presented in Chapter 5. In Chapter 6 results of the conducted experiments are evaluated. These results are further discussed in Chapter 7. In Chapter 8 all results are summed up and future work is proposed.

[1]https://github.com/abrinkmann/GMKUTL

Chapter 2

Identity Resolution

Identity Resolution describes the process of finding records from different databases referring to the same real-world entities. To find these records, matching rules are generated. As the goal of this thesis is to find generalised matching rules to ease the integration of new data into a KB, Identity Resolution is at the heart of this thesis. Consequently, this chapter introduces Identity Resolution in general to lay the foundations for the following chapters.

Identity Resolution is first step in the context of integrating data in general. Afterwards, a general approach towards Identity Resolution is given. This approach is taken from the literature and concludes with an overview of related systems.

This chapter is organised as follows. Section 2.1 sets Identity Resolution into the context of data integration in general. Then, section 2.2 defines Identity Resolution and illustrates general approaches towards solving the problem. The following sections 2.3, 2.4, 2.5 and 2.6 introduce an Identity Resolution framework found in the literature. This Identity Resolution framework covers the topics indexing, similarity measures, matching rules and evaluation. In the end, section 2.7 presents different related systems.

2.1 Challenge

Given the continuously growing amount of collected data by companies, governments and individuals, the need for effective tools to process this data is constantly increasing. Organizations drown in data. They aim to analyse this data instantaneously to receive a competitive advantage, to improve their productivity or to identify suspicious criminal activities [11]. Hence, academia and industry focus on research and development of sophisticated data warehousing strategies to store data in a clean, consistent and persistent manner. A main focus of this research is put on the improvement of an efficient application of data mining techniques to extract knowledge from the collected raw data [24].

To enrich data quality and to mine for patterns, which do not occur by analysing a single dataset, different data sources are integrated in many data mining projects. For example the linkage of cancer patients with Geo coordinates allows scientists to draw a connection between different regions and certain types of cancer. Thereby, scientists can better identify sources for cancer[11]. In general, the integration of data sources within an organization or across multiple organizations can accelerate the success of a data mining project [11]. Data ntegration includes the following three tasks:

- *Schema Matching* is done by determining corresponding elements in the matched table schemata. This involves a semantic mapping to specify how the different dataset schemata can be set into context, so that database tables, attributes and conceptional structures that belong to the same type of information are identified [8, 12].

- *Identity Resolution* describes the process of finding records from different databases, which refer to the same real-world entities [52]. These records are inter-source duplicates. The opposite are intra-source duplicates describing duplicate records within a single data source. This task is called *Duplicate Detection* [54]. In the literature the problem is addressed by terms like data matching [12, 16], record linkage [81], data reconciliation [14], entity identification problem [24], entity resolution [34] or entity matching [33].

- *Data Fusion* concludes the data integration by merging groups or pairs of records into a single record to form a consolidated picture of a real-world entity. Therefore, semantic uncertainties and contradictions need to be resolved, which arise from the data's imperfection and diversity. The data fusion algorithm has to decide for example, whether the average of two values is preferred over the maximum of these two values. Thus, conflict resolution functions can be created and applied [31].

This thesis focuses on the Identity Resolution step in the data integration pipeline. Formally the Identity Resolution is defined as follows. Two relational database tables X and Y are given. A record $x \in X$ *matches* a record $y \in Y$ if they refer to the same real-world entity. The goal of Identity Resolution is to find *all* matches between A and B[16].

As a prerequisite for Identity Resolution, it is assumed that the data is presented in structured and well-formed files or database tables. This way no further Information Retrieval related steps need to be carried out to tune the data [12]. However, a proper pre-processing to normalize and structure the data is important to improve the success rate of the matching process[60].

The prepared data sources might list an individual several times in a customer database, a single product might occur in several online catalogues or the resulting list of a web search might contain a document multiple times [25, 5]. These duplicate entries can lead to customer dissatisfaction, poor performance indicators and increased cost. The goal of Identity Resolution is to resolve these duplicates into single entries, which increases the data quality and thereby satisfies the stakeholders' needs [54].

Finding the entities to be resolved is a non-trivial task if no unique entity identifiers or keys are available. In case there are unique keys at hand, SQL statements can be used to efficiently cover this problem by joining the entities via identifiers. Even when these keys exist, they must be accurate, complete, robust and consistent over time. Otherwise wrongly matched records arise. If no entity identifiers are present fulfilling the described characteristics, the matching algorithm must rely on the available common attributes across databases [12, 54]. Therefore, dedicated algorithms are necessary to tackle Identity Resolution, which will be described in section 2.2.

2.2 Approaches

The most convenient way to find matches across relational tables, is to compare each record of a table X with all records from another table Y. This leads to a quadratically growing computational complexity of the matching algorithm. Simultaneously, the number of positive matches grows only linearly with each new record in either X or Y. This holds under the assumption that no duplicates occur in the two compared relational tables. Therefore, Identity Resolution approaches are necessary softening this problem by focusing on relevant candidate record-pairs for comparison. The reduced set of records can be compared in detail and classified into the groups *match* and *non-match* [12].

The first step focusing on relevant candidate record-pairs is called indexing [12]. In the indexing stage, a set of candidate record-pairs is selected. Thus, the algorithms can reduce the computational complexity arising from evaluating all possible record-pairs. During the second stage, the candidate record-pairs are classified by applying a matching rule, whereby matching record-pairs are identified. The list of candidate record-pairs is consolidated so that a 1:1 relationship between the data sources is established [5, 12].

The matching rule is generated to classify candidate record-pairs as *match* or *non-match* based on a comparison vector containing similarity measures of the candidate record-pairs' attributes. These matching rules can be hand-crafted or learned from training data.

2.3 Indexing

The goal of indexing is to reduce the number of comparisons by excluding as many non-matching pairs as possible without compromising the result's completeness [9]. In this section the two indexing techniques *Pairwise Comparison* and *Inverted Index* are introduced. Other approaches exist, which are more suitable for data with more complex relationships or include clustering algorithms. But these complex indexing approaches are not relevant in the context of this thesis [54]. To evaluate the different indexing techniques, the *reduction ratio* is introduced.

2.3.1 Pairwise Comparison

A common approach in the literature is to use a Pairwise Comparison algorithm for this step. Pairwise Comparison algorithms partition the records into smaller blocks of records. Therefore, a blocking key value is generated by a domain expert or as a result of the initial data exploration. The purpose of this blocking value is to select the records belonging to one block of records. A block of records is referred to as a partition. All comparisons among candidate record-pairs are conducted within one partition based on the assumption that duplicate records occur within the same partition. Since it defines the partitions in which Identity Resolution is executed, data partitioning is crucial to these algorithms [12].

The *Pairwise Comparison* technique depends on the blocking key value, because it may initially separate matching records, which compromises the Identity Resolution's result. Therefore, multiple blocking key values can be applied in several runs to overcome this issue [54].

Standard Blocking

One famous Pairwise Comparison Algorithm called *Standard Blocking* uses the blocking key value to generate disjoint subsets of the given dataset. These subsets are used to create candidate record-pairs. This leads to the reduction of comparisons [54]. One drawback of this method is the dependence of the size of the generated blocks on the frequency distribution of the blocking key value. Therefore, it is difficult to predict a block's size and the corresponding number of candidate record-pairs [12].

Sorted-Neighbourhood

The *Sorted-Neighbourhood* method is another frequently cited Pairwise Comparison Algorithm with a fixed block size. It uses the blocking key value to sort the dataset. Afterwards, a window of a fixed number of records is slid across the dataset and the first entry is compared to all other entries in the current window[12]. The fixed number of records comes with the disadvantage that one window may not cover all records with the same blocking key values [54]. The *Adaptive Sorted-Neighbourhood* method solves this problem by adapting the window size based on the similarity of records. When the next record is still similar to the first one of the former window, that window's size is increased to cover both records in one comparison run. Thereby, more similar entities can be covered [83].

2.3.2 Inverted Index

Assuming that the blocking key value is a string, approaches from the field of information retrieval like the inverted index can be applied. For the creation of the inverted index the blocking key value is tokenized. In the literature, different approaches exist, such as the tokenization by non-alphanumeric characters or the generation of q-grams (substrings of length q) [49, 9]. Subsequently, the created inverted index, can be used to calculate a similarity value among all indexed record-pairs. For the calculation of this similarity value measures like Jaccard similarity or overlap similarity can be used. In 2.4 different similarity measures are introduced. To identify relevant record-pairs, a similarity threshold can be applied. So, only record-pairs being above the chosen similarity threshold are kept as candidate record-pairs [49].

The *Inverted Index Technique* depends on the blocking key value, because it may initially separate matching records, which compromises the Identity Resolution's result. Therefore, multiple blocking key values can be applied in several runs to overcome this issue [54].

2.3.3 Evaluation

Besides measuring the quality of a matching rule, evaluating the efficiency and effectiveness of the Identity Resolution technique is also interesting. The *Reduction Ratio* (rr) measures the relative reduction of the comparison space accomplished by an indexing technique [18]:

$$\text{rr} = 1 - \frac{\text{size of the reduced comparison space}}{\text{size of the initial comparison space}} \tag{2.1}$$

The *Pairs Completeness* (pc) measures the relative number of matching pairs contained in the reduced comparison space resulting from an indexing technique [18]:

$$\text{pc} = \frac{\text{number of matching pairs in reduced comparison space}}{\text{number of matching pairs in entire comparison space}} \tag{2.2}$$

2.4 Similarity Measures

In this section similarity measures are introduced in general. So, the usage of similarity measures is set into context to the Identity Resolution process. Then, the relation of similarity measures and distance measures is defined. Afterwards, the six main categories for similarity measures are introduced. In the end of this section, the idea of applying pre-processing steps before measuring the similarity of two attributes values is explained.

2.4.1 Context of Identity Resolution

Let X and Y be two relational tables with an aligned schema $\{A_1, A_2, ..., A_n\}$. An intermediate step in Identity Resolution is to define a notion of similarity between a record $x \in X$ and a record $y \in Y$. The similarity of two records x and y is defined by a comparison vector $\vec{s}$. In this comparison vector $\vec{s}$ the similarity scores of all similarity measures applied to the attributes values of the two records are collected [12]:

$$\vec{s} = [sim_1(x[A_1], y[A_1]), sim_2(x[A_1], y[A_1]), ..., sim_m(x[A_n], y[A_n])] \tag{2.3}$$

Based on this comparison vector $\vec{s}$, a matching rule can decide whether a candidate record-pair is classified as *match* or *non-match* [9]. So, the chosen similarity measures are crucial to derive the similarity of two records. Additionally, the calculation of comparison vectors $\vec{s}$ needs to be the same for all candidate record-pairs to allow for consistency [12].

2.4.2 Similarity and Distance Score

Similarity scores are normalized between 0.0 and 1.0 to compare scores across different similarity measures. When applying a similarity measure, a normalized similarity score can be set in context to a normalized distance or cost measure as follows [16]:

$$\mathit{sim}(x[A_i], y[A_i]) = 1 - \mathit{dist}(x[A_i], y[A_i]) \tag{2.4}$$

2.4.3 Categories

The wide range of existing similarity and distance measures can be grouped into six main categories:

- *Exact* measures determine if two values are exactly the same. So, an exact measure returns either 0.0 or 1.0 [12].

- *Token-based* measures like Jaccard and TF-IDF Cosine similarity first divide a string value into tokens based on a tokenization. N-gram models can be considered for the tokenization. These substrings of the original string are then compared to the set of tokens from another string. Thereby, token-based similarity measures are less sensitive to word ordering than, similarity measures using the whole string [54].

- *Edit-based* measures like Levenshtein, Jaro, Hamming or Smith-Waterman distance consider the string as a whole and compute the distance between two strings based on the number of insertions, deletions and substitutions of characters necessary to transform one string into the other string [43, 54].

- *Hybrid* measures like Monge-Elkan combine tokenization and edit-based similarity to calculate a final similarity score [54].

- *Phonetic* encoding techniques like Soundex or Metaphone convert a string into a code considering the string's pronunciation. Afterwards, these newly created codes can be compared [10].

- *Type-specific* measures depend on the context in which they are applied, such as numerical comparisons, data and time comparisons or geographical distances [12].

2.4.4 Pre-processing

Blindly applying the similarity measures may distort them, because the attribute values may be in different formats. Therefore, some pre-processing before applying the similarity measures may be useful to improve the matching quality. Such pre-processing can be lower casing strings, removing unwanted substrings (e.g., values in brackets) or bringing the value into a unified format like for date values. The application of such pre-processing steps depends on the context of the task[12].

2.5 Matching Rules

In this section the generation of matching rules based on a comparison vector for candidate record-pairs is explained. First a general overview of the task is given, followed by the four different classification approaches: Threshold-Based, Rule-Based, Probabilistic and Supervised Learning.

2.5.1 Overview

The main goal of Identity Resolution is to classify candidate record-pairs into the classes *match* and *non-match*. In some cases the task is enhanced by a third class called *potential match*. This class contains candidate record-pairs, for which a certain level of confidence to classify the candidate record-pairs as *match* or *non-match* is not exceeded. Therefore, a human annotator decides whether a *potential match* is a *match* or *non-match* [12]. In the context of this thesis, candidate record-pairs are only classified into the classes *match* and *non-match*. To classify a candidate record-pair according to the given task a matching rule is created [12].

2.5.2 Threshold-Based Classification

The immediate way to generate such a matching rule is to sum up the values of a comparison vector. Once the sum of all similarity values exceeds a certain threshold, the candidate record-pair is classified as a *match*. The threshold can be set manually or can be learned if the necessary training data is available.

A drawback of this method is that it does not consider the importance and the discriminative power of each individual similarity score. To overcome this drawback, weights for each similarity score can be introduced. Thereby, the similarity scores are combined into a linear combination. This manual approach often requires a domain expert to set the weights accurately [12].

2.5.3 Probabilistic Classification

Besides using domain knowledge, a probabilistic approach can be found in the literature to determine these weights. The authors of this approach consider two populations X and Y with record-pairs in XxY, whereby each record-pair has a comparison vector. Conditioned on these comparison vectors, the probability whether a candidate record-pair is a *match* or a *non-match* is calculated. Under the assumption that this calculated probability is conditionally independent from the different attributes used to generate the comparison vector, weights for each attributes are calculated. Even though conditional independence is violated in real world cases, the probabilistic approach offers valid results [19, 26, 80].

2.5.4 Rule-based Classification

The downside of summing up the similarity values is that information like the variance of similarity scores or dependencies among the attributes is lost [12]. Rule-based approaches take this information into account by building rules based on similarity values or combinations of similarity values to classify a candidate record-pair. Therefore, a set of rules consisting of a condition and a consequence is created. A high coverage and accuracy of each rule is preferable. This means that a rule covers as many candidate pairs as possible and classifies them correctly. Thus, these rules need to be engineered in a precise fashion.

One approach to create such rules is to involve a domain expert, who creates these rules manually [24, 45].
Creating rules by hand is tedious, thus in recent years ML has been applied to overcome this challenge [12, 16]. In fact, Köpcke et al. [36] show in their study that most supervised learning approaches for Identity Resolution outperform non-learning approaches. By classifying candidate record-pairs into *matches* and *non-matches* the problem becomes a binary classification task. If training data is available, it is possible to obtain matching rules from a supervised classification model to label candidate record-pairs as *match* or *non-match* [9].

2.5.5 Supervised Learning Classification

Supervised learning approaches try to use all information available from the candidate record-pairs to learn a classification model. This information basically relies on a comparison vector generated from the similarity values between two candidate record-pairs. The comparison vector's values are used as features to train a matching rule model. After evaluating the matching rule, it can be used to label unknown candidate record-pairs [9].

Three phases are conducted to learn a supervised classification model before applying it to classify unknown candidate record-pairs [24]:

- In the *training phase* a classifier is built, which describes a predefined set of classes from a training set. In the context of Identity Resolution, the classifier learns a mapping from the comparison vectors of a candidate record-pair to the classes *match* or *non-match*.

- In the *validation phase* the built model's accuracy is estimated on a validation set. Thereby, the built model can be fine-tuned. Like in other ML contexts, this can be done by cross-validating the model.

- During the *testing phase* the classifier is evaluated on a test set by classifying the test candidate record-pairs and measuring the result's quality.

To avoid overfitting, the training, validation and test set have to be disjoint from each other[24].

A major challenge supervised classification faces in terms of Identity Resolution is the common imbalance of the set of candidate record-pairs. This is due to the fact that potentially more *non-matches* exist than *matches* among the candidate record-pairs. Even though indexing tries to reduce the number of *non-matches* initially. Therefore, different sampling can be beneficial to balance the two classes and to generate more accurate matching rules [9].

Furthermore, obtaining the training data is time-consuming when done manually. As an alternative, active learning has been proposed to create training data interactively. Instead of an effortful labelling of random candidate pairs, the user is requested to label only the most informative candidates. Therefore, the classifier learns a model from a small labelled training set and asks the user to label one or more selected instances. Afterwards, the newly classified instances are added to the training set and the classifier is trained in a regular supervised fashion [67, 29].

To keep the focus on transfer learning, active learning is not further considered in this thesis.

In recent years deep learning has become a major direction in ML research [21, 66, 38]. Recent studies suggest that Identity Resolution can benefit from deep learning as well [53]. Unfortunately, the importance of different features and parts of the learned model is hard to explain [44, 30]. However, understanding the importance of different features and parts of a learned model are essential to answer the research questions in the context of this thesis. Accordingly, deep learning models are not further covered.

2.6 Evaluation

Having a mechanism to label candidate record-pairs as *match* or *non-match*, the quality of this classification needs to be evaluated. For this evaluation a ground truth, also known as gold standard, is necessary. The characteristics of such a gold standard are introduced. Via the gold standard, the quality of the classification and the underlying matching rule can be evaluated using different quality metrics as introduced in this section.

2.6.1 Gold Standard

The gold standard contains *matches* as well as *non-matches*. If a record-pair is labelled as *match* in the gold standard, the two records describe the same real world entity. Otherwise, if a record-pair is labelled as *non-match* in the gold standard, the two records describe different entities in the real world. Thereby, the gold standard reflects the characteristics of the matched data [12].

A possible way to obtain a gold standard is to manually label candidate record-pairs [12]. Creating the ground truth manually is very tedious and time-consuming. In some cases even for a human annotator it may be difficult to decide if two records refer to the same real world entity. For example, the first record describes variations of an entity. If this variation is not covered by the second record of the record-pair, the record-pair has to be classified as *non-match*. Due to the difficulty of detecting such cases, a gold standard may contain wrongly classified record-pairs. To ensure a good quality, multiple human annotators can annotate the data and an inter-annotator agreement guarantees the quality of the gold standard. The agreement depends on the data ambiguity, skills of the annotators and the task at hand [57]. Nevertheless, a gold standard is the best available data corpus to evaluate the quality of a matching rule [12].

As a lot of data can be beneficial, a ground truth can be automatically created from different legacy ground truths. Such a data corpus is not fully checked by a human annotator. Therefore, this data corpus is called silver standard instead of gold standard. Rebholz-Schumann et al. [63] evaluate the performance of such a silver standard against a gold standard in the context of their CALBC[1] project. It turns out that a drop in recall between the model learned on the gold standard and the model learned on the silver standard is observed. The reason for this drop is that some annotations of *matches* are missing in the silver standard. Concluding from this finding, the substitution of a gold standard by a silver standard is only sensible for applications that do not require a high recall [63, 82].

2.6.2 Quality Metrics

As soon as a gold standard is available the evaluation can be treated like any other classification problem [12]. Based on the gold standard, the labelled candidate records can be sorted into a confusion matrix shown in table 2.1 [13].

		Classification	
		match	non-match
Actual	match	True positive (TP)	False negative (FN)
	non-match	False negative (FN)	True negative (TN)

Table 2.1: Confusion Matrix for Record-Pair Classification

Based on the confusion matrix from table 2.1 different quality measures can be calculated:

- *Precision* (Pr) or *Pairs quality* in the context of Identity Resolution techniques shows the proportion of candidate record-pairs, which are correctly classified as a *match* against all record-pairs classified as a *match*[13]:

$$\text{prec} = \frac{TP}{TP + FP} \tag{2.5}$$

- *Recall* (Rec) or *Pairs completeness* in the context of Identity Resolution techniques shows the proportion of candidate record-pairs, which are correctly classified as a *match* against all matching record-pairs in the dataset[13, 18]:

$$\text{rec} = \frac{TP}{TP + FN} \tag{2.6}$$

- *f-measure* is the harmonic mean of Precision and Recall, which helps to find the best compromise among these two measures[13]:

$$\text{f-measure} = \frac{2 * Pr * Rec}{Pr + Rec} \tag{2.7}$$

[1]https://www.calbc.eu/

Other frequently used quality measures like Accuracy, Specificity and False Positive Rate are not suitable to evaluate Identity Resolution tasks. They include True Negatives, which dominate the formula due to their high number. The reason for the high number is that it represents the amount of non-matching pairs. Furthermore, ROC curves suffer from a comparable problem. ROC curves plot the True Positive Rate against the False Positive Rate, which considers the high number of True Negatives and thereby becomes small. This way an over-optimistic ROC curve is calculated[13].

2.7 Related Systems

Different representative approaches for learning matching rules can be found in the literature. Konda et al. [32] provide a comprehensive study of different Identity Resolution systems. This study is used to present and cluster these systems in this section.

This section is organised as follows. In section 2.7.1 the related systems are presented as well as the dimensions used to compare them. Following sections introduce the different related systems in detail.

2.7.1 Overview

Table 2.2 illustrates a reduced set of related systems presented by Konda et al. [32]. The chosen dimensions are the dimensions, which can be related to the Identity Resolution process described in this thesis. These dimensions show the following:

- The systems focus on the scenarios single table and two tables matching.
- A wide range of different well-known indexing techniques is provided.
- The systems provide a variety of matching approaches with a focus on rule-based matching approaches.
- Apart from Magellan all presented approaches are implemented in Java.

These dimensions reveal that the different systems are related to each other. In the subsequent sections a focus is put on the main differences across the related systems.

2.7.2 DuDe

Draisbach et al. [17] propose the Duplication Detection Toolkit (DuDe). DuDe's main purpose is to provide a general extendable framework for duplicate detection

Name	Scenario	Indexing	Matching	Language
DuDe	Single table, Two tables	Standard Blocking, Sorted-Neighbourhood	Rule-based	Java
Magellan	Two tables	Standard Blocking	Rule-based	Python
SILK	RDF Data	Standard Blocking	Rule-based	Java
TAILOR	Single table, Two tables	Standard Blocking, Sorted-Neighbourhood	Probabilistic, Rule-based	Java

Table 2.2: Characteristics of Identity Resolution Systems [32]

along with different data sets and gold standards. Hence, experiments from different researchers can be compared via DuDe.

The Identity Resolution process follows the building blocks. For the Identity Resolution process, the main building blocks are *indexing*, *comparators* and *postprocessor*. So, it is possible to choose a suitable *indexing* techniques like Standard Blocking or Sorted-Neighbourhood. Afterwards, *comparators* with different preprocessing steps and similarity measures can be implemented to form a comparison vector. Based on this comparison vector, thresholds can be manually set to determine whether a candidate record-pair is labelled as *match* or *non-match*. The *postprocessor* mainly calculates the transitive closure for all record-pairs labelled as *match*. These building blocks can be adapted and enhanced by the user depending on the requirements at hand[17].

Unlike all other approaches, DuDe focuses on a manual rule-based approach to classify record-pairs.

2.7.3 Magellan

Konda et al. [33] propose Magellan. Magellan is an Identity Resolution system that seeks to cover the whole Identity Resolution process. Magellan tries to make use of the existing Python open-source data science ecosystem.

Like in all other approaches, Magellan contains different building blocks that can be adapted depending on the context. Different indexing strategies like Sorted-Neighbourhood can be used. The main difference to the other presented systems is that Magellan offers functionalities to debug and to analyse the blocker in depth. Thereby, users can evaluate the most suitable indexing strategy.

A subset of the blocked record-pairs is now labelled by a user and used as training data for the following supervised classification. For the classification of record-pairs as *match* or *non-match*, Magellan relies on the ML capabilities of the existing Python open-source data science ecosystem. Hence, the user can choose among a wide range of algorithms to solve the matching problem.

Magellan is unique in the sense that the whole Identity Resolution process is built around the Python open-source data science ecosystem. Concluding, Magellan offers a lot of freedom to the user, when it comes to the design of the Identity Resolution process.

2.7.4 SILK

Volz et al. [76] propose SILK. SILK is a web-application developed in Java and Scala that can be used by Linked Data publishers and consumers to generate a set of RDF links. Instead of integrating tables, SILK aims at linking RDF instances from the web with local known instances.

Isele et al.[29, 28] enhance SILK by using an active learning approach combined with genetic programming to deal with Identity Resolution. This active learning approach is called ActiveGenLink. To perform well on unknown data, training data for supervised learning algorithms needs to include all relevant corner cases. Therefore, ActiveGenLink learns a matching rule by asking the user whether a set of candidate record-pairs is a *match* or a *non-match*. These candidate record-pairs are actively selected via the algorithm ActiveGenLink, because these candidate record-pairs provide the highest information gain for manual verification.

The creation of matching rules is done by the algorithm using genetic programming. The algorithm uses specialized crossover operators to derive matching rules. In contrast to all previously presented approaches, GenLink selects the attributes of the records, which are compared as well as the applied similarity measures and the used thresholds by itself. Additionally GenLink includes chains of data transformation to normalize the data upfront. Thereby the user's effort in the Identity Resolution process is reduced to the task of confirming or rejecting a few candidate record-pairs. Isele et al. show that ActiveGenLink provides accurate matching rules by only using a small amount of manually labelled candidate record-pairs[29, 28].

SILK's approach towards Identity Resolution is different from all other approaches, because active learning and genetic programming are used.

2.7.5 TAILOR

Elfeky et al. [18] propose interactive Record Linkage Toolbox (TAILOR). TAILOR offers users the opportunity to implement their own Identity Resolution process by tuning system parameters. Elfeky et al. [18] demand that they are the first ones to implement such a toolbox for Identity Resolution. In their process, first indexing is used to select appropriate candidate record-pairs. For indexing, standard blocking and sorted neighbourhood blocking are offered to the user. Then comparison vectors are created by comparing all attributes from the two candidate record-pairs with the similarity functions Hamming, Smith-Waterman, Jaro, a combination of n-gram models and Jaro, as well as Soundex. Afterwards, a subset of the candidate record-pairs is labelled as *match* or *non-match* by applying domain knowledge to receive labelled data. This labelled data is split into a training and a test set.

For the following matching step, users can chose among the matching algorithms, probabilistic classification and a ML approach. More precisely, the ML approach is a decision tree that learns to label new record-pairs as *match* or *non-match* via the mentioned training set. Elfeky et al. show that the decision tree outperforms the probabilistic classification regardless of the percentage of training data used to train the ID3 Decision Tree[18].

When comparing TAILOR to the other approaches, it is possible to see that the functionality is limited to a set of chosen approaches for indexing and matching. Unlike the other system TAILOR is not available under the open source licence. Nevertheless, TAILOR is one of the first systems that support the Identity Resolution process.

Chapter 3

Transfer Learning

A major problem of applying supervised learning for Identity Resolution is the lack of training data [12, 29]. To overcome the problem of missing training data, transfer learning has become popular within the ML community in recent years [58, 79, 71]. The goal is to reuse existing training data[58].

This chapter introduces transfer learning in general. This covers the underlying ideas of transfer learning, as well as different approaches found in the literature for transfer knowledge across ML models. Thereby, a focus is set on domain adaptation, which is a sub-setting of transfer learning. Domain adaptation is the setting used in this thesis for improving matching rules.

This chapter is organised as follows. Section 3.1 introduces the idea of transfer learning in general including specific notations. Based on these notations section 3.2 illustrates different transfer techniques. Section 3.3 gives an overview of the problem of negative transfer and transfer bounds. Coming from the transfer techniques and transfer bounds, section 3.4.2 presents approaches suitable for transferring knowledge in the context of Identity Resolution. Section 3.5 concludes this chapter with an overview of related systems that combine Identity Resolution with transfer learning.

3.1 Foundations

Traditional ML techniques try to learn the given task from scratch. Transfer learning in contrast utilizes existing knowledge from a source task to improve the performance on a target task [58, 79]. To analyse the different transfer learning techniques in depth, it is necessary to introduce some notations. These notations are taken from Pan et al. [58] and used throughout the whole thesis.

According to Pan et al. [58] the first two components are the *domain* and the *task*. A *domain* $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where X=$\{x_1, ..., x_n\} \in \mathcal{X}$. In the scenario of Identity Resolution, $\mathcal{X}$ defines the comparison feature space. Following from a *domain* being $\mathcal{D}$ = $\{\mathcal{X}$,$P(X)\}$, a *task* $\mathcal{T}$ is defined by a label space $\mathcal{Y}$ and a function $f(\cdot)$. Consequently, a *task* $\mathcal{T}$ is denoted by $\mathcal{T}$ = $\{\mathcal{Y}, f(\cdot)\}$. In the context of this thesis, the label space $\mathcal{Y}$ corresponds to the set of label being True for match or False for non-match. The matching function $f(\cdot)$ can be learned from the training data, which contains pairs x_i, y_i, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. To predict the corresponding label for $f(x)$ of a new instance x the function $f(\cdot)$ can be used. $f(x)$ can also be written as P($y|x$) to represent the related probabilities.

It is distinguished between *source domains* $\mathcal{D}_S$ and *target domains* $\mathcal{D}_T$. *Source domain data* is expressed by $\mathcal{D}_S$ = $\{\{x_{S1}, y_{S1}\}$,...,$\{x_{Sn}, y_{Sn}\}\}$, where $x_{Si} \in \mathcal{X}_S$ is the comparison feature vector of a record-pair in the source domain. $y_{Si} \in \mathcal{Y}_S$ represents the corresponding class label. Similarly, *target domain data* is expressed by $\mathcal{D}_T$ = $\{\{x_{T1}, y_{T1}\}$,...,$\{x_{Tn}, y_{Tn}\}\}$, where $x_{Ti} \in \mathcal{X}_T$ is the comparison feature vector of a record-pair in the target domain. $y_{Ti} \in \mathcal{Y}_T$ represents the corresponding class label [58].

Resulting from these notations transfer learning can be formally defined:

Definition 3.1.1 *Transfer Learning: Given a source domain $\mathcal{D}_S$ and a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target matching function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$ [58].*

Concluding from this definition of transfer learning and a domain being a pair $\mathcal{D}$ = $\{\mathcal{X}$,$P(X)\}$, $\mathcal{D}_S \neq \mathcal{D}_T$ implies that either $\mathcal{X}_S \neq \mathcal{X}_T$ or $P(X_S) \neq P(X_T)$. If $\mathcal{X}_S \neq \mathcal{X}_T$, transfer learning is defined as heterogeneous transfer learning. Throughout this thesis, the comparison features in *source domain* $\mathcal{D}_S$ and *target domain* $\mathcal{D}_T$ are equivalent. Consequently, this thesis focuses on homogeneous transfer learning, which means that $\mathcal{X}_S = \mathcal{X}_T$. $P(X_S) \neq P(X_T)$ expresses that the marginal probability distributions are different between *source domain* $\mathcal{D}_S$ and *target domain* $\mathcal{D}_T$. As the input data of *source domain* $\mathcal{D}_S$ and *target domain* $\mathcal{D}_T$ are different in this thesis, the marginal probability distributions are different as well [79, 58].

In the same way, concluding from this definition of transfer learning and a task being a pair $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, $\mathcal{T}_S \neq \mathcal{T}_T$ implies that either $\mathcal{Y}_S \neq \mathcal{Y}_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$. As the label space for any Identity Resolution task is either True meaning *match* or False meaning *non-match*, the label spaces of all presented tasks in this thesis are equivalent. $P(Y_S|X_S) \neq P(Y_T|X_T)$ implies that the conditional probability distributions are different between *source task* $\mathcal{T}_S$ and *target task* $\mathcal{T}_T$ [79, 58]. In the context of this thesis, it means that the same values of the comparison vector yield different results *source domain* and *target domain*.

If there exists some relationship between feature spaces of source and target domains, these domains are defined as *related*[58]. The main goal of this thesis is to identity if and how the feature spaces of the different domains are related. Please refer to Chapter 4 for further details on the goals of this thesis.

All notation used for transfer learning are aggregated in table 3.1.

Notation	Description	Notation	Description
$\mathcal{X}$	Input Feature Space	$P(X)$	Marginal Distribution
$\mathcal{Y}$	Label Space	$P(Y\|X)$	Conditional Distribution
$\mathcal{T}$	Predictive Learning Task	$P(Y)$	Label Distribution
Subscript S	Denotes Source	$\mathcal{D}_S$	Source Domain Data
Subscript T	Denotes Task	$\mathcal{D}_T$	Target Domain Data

Table 3.1: Comprehensive Overview of Transfer Learning Notations [79]

3.2 Transfer Techniques

According to Pan et al. [58] transfer learning can be categorized under three different settings: inductive transfer learning, transductive transfer learning and unsupervised transfer learning [58]. The categorization is based on different relations between source domain and target domain as well as source task and target task [58]. Table 3.2 provides a general overview of the different transfer learning settings.

Learning Setting		Source and Target Domain	Source and Target Task
Traditional ML		the same	the same
Transfer Learning	Inductive	the same	related
	Unsupervised	related	related
	Transductive	related	the same

Table 3.2: Relationship of Traditional ML & Transfer Learning Settings [58]

3.2.1 Inductive Transfer Learning

In the inductive transfer learning setting the tasks of source and target are different. Pan et al. [58] assume that some labelled data is available in the target domain of an inductive transfer learning setting. This labelled data from the target domain is necessary to induce a predictive model $f_T(\cdot)$. This predictive model $f_T(\cdot)$ can be applied in the target domain.

3.2.2 Unsupervised Transfer Learning

In the unsupervised transfer learning setting the source and target domain are different but related and the source and target task are different but related as well. So, no labelled data is available for neither the source nor the target domain. Algorithms applied in this scenario focus on clustering, dimensionality reduction and density estimation [58, 15, 78].

3.2.3 Transductive Transfer Learning

In the transductive transfer learning setting source and target task are the same while source and target domain are related but different. No labelled data is available in the target domain while a lot of labelled data is available in the source domain [58]. Within transductive transfer learning a further distinction is made. In the first sub-setting the feature spaces of source and target domain are different, $\mathcal{X}_S \neq \mathcal{X}_T$. In the second sub-setting the feature spaces of source and target domain are the same, $\mathcal{X}_S = \mathcal{X}_T$. Only the marginal probability distributions are different $P(X_S) \neq P(X_T)$ [58]. In the literature, the second sub-setting is referred to as Domain Adaptation [2], Co-variate Shift [68] or sample selection bias [85]. The term Domain Adaptation is used in this thesis.

3.3 Transfer Bounds and Negative Transfer

Negative transfer in the context of transfer learning happens when the transferred information from the source domain reduces the performance of the target learner [79]. In these cases transfer learning hinders the target learner, because *source domain* and *target domain* are too dissimilar [65]. To deal with this problem the transfer bounds have to be identified, such that no negative information is transferred from the source domain to the target domain[79, 58]. In general, Rosenstein et al. [65] show that the transfer of too dissimilar knowledge among source and target domain can hurt the performance of a ML model. The approach to detect these transfer bounds depends on the approach used to transfer information. Section 3.4.2 gives an overview of transfer approaches. The overview includes strategies to avoid negative transfer.

3.4 Transferring Knowledge

In the context of domain adaptation two approaches for transferring knowledge are presented by Pan et al. [58]: Transferring of the Knowledge of Instances and Transferring Knowledge of Feature Representations. Both approaches are shortly introduced in this section.

3.4.1 Transferring Knowledge of Feature Representations

In the context of Transferring Knowledge of Feature Representation the goal is to reduce the difference between the source and target domain. By reducing this difference, the goal is to avoid negative transfer. Blitzer et al. [6] proposes *structural correspondence learning* to induce correspondences among features from the labelled source domain and the unlabelled target domain. Out of the correspondences found, a set of new features is created and used to learn the new ML model.

3.4.2 Transferring Knowledge of Instances

The main motivation behind an instance-based transfer is importance sampling. Via importance sampling only selected labelled instances from the source domain are transferred to the target domain. Thereby, the impact of negative transfer is reduced.

The literature suggests that $P(X_S) \neq P(X_T)$ that is the reason for importance sampling in the described transfer learning scenario [58]. If $P(X_S) = P(X_T)$ a naive transfer of instances is possible. In this case, all labelled instances from the source domain are transferred to the target domain to learn a ML model[74].

Assuming $P(X_S) \neq P(X_T)$, the estimation of $\frac{P(x_(S_i))}{P(x_(T_i))}$ for each instance is beneficial to determine the importance of each instance for the target domain[58]. Various approaches can be found in the literature for the estimation of $\frac{P(x_(S_i))}{P(x_(T_i))}$. Zadrozny constructs classification problems that estimate $P(X_S)$ and $P(X_T)$ independently [85]. Huang et al. [27] match the means of the source domain data and the target domain data in a reproducing Kernel Hilbert Space (RKHS). From this RKHS, a kernel-mean matching algorithm is learned to estimate $\frac{P(x_(S_i))}{P(x_(T_i))}$.

In the context of this thesis, a naive transfer of instances is conducted. Answering the proposed research questions evaluates how importance sampling can further improve the performance results.

3.5 Related Systems

In the literature only little research is found on Transfer Learning regarding Identity Resolution. The following three presented systems combine Transfer Learning and Identity Resolution.

3.5.1 Transferring Matching Rules

Ngomo et al. [56] present a formal model for Transfer Learning on matching rules. The starting points for the general matching process are a source and a target domain. Two steps are carried out within the matching process: First, a matching rule is discovered to retrieve high-quality matches. Second, the matching rule is used to link the instances from the source and the target domain. Now it is explored how to reuse existing matching knowledge to learn new matching rules f_T. The general idea is that the more related the feature spaces $\mathcal{X}_S$ and $\mathcal{X}_T$ of source and target domain, the higher the probability f_S is a good initial value for f_T. By mapping $\mathcal{X}_S$ to $\mathcal{X}_T$ the two feature spaces are set into context, to identify similar record-pairs from the source and the target domain. Additionally, the accuracy of each task (f_S,Y_S) in the source domain is calculated to make use of the better solved tasks. The similarity of record-pairs and the accuracy of how a matching rule maps two records, is used as background knowledge. Now the match of two instances can be detected via $\mathcal{X}_T$ by using the background knowledge. So, for each possible new match the most similar and most valuable source tasks are used as background knowledge for the matching. As Ngomo et al. [56] determine the relatedness of source and target tasks, a measure for the relatedness of the overall tasks is given.

This measure of relatedness can be used to determine how useful the described approach in a specific scenario is. According to Ngomo et al. [56], the described approach can achieve a high reliability for the detection of matching rules.

3.5.2 TRANSFER

According to Negahban et al. [55], the reality of heterogeneous sources requires the number of labeled traning data to scale quadratically in the number of data sources for state-of-the-art pairwise Identity Resolution techniques. Otherwise precision and recall cannot be kept constant. This problem is addressed by TRANSFER. TRANSFER is a generic Transfer Learning approach, which is trained using fast convex optimization. The main idea is to share the structure of one Identity Resolution problem across Identity Resolution problems that have one data source in common. So, TRANSFER learns to match pairs of multiple data sources while adaptively sharing common patterns of data quality. For example, assume the data sources A, B and C come from the same topic. When learning matching rules between A and B and between A and C, the knowledge about the common data source A is leveraged. This leverage happens by simultaneous learning the two matching tasks via a linear classifier. The weights of the linear classifier are decomposed as follows:

$$w_{ij} = w_0 + w_i + \triangle_{ij}, \tag{3.1}$$

where the vector w_0 captures the general trend of the topic. For example, movies with the same casts are similar. w_i measures the trend of the particular data source induced. $\triangle_{ij}$ handels the specific deviation resulting from the match of two data sources. Using this encoded knowledge, the learned linear classifier matches pairs from B and C. Negahban et al. [55] show that TRANSFER significantly reduces the amount of necessary training data.

3.5.3 Reuse and Adaptation for Entity Resolution through Transfer Learning

Thirumuruganathan et al. [74] investigate how to improve the performance of Identity Resolution on a data set D_T with limited to no training data using Transfer Learning. In fact, selected labelled data and features from a related data set D_S are transferred to learn a matching rule for data set D_T. For the transfer, four scenarios are covered: Scenario 1 (Adequate, Nothing) is classified as the hardest, because the training data of D_S is used to build a classifier that satisfies the matching task of D_T. In scenario 2 (Adequate, Limited) the limited training data in D_T is augmented by training data from D_S. The difficulty in scenario 2 is to avoid negative transfer. For scenario 3 (Limited, Limited) a pooling approach is chosen to use the limited training data from D_S and D_T to learn a satisfying matching rule. Scenario 4 (Adequate, Adequate) is considered to be the easiest scenario, because a sufficient amount of training data is available in D_T to use traditional ML techniques. Still it is possible to make the matching rule of scenario 4 more robust by using available knowledge from D_S. The approach defined in this thesis is close to scenario 1, so a focus is put on this scenario.

To align D_S and D_T a feature space standardisation is advocated by Thirumuruganathan et al. [74]. The record-pairs of D_S and D_T are encoded into a standard feature space. For this encoding the Distributed Representation for Words (DR) also known as word embeddings are used. Therefore, the attribute boundaries of the data sources are omitted to represent each record as a sentence. Each word in this sentence is than mapped into a high dimensional vector. The distance between the vectors of two words determines the relationship of these two words.

Based on the standardised feature space, algorithms dealing with the four different scenarios are introduced. Since scenario 1 defines the scenario covered in this thesis, this scenario is described in depth. For scenario 1(Adequate, Nothing) two valid approaches are defined. These approaches are a naive transfer of all records from D_S to D_T or an importance sampling algorithm. For the importance sampling algorithm a classification problem is defined: In this problem, a ML model learns to distinguish records coming from D_S and D_T. The ML model used for this distinction is a logistic regression. If the the learned model cannot confidently decide whether a labelled record belongs to D_S or D_T, this record is close to the records in D_T. Hence, the labelled records, which are close to the records coming from D_T, are valuable for learning a matching rule in D_T. Along this idea, valuable records are higher weighted for learning the matching rule than records that can easily be assigned to D_S by the logistic regression model.

Overall the described importance sampling approach yields satisfiable matching performance compared to the baseline naive transfer. Thirumuruganathan et al. [74] show that their importance sampling is valuable in the context of Transfer Learning.

Chapter 4

Experimental Setup

The general question of this thesis is to evaluate to which extent it is possible to transfer matching rules between matching tasks within the same semantic topic. So, this chapter provides a standardised setup for a set of experiments that try to answer this general question. Such a standardised experimental setup allows for conclusions on the requirements of a generalised matching rule.

In this chapter the technical foundations for the experiments are explained. This includes the Identity Resolution process and the generation of training and test data. In the end, the different experiments are introduced as well as the measures used to interpret them.

This chapter contributes a standardised experimental setup to measure the performance of learned and transferred matching rules. This includes the assessment of the importance of different matching rule parts. Additionally, an algorithm to create silver standards for Identity Resolution problems with different levels of difficulty is contributed.

This chapter is organised as follows. Section 4.1 introduces Web Data INTEgRation Framework (WInte.r)[1], which provides general support for Identity Resolution. Section 4.2 explains the applied indexing steps. The silver standard creation algorithm is explained in detail in section 4.3. Section 2.4 demonstrates how the comparison vectors for candidate record pairs are calculated. The learning algorithms for the matching rules are introduced in section 4.5. Based on these foundations, the standardised experimental setup is presented in section 4.6. Section 4.7 explains how the experiments are interpreted and linked to the research questions.

[1]https://github.com/olehmberg/winter

4.1 Execution Environment - WInte.r

As a basis for the implemented Identity Resolution and transfer learning experiments, WInte.r[2] is chosen. WInte.r is a framework for end-to-end-data integration, which provides methods for data pre-processing, schema matching, Identity Resolution, data fusion, and result evaluation [41]. For the purpose of this thesis, WInte.r is enhanced to allow the transfer of learned matching rules among similar domains. So, a matching rule is learned on a source domain $\mathcal{D}_S$ for its source task $\mathcal{T}_S$ and can be transferred to a target domain $\mathcal{D}_T$. The matching rules are learned via the Waikato Environment for Knowledge Analysis (WEKA) library, which provides a toolbox of learning algorithms [22].

4.2 Indexing

For the experiments conducted in this thesis no indexing is applied. All candidate record-pairs are defined in the silver standard.

4.3 Silver Standard Creation

In this section the motivation for creating silver standards for all matching tasks is explained. Based on this motivation, the silver standard creation process is illustrated and defined.

4.3.1 Motivation

The main motivation to generate silver standards instead of using gold standards is the lack of suitable benchmark data sets [4, 36, 53]. For the purposes of this thesis, different topics with several data sets and matching tasks are necessary. As the topics shall be semantically related, it is not possible to construct the topics from the available benchmark data sets [74].

Labelling data for all topics is tedious. Therefore, the idea is to exploit existing matches between KBs and different data sets. Bilenko et al. [4] combine legacy matches with non-matches, which are selected via an unsupervised approach, to create training sets for a matching rule. A minor drop in the performance of the generated rules is observed, which is explained by noise in the generated training set. Despite this noise it is shown that in cases where human labelling is too expensive, an unsupervised approach to acquired labelled non-matches is a viable

[2]https://github.com/olehmberg/winter

alternative.

As a sufficient amount of data is necessary and human labelling is too expensive, complete silver standards are generated. Using such an unsupervised approach enables control about the difficulty of the matching task. The difficulty of a matching task can be measured via the generated baselines.

As a side effect, the influence of different training data on the transferability of matching rules can be studied. Köpcke et al. [34] show that data selection via a similarity threshold has a positive impact on the performance of the learned matching rules. This is especially true for difficult matching tasks. Hence, the training data selection has a high influence on the learned matching rules. In the context of this thesis, the influence of the training data on the transferability of matching rule is interesting. Having control over the silver standard provides the opportunity to analyse this influence.

4.3.2 Creation Process

In this subsection the process to generate the silver standards is explained. Figure 4.1 shows the whole creation process. Based on the input an indexing is applied using the inverted index technique is done. Afterwards, the existing positive examples are used for an importance sampling over the blocked pairs. The outcome of the importance sampling is a silver standard. In the end this generated silver standard is split into a training and a test set. Along this process, the following sections explain the steps in detail.

Requirements

For each matching task a dedicated silver standard is generated. The inputs for the silver standard generation are the KB table, the corresponding data set table and a table of positive examples. The positive examples are all known matches for the given matching task between the KB table and the matching data set table.

Indexing

A first step to create the silver standard is indexing based on one chosen attribute of a topic. Please refer to Chapter 5 for details on which attribute is used for indexing in a topic.

An inverted index is used for indexing as explained in section 4.2. For the block-

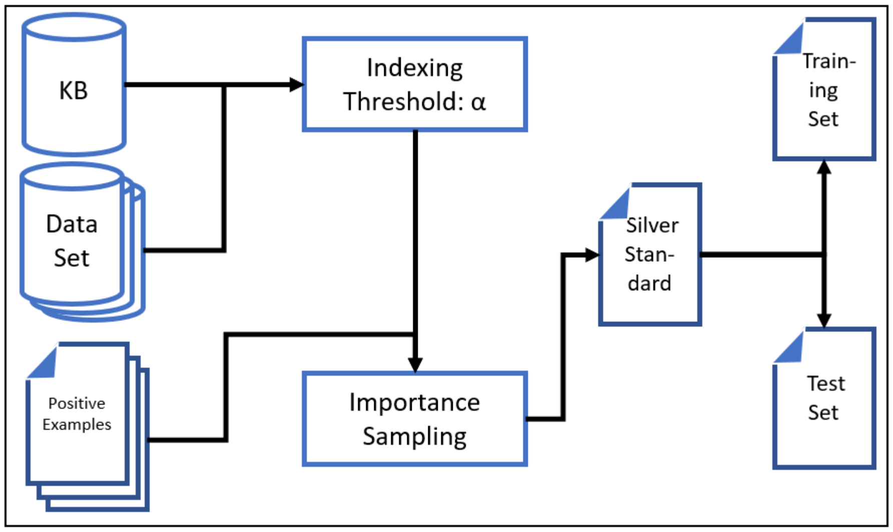


Figure 4.1: Silver Standard Creation Process

ing key value, substrings in brackets are removed from the chosen attribute's instance value. Then, the pre-processed blocking key value is tokenized using non-alphanumeric characters and the tokens are put into an inverted index. Using this inverted index, the overlap similarity of two blocking key values is calculated as shown in 4.1 with c being the count [12].

$$sim_{overlap}(s_1, s_2) = \frac{c_{common\_tokens}}{min(c_{tokens\_s_1}, c_{tokens\_s_2})} \tag{4.1}$$

A threshold α for the calculated overlap similarity score is set to reduce the number of blocked pairs. Only candidate record-pairs that have blocking key values, which have an overlap similarity score above α, are considered for further processing. Hence, α is a parameter, which has a high influence on the silver standard heterogeneity and consequently its difficulty. In the context of this thesis, α is used to change the difficulty of the silver standard. This threshold follows Köpcke et al.'s [34] approach to automatically select training data via a threshold. The remaining blocked pairs are handed over to the importance sampling.

Importance Sampling

An importance sampling is applied to the remaining blocked pairs in order to deal with the common problem of extreme class imbalance when conducting Identity

Resolution [50]. In other words, the idea is to balance matches and non-matches, which tends to make the learned matching rule more robust [34].

The implemented algorithm for sampling blocked pairs uses a comparable approach to Srinivas et al. [69], who use a nearest neighbours algorithm to find surface forms of strings that can potentially be joined together. Analogous to these string surface forms, the blocking key values are used to cluster candidate record-pairs that potentially describe the same real world entity.

The starting point for each cluster are the matching record-pairs from the provided ground truth. As it is assumed that no duplicates are contained in the initially provided data sets, all clusters are disjoint. Each cluster contains at most one matching record-pair.

Then each cluster is enhanced by candidate record-pairs from the indexing step, which are non-matching record-pairs according to the provided ground truth. Therefore, a non-matching record-pair is assigned to a cluster if one record of the pair is already assigned to this cluster. If two clusters are suitable, the first found cluster is chosen to keep the clusters disjoint.

After assigning all indexed non-matching record-pairs to a cluster, clusters containing only the initial matching record-pair are neglected.
Only in the mixed clusters, a discrimination between matching and non-matching record-pairs is necessary and relevant for learning a matching rule [69].
Within each cluster, non-matching record-pairs which have an overlap similarity score above or equal to the similarity score of the corresponding matching record-pair in the same cluster, are regarded as hard non-matches. All other non-matching record-pairs are considered to be semi-hard non-matches. Hard non-matches are hard to distinguish from the matching record-pair and are considered to be important for training a matching rule [69, 34, 1]. Figure 4.2 shows the distinction between hard non-matches and semi-hard non-matches according to the overlap similarity score.

For the importance sampling, record-pairs from the mixed clusters are added to the silver standard. The matching record-pair of a cluster is always added to the silver standard. To balance the silver standard, a maximum of three non-matches is added to the silver standard. As hard non-matches are considered to be more important than semi-hard non-matches, one semi-hard non-match counts for two hard non-matches. Figure 4.2 shows an explanatory example. The mixed cluster contains a match, a hard non-match and two semi-hard non-matches. Only the match, the

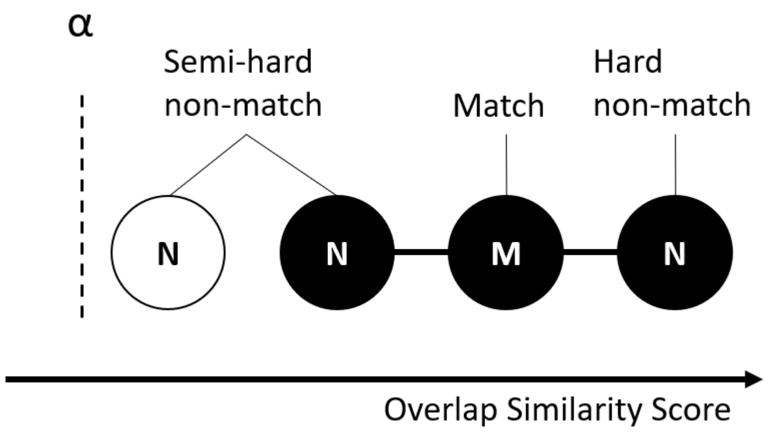


Figure 4.2: Determination of Record-Pairs for Silver Standard

hard non-match and the most similar semi-hard non-match are added to the silver standard. The second semi-hard non-match is neglected. The chosen record-pairs are filled with colour in figure 4.2. After the importance sampling, the final silver standard is generated.

Training and Test Sets

To enable matching rule learning, each silver standard is split into a training and a test set. 80 percent of the matching record-pairs of a silver standard are assigned to the training set and 20 percent of the non-matching record-pairs of a silver standard are assigned to the test set. Via the cluster of a matching record-pair, the non-matching record-pairs are assigned to the same set. This way the matching rule can learn how to identify the hard non-matches to discriminate between matching and non-matching record-pairs. Consequently, the final relative size of the silver standard varies slightly from the initial percentages of 80 percent for the training set and 20 percent for the test set.

4.4 Comparison Vector

As explained in Section 2.4, a matching rule can decide whether a candidate record-pair is classified as *match* or *non-match* based on a comparison vector [9]. To do so, the comparison vector has to be built for each blocked candidate record-pair. This section explains, how these comparison vectors are built.

This section is organised as follows. In Section 4.4.1 a general overview of the components of the used comparators is given. In the following sections the comparators for the data types *string*, *date*, *list*, *numeric* and *Record* are presented.

4.4.1 Comparator Architecture

Every comparator implements exactly one similarity measure. All similarity measures are normalized between 0.0 and 1.0. Different pre-processing steps and thresholds complement these similarity measures. Thresholds are for each comparator and can have values between 0.0 and 1.0. If a similarity score is below the defined threshold, the similarity score is set to 0.0. By setting the similarity score to 0.0, a minimum of similarity is required for this comparator [12].

To keep the comparison vectors similar across all topics, a list of comparators is assigned to each data type. Whenever attribute values are compared, the data type of the attribute is used to determine the specified list of comparators. These comparators calculate the similarity scores for the attribute values. At the end of the following sections, the list of comparators for the presented data type is illustrated. Additionally, the similarity measures as well as pre-/postprocessing steps defined for this data type are introduced in details.

4.4.2 General Measures

Similarity Measure - Exact

The *Exact* similarity measure checks whether the two presented values are equivalents. If both values are equivalent 1.0 is returned, otherwise the similarity score is 0.0 [12]. As this similarity measures return either 1.0 or 0.0, thresholds between 0.0 and 1.0 are not applicable.

Similarity Measure - Not Empty

If at least one value of the two presented values is null, the *Not Empty* similarity measure returns 0.0. Otherwise the *Not Empty* similarity measure returns 1.0. Checking for null values seems to be reasonable after the data profiling, since some domains contain sparsely filled attributes. Please refer to the data profiling sections of the topics in Chapter 5.

Comparators

The mentioned similarity measures in this section are used for all other data types. Hence, the comparators based on these similarity measures are mentioned at the comparator section of the following data types.

4.4.3 String

Pre-processing

Before calculating a similarity value for two *string* values, pre-processing steps can be applied. These pre-processing steps are lower casing (LC) and removing values in brackets (RB). To measure their influence, the pre-processing steps are either turned on or off.

Similarity Measure - Truncate Begin

The *Truncate Begin* similarity of a string evaluates whether the first character of two strings is equivalent. This similarity measure is a boolean measure, which returns either 0.0 or 1.0 [12]. Consequently, thresholds between 0.0 and 1.0 are not applicable.

Similarity Measure - Jaccard

Equation 5.1 shows the calculation of the token-based *Jaccard* similarity with c being the count [12]. To tokenizing the string values, two different approaches are applied. Strings are tokenized by non-alphanumeric characters and bi-gram character tokens are created. Resulting from these tokenizing approaches, two different similarity measures are derived.

$$sim_{jaccard}(s_1, s_2) = \frac{c_{common\_tokens}}{c_{tokens\_s_1} + c_{tokens\_s_2} - c_{common\_tokens}} \tag{4.2}$$

Similarity Measure - Levenshtein

Levenshtein is a basic edit distance measure $dist_{levenshtein}$, which defines the smallest number of single character insertions, deletions and substitutions necessary to transform one string into another string [43]. To convert the edit distance into a normalized distance between 0.0 and 1.0, equation 4.3 is employed [12].

$$sim_{levenshtein}(s_1, s_2) = 1.0 - \frac{dist_{levenshtein}(s_1, s_2)}{max(|s_1|, |s_2|)} \tag{4.3}$$

Similarity Measure - Jaro

Jaro is a variation of an edit distance measure, which considers a string's length. It is developed to account for the errors humans make in alphanumeric strings [80].

The *Jaro* similarity measure sets the number of common characters c_{common} into context with the string's character length. If $c_{common} = 0$, the *Jaro* similarity measure returns 0. Additionally, the number of transpositions $c_{transpositions}$ is considered. Transposition means that two characters are swapped in the two input strings, like 'ab' and 'ba'. Equation 4.4 shows the formula for the *Jaro* similarity measure [84]:

$$sim_{jaro}(s_1, s_2) = \frac{1}{3} - (\frac{c_{common}}{|s1|} + \frac{c_{common}}{|s2|} + \frac{c_{common} - c_{transpositions}}{c_{common}}) \quad (4.4)$$

Similarity Measure - JaroWinkler

JaroWinkler is an enhancement to the existing *Jaro* similarity measure. Empirical studies of misspellings reveal that less misspellings occur in the prefix than in the middle and the suffix of words [61]. So, for $sim_{winkler}$ the sim_{jaro} is increased if the beginnings of two strings have the same characters. The upper limit for the number of shared characters p is four [80]. Equation 4.5 shows the formula for the *Jaro* similarity measure [84]:

$$sim_{winkler}(s_1, s_2) = sim_{jaro}(s_1, s_2) + (1.0 - sim_{jaro}(s_1, s_2))\frac{p}{10} \quad (4.5)$$

Post-processing

After calculating a similarity score, it is possible to square the similarity score (SSC). By squaring the final similarity score, the importance of low values is further diminished. Squaring the similarity score is a post-processing step that can either be turned on or off.

Comparators

Each row in Table 4.1 represents a comparator that is applied to measure the similarity between string values. Every comparator implements one similarity measure. The pre- and post-processing steps can be independently turned on or off. Additionally, at most one threshold can be set. So, for one comparator multiple configurations are possible. In the context of this thesis, all possible combinations of pre- and postprocessing steps, as well as thresholds of a comparator are applied to calculate a similarity score for string values. The comparator implementing *Jaccard* with the underlined pre- and postprocessing steps and the underlined threshold is used to measure the similarity of string values in the linear combination baseline presented in 4.6.

Pre-/Post-Processing	Similarity Measure	Thresholds
	Exact	
Lower Case, No Brackets, Square	Jaccard	0.0, 0.2, 0.4, 0.6, 0.8
Lower Case, No Brackets, Square	Jaccard on Bigrams	0.0, 0.2, 0.4, 0.6, 0.8
Lower Case, No Brackets, Square	Jaro	0.0, 0.2, 0.4, 0.6, 0.8
Lower Case, No Brackets, Square	JaroWinkler	0.0, 0.2, 0.4, 0.6, 0.8
Lower Case, No Brackets, Square	Levenshtein	0.0, 0.2, 0.4, 0.6, 0.8
	Not Empty	
	Truncate Begin	

Table 4.1: String Comparators

4.4.4 Date

Pre-processing

For the data type *date* no configurable pre-processing is applied. The applied pre-processing step is a pattern matching for the data type *date* implemented in WInte.r[3]. Using the detected pattern, all *date* values are parsed into a unified format. Consequently, the pre-processed *date* values are all in the same format, when presented to the comparators.

Similarity Measure - ExactMonth

ExactMonth is a boolean similarity measure, which compares the year and the month of a date for equivalence. Respectively, the *ExactMonth* similarity measure returns 1.0 if year and month are equivalent or otherwise 0.0.

Similarity Measure - Levenshtein

Levenshtein is a basic edit distance measure $dist_{levenshtein}$, which defines the smallest number of single character insertions, deletions and substitutions necessary to transform one string into another string [43]. The *Levenshtein Date* similarity measure takes the pre-processed date value as input. This pre-processed data value is converted into a string. This way, all compared string values are in the same date format. To convert the edit distance into a normalized distance between 0.0 and 1.0, equation 4.6 is employed [12].

[3]https://github.com/olehmberg/winter/wiki/Data-Normalisation

$$sim_{levenshtein}(s_1, s_2) = 1.0 - \frac{dist_{levenshtein}(s_1, s_2)}{max(|s_1|, |s_2|)} \tag{4.6}$$

Similarity Measure - Absolute Difference

An absolute difference d_{max} of years between the two year values y_1 and y_2 of the date values is tolerated by the *AbsoluteYearDifference* similarity measure. The *AbsoluteYearDifference* similarity measure is used twice with d_{max} = 2 and d_{max} = 5. Based on equation 4.7, the *AbsoluteYearDifference* similarity score of two date values is calculated.

$$sim_{years\_abs}(y_1, y_2) = \begin{cases} 1.0 - \frac{|y_1 - y_2|}{d_{max}} & \text{if } |y_1 - y_2| < d_{max}, \\ 0.0 & \text{else.} \end{cases} \tag{4.7}$$

Comparators

Each row in Table 4.2 represents a comparator that is applied to measure the similarity between date values. Every comparator implements one similarity measure. For the comparator implementing the *Absolute Year Difference*, the absolute applied distance has to be defined. For the comparator implementing the *Levenshtein*, at most one threshold can be set. So, for one comparator multiple configurations are possible. In the context of this thesis, all possible combinations of absolute distances as well as thresholds of a comparator are applied to calculate a similarity score for date values. The underlined comparator implementing *Absolute Year Difference* with an absolute difference of 5 years is used in the linear combination baseline presented in 4.6.

Absolute Distance	**Similarity Measure**	**Thresholds**
2, <u>5</u>	<u>Absolute Year Difference</u>	
	Exact	
	Exact Month	
	Levenshtein	0.0, 0.2, 0.4, 0.6, 0.8
	Not Empty	

Table 4.2: Components of Date Comparators

4.4.5 List

Similarity Measure - Jaccard

The *Jaccard* similarity measure is *token-based*. The string values of the provided lists are tokenized by non-alphanumeric characters and put into a consolidated list of tokens. The *Jaccard* similarity measure is applied on this consolidated list of tokens. Equation 4.8 shows the calculation *Jaccard* similarity measure with c being the count [12].

$$sim_{jaccard}(s_1, s_2) = \frac{c_{common\_tokens}}{c_{tokens\_s_1} + c_{tokens\_s_2} - c_{common\_tokens}} \tag{4.8}$$

Similarity Measure - Overlap

Overlap similarity measure is *token-based*. The string values of the provided lists are tokenized by non-alphanumeric characters and put into a consolidated list of tokens. The *Overlap* similarity measure is applied on this consolidated list of tokens. Equation 4.9 shows the calculation *Overlap* similarity measure with c being the count [12].

$$sim_{overlap}(s_1, s_2) = \frac{c_{common\_tokens}}{min(c_{tokens\_s_1}, c_{tokens\_s_2})} \tag{4.9}$$

Comparators

Each row in Table 4.3 represents a comparator that is applied to measure the similarity between list values. Every comparator implements one similarity measure. The pre- and postprocessing steps can be independently turned on or off. Additionally, at most one threshold can be set. So, for one comparator multiple configurations are possible. In the context of this thesis all possible combinations of pre- and postprocessing steps as well as thresholds of a comparator are applied to calculate a similarity score for lists. The comparator implementing *Overlap* with the underlined pre- and postprocessing steps and the underlined threshold is used to measure the similarity of list values in the linear combination baseline presented in 4.6.

4.4.6 Numeric and Coordinate

Due to their similarity, numeric and coordinate values are compared using the same similarity measures [12].

Pre-/Post-processing	Similarity Measure	Thresholds
	Exact	
Lower Case, No Brackets, Square	Jaccard	0.0, 0.2, 0.4, 0.6, 0.8
Lower Case	Overlap	0.0, 0.2, 0.4, 0.6, 0.8
	Not Empty	

Table 4.3: Components of List Comparators

Similarity Measure - Absolute Difference

An absolute difference d_{max} between the two numeric values n_1 and n_2 is tolerated by the *Absolute Difference* similarity measure. d_{max} depends on the semantic context in which the *Absolute Difference* similarity measure is applied. For example, for latitude values d_{max} = 180 and for longitude values d_{max} = 360. Based on equation 4.10, the *Absolute Difference* similarity score of two numeric values is calculated.

$$sim_{num\_abs}(n_1, n_2) = \begin{cases} 1.0 - \frac{|n_1 - n_2|}{d_{max}} & \text{if } |n_1 - n_2| < d_{max}, \\ 0.0 & \text{else.} \end{cases} \tag{4.10}$$

Similarity Measure - Relative Difference

For the *Relative Difference* similarity measure no semantic knowledge is necessary. The relative distance of two numerical values is calculated as shown in equation 4.11. Then the distance measure is converted into a similarity measure.

$$sim_{num\_perc}(n_1, n_2) = 1.0 - \frac{|n_1 - n_2|}{max(|n_1|, |n_2|)} \tag{4.11}$$

Comparators

Each row in Table 4.4 represents a comparator that is applied to measure the similarity between numeric or coordinate values. Every comparator implements one similarity measure. For some comparators a threshold can be set. Consequently, for one comparator multiple configurations are possible. In the context of this thesis, all possible combinations of pre-processing steps as well as thresholds of a comparator are applied to calculate a similarity score for numeric and coordinate values.

Similarity Measure	Thresholds
Exact	
Absolute Difference	0.0, 0.2, 0.4, 0.6, 0.8
Relative Difference	0.0, 0.2, 0.4, 0.6, 0.8
Not Empty	

Table 4.4: Components of Numeric and Coordinate Comparators

4.4.7 Record

Similarity Measure - Overlap

The *Overlap* similarity measure is *token-based*. To generate the list of tokens, a bag of words based on all values of a record is created. The values in this bag of words are parsed to become string values. All resulting string values are tokenized by non-alphanumeric characters and put into a consolidated token list. Equation 4.12 shows the calculation *Overlap* similarity measure with c being the count [12].

$$sim_{overlap}(s_1, s_2) = \frac{c_{common\_tokens}}{min(c_{tokens\_s_1}, c_{tokens\_s_2})} \tag{4.12}$$

Comparators

Each row in Table 4.5 represents a comparator that is applied to measure the similarity between whole records. Every comparator implements one similarity measure. The pre-processing steps can be independently turned on or off. Additionally, at most one threshold can be set. So, for one comparator multiple configurations are possible. In the context of this thesis, all possible combinations of pre-processing steps as well as thresholds of a comparator are applied to calculate a similarity score for records.

Pre-processing	Similarity Measure	Thresholds
Lower Case	Overlap	0.0, 0.2, 0.4, 0.6, 0.8

Table 4.5: Components of Record Comparators

4.5 Learning Algorithms Used

In this section the learning algorithms for learning matching rules are introduced. The chosen algorithms for learning matching rules are an implementation for learning Decision Trees[4] and an implementation for learning a Logistic Regression[5]. These two models for matching rules are chosen, because the different knowledge components used by the matching rules can be easily stored and quickly interpreted by a human. Hence, it is possible to determine the importance of different matching rule parts.

This section is organised as follows. Section 4.5.1 introduces the algorithm to learn a Decision Tree and to measure feature importance in this decison tree. Section 4.5.1 introduces the algorithm to generate a Logistic Regression and to measure the influence of the different features on the Logistic Regression's result.

4.5.1 Decision Tree

Algorithm

A Decision Tree is a tree data structure that can be used to classify new instances. Each instance contains a set of features, which are within the Identity Resolution case similarity score of the comparison vector. At each node, a boolean test is done on a single feature to determine the path to follow. When an instance reaches a leaf node, the label i stored in the leaf node, is assigned to the instance. For all labels i it is given that $i \in I$, where I defines the label space [62]. In the context of Identity Resolution, the labels in the label space are *match* and *non-match* [12].

To learn a new Decision Tree, a training set with labelled instances is necessary. The C4.5 algorithm is a divide and conquer approach that tries to determine the best split at a each new node for the training set. Therefore, the C4.5 algorithm is configured to use the $GainRatio$ measure [62]. The ingredients for the $GainRatio$ measure are $Entropy$, $Gain_{split}$ and $SplitInfo$ [72]. Section 4.5.1 gives an overview of these measures.

Using a divide and conquer approach, the C4.5 algorithm grows a Decision Tree, which has only pure leaf nodes. Pure means that all training instances at a leaf node belong to the same class. Now, C4.5 applies pruning by estimating the error rate at each subtree and corresponding leaf node. If the leaf node's error rate is

[4]http://weka.sourceforge.net/doc.stable/weka/classifiers/trees/J48.html
[5]http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SimpleLogistic.html

lower than the subtree's error rate, the subtree is replaced by the leaf node [62]. The resulting Decision Tree can be used to classify unknown instances.

The technical implementation to learn a Decision Tree as described above is called J48[6]. J48 is implemented in the WEKA library [22].

Measures

The used measures from Information Theory are introduced in the following. When assuming that $p(i|t)$ denotes the fraction of instances belonging to class i at node t, the $Entropy$ is calculated as follows [72]:

$$Entropy(t) = -\sum_{j} p(i|t) * log_2(p(i|t)) \tag{4.13}$$

Using the $Entropy$, the formula for the Information $Gain_{split}$ is defined as:

$$Gain_{split} = Entropy(p) - \left(\sum_{j=1}^{k} \frac{N(v_j)}{N} Entropy(v_j) \right), \tag{4.14}$$

where N is the total number of instances at the parent node p, k is the number of partitions at p and $N(v_j)$ is the number of instances associated with the child node v_j [72].

The $SplitInfo$ is specified as follows:

$$SplitInfo = -\sum_{j=1}^{k} \frac{N(v_j)}{N} * log_2(\frac{N(v_j)}{N}), \tag{4.15}$$

where N is the total number of instances at the parent node p and $N(v_j)$ is the number of instances associated with the child node v_j [72].

Combining $Gain_{split}$ and $SplitInfo$, the $GainRatio$ is calculated as follows:

$$GainRatio = \frac{Gain_{split}}{SplitInfo} \tag{4.16}$$

[6]http://weka.sourceforge.net/doc.stable/weka/classifiers/trees/J48.html

Feature Importance

In the context of this thesis, the importance of different parts of the matching rule have to be determined. As described above, each split and consequently the underlying feature x_m have an impact on the subsequently chosen label. To measure this impact, the $GainRatioImportance$ of a feature x_m is introduced. The $GainRatioImportance$ is calculated by adding up the weighted $GainRatio$ for all nodes t, where x_m is used [7, 47]:

$$GainRatioImportance = \sum_{v(s_t)=x_m} p(t) GainRatio \tag{4.17}$$

and where $p(t)$ is the proportion $\frac{N_t}{N}$ of instances reaching t and $v(s_t)$ is the feature used in split s_t [47]. The value range for the $GainRatioImportance$ is between 0 and 1. Instead of aggregating the splits by feature, it is possible to use other parts of the matching rule for the aggregation. Depending on the asked question the $GainRatioImportance$ is adjustable. Please refer to Section 4.7 for further details.

4.5.2 Logistic Regression

Algorithm

A Logistic Regression models the posterior class probabilities $P(G = i|X = x)$ for classes in I. Given the estimators for the class probabilities, unseen instances are classified as follows [37]:

$$i^* = \arg\max_i P(G = i|X = x) \tag{4.18}$$

The main ingredients of a Logistic Regression model are the odds [75]:

$$\frac{P(y_i = 1|x_i)}{1 - P(y_i = 1|x_i)} \tag{4.19}$$

A Logistic Regression model of a two class problem is specified by the log-odds that separate class 1 from the base class 0 [37]:

$$log \frac{P(y_i = 1|x_i)}{1 - P(y_i = 1|x_i)} = \beta_i^T x \tag{4.20}$$

To fit a Logistic Regression model, the parameters vector β_i has to be estimated. In the *Simple Logistic Regression* implementations at hand, the LogitBoost algorithm for fitting additive Logistic Regression models by maximum likelihood is applied [20]. The general form of this Logistic Regression models is:

$$P(G = i|X = x) = \frac{e^{F_i(x)}}{\sum_{k=1}^{I} e^{F_k(x)}}, \quad \sum_{k=1}^{I} F_k(x) = 0, \tag{4.21}$$

where $F_i(x) = \sum_{m=1}^{M} f_{mi}(x)$. Each linear function $f_{mi}(x)$ predicts the label based on one input feature [37].

LogitBoost proposed by Friedman et al. [20] performs a forward stage-wise fitting. In each iteration of LogitBoost, the response variable that encodes the error of the current model on the training instances is computed. Adding a function $f_{mi}(x)$ to the committee F_i aims at improving the algorithm. A cross-validation determines the optimal number of LogitBoost iterations [37].

The implementation to generate a Logistic Regression as described above is called Simple Logistic Regression[7]. The WEKA library [22] implements the Simple Logistic Regression[8] used in this thesis.

Feature Importance

To interpret the resulting matching rule, the influence of the parts of the matching rule has to be measured. For this measurement, the odd of the all zero feature vector $Odd(x_1 = 0)$ is calculated first. Additionally, the $Odd(x_1 = 1)$ is determined. x_1 represents the part of the matching rule for which the importance is measured. The two odds are set into context via the $logOddRatio$ [75]:

$$log\left(\frac{Odd(x_1 = 1)}{Odd(x_1 = 0)}\right) = importance((x_1 = 1) - (x_1 = 0)) \tag{4.22}$$

From the variable $importance$, the influence of the part of the matching rule that represents x_1 can be derived [75]. The value range for the $importance$ is from $-\infty$ to ∞.

[7]http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SimpleLogistic.html
[8]http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SimpleLogistic.html

4.6 Experiments

This section introduces the framework for the conducted experiments.

The section is organised as follows. Section 4.6.1 sets Identity Resolution and Transfer Learning into context. Based on these definitions, the matching tasks in the different topics are identified in Section 4.6.2. Section 4.6.3 introduces the configuration for generating the corresponding silver standards of the matching tasks. Section 4.6.4 explains which experiments are conducted on the different matching tasks.

4.6.1 Identity Resolution using Transfer Learning

Before using Transfer Learning in the context of Identity Resolution, the two approaches have to be formally linked. Domain D and Task T are the starting point for the transfer learning definition. A domain is defined as $\mathcal{D} = \{\mathcal{X}, P(X)\}$. In the context of Identity Resolution $\mathcal{X}$ represents the comparison vectors generated as explained in Section 4.4. As the comparison vectors within a topic are generated the same way, $\mathcal{X}$ is identical for all domains in a topic. The marginal probabilities $P(X)$ are derived respectively for each domain D. It is assumed that $P(X)$ varies across domains. This variation of $P(X)$ is assessed in the defined RQs.

A task is defined as $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$. In the context of Identity Resolution, $\mathcal{Y}$ contains the two labels for *match* and *non-match*. As all defined tasks are Identity Resolution tasks, $\mathcal{Y}$ is identical for all matching tasks defined in this thesis. $f(\cdot)$ describes the respective matching rule, which aims to find matches in the list of candidate record-pairs.

4.6.2 Matching Tasks

Figure 4.3 shows the relations of a central KB table and the data set tables within a topic. The chosen example topic in the figure is the topic *Author*. All following explanations can be generalised for all other topics as well.

A matching task is always defined as matching instances from the KB table and either of the data set tables. Apart from the data set tables defined in Chapter 5, the additional data set table ’All’ is introduced. The ’All’ data set table contains all instances from the other data set tables. The idea behind the ’All’ data set table is to learn a general matching rule for matching instances from either of the data set tables and the KB table.

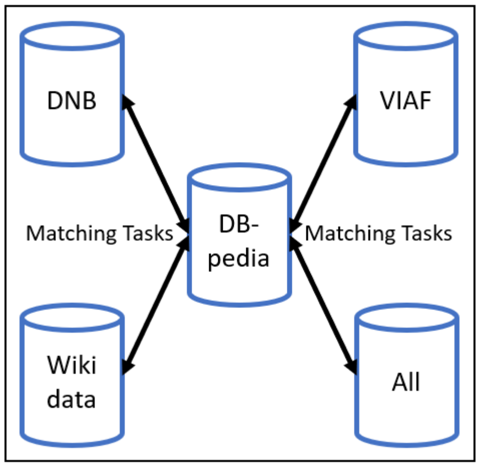


Figure 4.3: Matching Tasks - Topic *Author*

For every matching task shown in Figure 4.3 a silver standard is necessary to learn and evaluate matching rules. Section 4.6.3 explains the configuration for the generation of the silver standards.

4.6.3 Generating Silver Standards

For each matching task shown in Figure 4.3, two silver standards are generated:

- An 'easy' silver standard with an indexing threshold $\alpha = 0.45$
- A 'difficult' silver standard with an indexing threshold $\alpha = 0.7$

The two 'All' silver standards are constructed from all other silver standards with the same indexing threshold α. Each data set has to be equally represented in the training and test data set of the 'All' silver standard. Therefore, the smallest training set in terms of the number of matches and non-matches is completely transferred to the 'All' training data set. From the remaining training data sets, the same number of matches and non-matches is added to the 'All' training data set. Thereby, the mixed clusters are kept as explained in section 4.6.3. The same approach is applied for the test data set. Both of the 'All' silver standards are treated like the other silver standards in the topic.

Based on the findings of Köpcke et al. [34] it is assumed that a stricter threshold generates a more difficult corpus with more corner cases. Hence, all generated matching rules are tested using the test data from the silver standard with an indexing threshold $\alpha = 0.7$. This way the effect of different training data can be measured.

4.6.4 Experiments

Based on the matching tasks, three categories of experiments are conducted. First, the baselines are calculated. Second, the matching rules are learned and evaluated. Third, the learned matching rules are transferred and evaluated.

Baselines

To examine the quality of the learned matching rules, three baselines are generated:

- **Label Baseline**: The Jaccard similarity score on bi-grams of two labels is calculated.
- **Bag of Words Baseline**: The overlap similarity score between all tokens from all fields is calculated.
- **Linear Combination Baseline**: A linear combination is created, which values the Jaccard similarity on bi-grams of the two labels with 50 percent. The other 50 percent are equally shared amongst the remaining attributes. For each attribute, the calculated similarity score of one chosen comparator is added with the attributes' share to the linear combination. Please check Section 4.4 for details of the chosen comparator per data type.

After several iterations of manual assessment on the training sets, a threshold of 60 percent is set for all baselines. If the calculated similarity score of a baseline is above a value of 60 percent, the corresponding candidate record-pair is considered to be a match. All baselines are applied to the test data sets of the matching task to serve a reference point for the matching task's difficulty. Table 4.6 provides an overview of all baseline experiments applied to the topic *Author*.

Learning Matching Rules

For each matching task and silver standard, two matching rules are learned. The algorithms to learn matching rules are a Decision Tree (DT) algorithm[9] and a Lo-

[9] http://weka.sourceforge.net/doc.stable/weka/classifiers/trees/J48.html

Experiment	**Task**
Bag of Words Baseline	DBpedia-DNB
Bag of Words Baseline	DBpedia-VIAF
Bag of Words Baseline	DBpedia-Wikidata
Bag of Words Baseline	DBpedia-All
Label Baseline	DBpedia-DNB
Label Baseline	DBpedia-VIAF
Label Baseline	DBpedia-Wikidata
Label Baseline	DBpedia-All
Linear Combination Baseline	DBpedia-DNB
Linear Combination Baseline	DBpedia-VIAF
Linear Combination Baseline	DBpedia-Wikidata
Linear Combination Baseline	DBpedia-All

Table 4.6: Baseline Experiments for Matching Tasks - Topic *Author*

gistic Regression (LR) algorithm [10]. Both algorithms are introduced in Section 4.5. In total, four matching rules are learned for each matching task. All four learned matching rules are evaluated using the test data of the 'difficult' silver standard with an indexing threshold $\alpha = 0.7$.

Table 4.7 shows the experiments applied based on one silver standard of the topic *Author*. Overall, the number of experiments applied for learning matching rules is twice the number of experiments shown in Table 4.7.

Transfer Matching Rules

The matching task T, for which the previously mentioned matching rules are learned, is referred to as source task T_S. All other matching tasks in the same topic are defined as target matching task T_T. Each matching rule learned for T_S is transferred to all T_Ts. For this transfer, the learned matching rules are evaluated on the matching task's test data from the 'difficult' silver standard with an indexing threshold $\alpha = 0.7$.

Figure 4.4 shows how the matching rules are learned for the source matching task T_S and transferred to the target matching task T_T. In Figure 4.4, the red arrow between DBpedia and DNB symbolises the source matching task T_S for which the

[10]http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SimpleLogistic.html

Experiment	**Task**	**Algorithm**
Learn Rule	DBpedia-DNB	DT
Learn Rule	DBpedia-VIAF	DT
Learn Rule	DBpedia-Wikidata	DT
Learn Rule	DBpedia-All	DT
Learn Rule	DBpedia-DNB	LR
Learn Rule	DBpedia-VIAF	LR
Learn Rule	DBpedia-Wikidata	LR
Learn Rule	DBpedia-All	LR

Table 4.7: Learned Matching Rule Experiments for one Silver Standard - Topic *Author*

matching rule is learned. The black arrows between DBpedia and the other data sets symbolise the target matching tasks T_T, to which the matching rule learn for T_S is transferred.

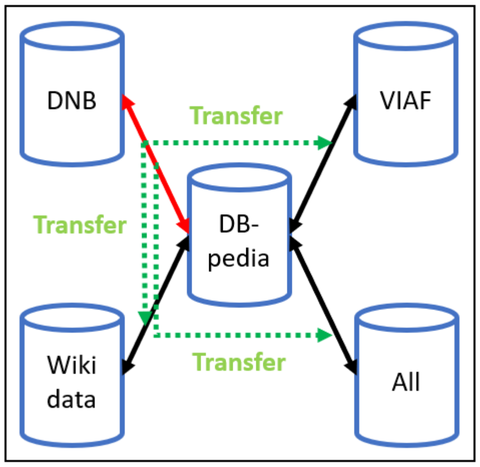


Figure 4.4: Architecture of matching rule transfer

Table 4.8 shows all experiments for the transfer of matching rules based on one silver standard of the topic *Author*.

Experiment	**Target Task**	**Source Task**	**Algorithm**
Transfer Rule	DBpedia-VIAF	DBpedia-DNB	DT
Transfer Rule	DBpedia-Wikidata	DBpedia-DNB	DT
Transfer Rule	DBpedia-All	DBpedia-DNB	DT
Transfer Rule	DBpedia-DNB	DBpedia-VIAF	DT
Transfer Rule	DBpedia-Wikidata	DBpedia-VIAF	DT
Transfer Rule	DBpedia-All	DBpedia-VIAF	DT
Transfer Rule	DBpedia-DNB	DBpedia-Wikidata	DT
Transfer Rule	DBpedia-VIAF	DBpedia-Wikidata	DT
Transfer Rule	DBpedia-All	DBpedia-Wikidata	DT
Transfer Rule	DBpedia-DNB	DBpedia-All	DT
Transfer Rule	DBpedia-VIAF	DBpedia-All	DT
Transfer Rule	DBpedia-Wikidata	DBpedia-All	DT
Transfer Rule	DBpedia-VIAF	DBpedia-DNB	LR
Transfer Rule	DBpedia-Wikidata	DBpedia-DNB	LR
Transfer Rule	DBpedia-All	DBpedia-DNB	LR
Transfer Rule	DBpedia-DNB	DBpedia-VIAF	LR
Transfer Rule	DBpedia-Wikidata	DBpedia-VIAF	LR
Transfer Rule	DBpedia-All	DBpedia-VIAF	LR
Transfer Rule	DBpedia-DNB	DBpedia-Wikidata	LR
Transfer Rule	DBpedia-VIAF	DBpedia-Wikidata	LR
Transfer Rule	DBpedia-All	DBpedia-Wikidata	LR
Transfer Rule	DBpedia-DNB	DBpedia-All	LR
Transfer Rule	DBpedia-VIAF	DBpedia-All	LR
Transfer Rule	DBpedia-Wikidata	DBpedia-All	LR

Table 4.8: Transfer Matching Rule Experiments for one Silver Standard - Topic *Author*

4.7 Evaluation of Research Questions

In this section the results of the experiments are linked to the RQs. The used measures, illustrations for these measures, as well as their interpretation are introduced. In Chapter 6, these foundations are used to evaluate the RQs.

One research question is introduced per subsection.

4.7.1 Research Question 1

RQ1: What is the performance of the learned matching rules?

Measurement

Four matching rules are learned for each matching task because of the two learning algorithms (Decision Tree and Logistic Regression) and the two indexing thresholds $\alpha = 0.45$ and $\alpha = 0.7$ for the silver standard creation. Additionally, three baselines are defined for each matching task (Bag-of-Words, Label and Linear Combination). For every learned matching rule and each baseline precision, recall and F are calculated.

Via the highest F score, the best baseline is selected for each matching task. To determine the quality of a learned matching rule, the $\triangle$ of precision, recall and F of the matching rule and the best baseline is determined. Exemplary for all $\triangle$s, the Precision-$\triangle$ of a learned matching rule is calculated as follows:

$$\text{Precision-}\triangle = \text{Precision}_{\text{Learned Matching Rule}} - \text{Precision}_{\text{Best Baseline}} \tag{4.23}$$

Illustration

Table 4.10 shows an exemplary evaluation of the learned matching rules on the matching task DBpedia-DNB (DNB). To interpret these results, Table 4.9 provides an overview of the used acronyms.

Acronym	**Meaning**	**Acronym**	**Meaning**	**Acronym**	**Meaning**
α	Indexing Threshold Silver Standard	BBL	Best Baseline	BoW	Bag of Words
DT	Decision Tree	LC	Linear Combination	LR	Logistic Regression
MT	Matching Task	MR	Matching Rule	Pr	Precision
Rec	Recall	Subscript S	Source	Subscript T	Target

Table 4.9: Overview of Acronyms

MT	α	MR	Pr	Rec	F	BBL	Pr-△	Rec-△	F-△
DNB	0.45	DT	0.896	0.968	0.930	BoW	0.133	-0.016	0.071
DNB	0.45	LR	0.824	0.984	0.897	BoW	0.062	0.000	0.038
DNB	0.7	DT	0.967	0.935	0.951	BoW	0.204	-0.048	0.092
DNB	0.7	LR	0.924	0.984	0.953	BoW	0.162	0.000	0.094

Table 4.10: Comparison Learned Matching Rule & Best Baseline - Matching Task *DBpedia-DNB*

Interpretation

If all performance deltas are positive for a matching rule, the learned matching rule outperforms the baseline. A matching rule which outperforms a baseline, is good. These results are set into context of the data profiling of the given topic.

According to the F-△ in Table 4.10, all matching rules outperform the baseline. The negative recall scores of the Decision Tree are compensated by high precision scores.

Additionally, it is possible to determine the impact of the different silver standards on the overall matching rule's performance. Therefore, the performance of the matching rules learned on an 'easy' silver standard ($\alpha = 0.45$) is compared to the performance of the matching rules learned on a 'difficult' silver standard ($\alpha = 0.7$). If the matching rules learned on a 'difficult' silver standard perform better, a positive impact of the training data selection via the indexing threshold α is observed.

In Table 4.10 the F-△ for the 'difficult' silver standard is higher than the F-△ for the 'easy' silver standard. This can indicate a positive impact of the training data selection.

4.7.2 Research Question 2

RQ2: To which extent can entire matching rules be transferred?

Measurement

A focus is set on the matching rules learned on the 'difficult' silver standard, as it is assumed that this silver standard has a positive influence on the matching rule's performance [34]. Two matching rules are learned for each matching task because of the two learning algorithms (Decision Tree and Logistic Regression). Every learned matching rule is transferred to all other matching tasks in the topic. These matching tasks become the target matching tasks, as explained in Section 4.6.4. The performance of the transferred matching rules on these target matching tasks is of interest. The benchmarks for this performance are the three baselines (Bag-of-Words, Label and Linear Combination). So, for every transferred matching rule and each baseline precision, recall and F are calculated.

Via the highest F score, the best baseline is selected for each matching task. To determine the quality of a transferred matching rule, the $\triangle$ of precision, recall and F of the matching rule and the best baseline is determined. Exemplary for all $\triangle$s, the Precision-$\triangle$ of a transferred matching rule is calculated as follows:

$$\text{Precision-}\triangle = \text{Precision}_{\text{Transferred Matching Rule}} - \text{Precision}_{\text{Best Baseline}} \tag{4.24}$$

Illustration

Table 4.11 shows an exemplary evaluation of the transferred matching rules on the matching task DBpedia-DNB (DNB). To interpret these results, Table 4.9 provides an overview of the used acronyms.

MT_T	**MT_S**	**MR**	**Pr**	**Rec**	**F**	**BBL**	**Pr-$\triangle$**	**Rec-$\triangle$**	**F-$\triangle$**
DNB	VIAF	DT	0.881	0.952	0.915	BoW	0.118	-0.032	0.056
DNB	VIAF	LR	0.950	0.919	0.934	BoW	0.188	-0.065	0.075
DNB	WD	DT	0.962	0.806	0.877	BoW	0.199	-0.177	0.018
DNB	WD	LR	0.978	0.726	0.833	BoW	0.216	-0.258	-0.026
DNB	All	DT	0.967	0.952	0.959	BoW	0.205	-0.032	0.100
DNB	All	LR	0.937	0.952	0.944	BoW	0.174	-0.032	0.085

Table 4.11: Comparison Transferred Matching Rule & Best Baseline - Matching Task *DBpedia-DNB*

Interpretation

If all performance $\triangle$ are positive for a matching rule, the transferred matching rule outperforms the baseline. A positive delta indicates that a transfer of this matching rule is beneficial. These results are set into context of the data profiling of the given topic.

According to the Precision-△ in Table 4.11, all matching rules outperform the baseline. Additionally, only the logistic regression learned on the matching task DBpedia-Wikidata (WD) leads to a slightly negative F-△. In general, the transferred rules are beneficial for the matching task DNB.

Furthermore, signs for transfer boundaries can be observed in the performance score table. As shown in Table 4.11, the two learned matching rules on the matching task WD reveal differences of source and target matching task. The recall scores of the two transferred matching rules are up to 0.258 lower than the baseline. This may be a sign for a transfer boundary, which has to be examined in detail in RQ3.

Lastly, the performance of the matching rule learned from the matching task DBpedia-All (All) is analysed. A positive delta shows how beneficial a fraction of training data from the target matching task is for the performance of the matching rule on the target matching task.

4.7.3 Research Question 3

RQ3: Is it more beneficial to transfer only parts of the matching rule instead of the entire rule?

Measurement

To answer RQ3, it is important to recall that each feature of the comparison vector is derived from a comparator that has an attribute and a similarity measure. Some comparators have a threshold as well. Therefore, three different matching rule parts are defined: *[Attribute]*, *[Attribute, Similarity Measure]* and *[Attribute, Similarity Measure, Threshold]*. Depending on the matching rule part in question, the importance of this part is calculated. The importance of a feature is determined differently for Decision Trees and Logistic Regression. For the Decision Tree, the additional matching rule part *[Attribute, Similarity Measure, Threshold, Split]* is determined. Hence, the split values of a Decison Tree can be exploited. A detailed explanation about feature importance for the two models is given in 2.5.

For the importance comparison, all matching rules are grouped by model and topic. Then, the matching rule parts in each group are ranked by their importance score. For each group, mean importance and the standard deviation of the mean importance is calculated. Due to the different importance value ranges of the two ML models, it is not possible to directly compare the importance scores of the models. Thus, for each group the mean rank and the standard deviation of the mean rank is calculated as well. In these calculations, all matching rules are weighted equally, because it is assumed that this rule is the best rule that can be learned for the given matching task. The learned matching rule depends on the provided comparison feature space $\mathcal{X}$, which does not allow any weighting of matching rules upfront.

Illustration

Table 4.12 shows an exemplary evaluation of a Decision Tree model's feature importance. The features are aggregated on attribute level. For the rank (R) and importance(I), mean and standard deviation are calculated.

	$\mathbf{R}_{mean}$	$\mathbf{I}_{mean}$	$\mathbf{R}_{var}$	$\mathbf{I}_{var}$
Attribute				
all	2.00	0.423702	1.224745	0.288305
birthdate	1.75	0.407125	0.829156	0.277161
label	2.50	0.129567	0.500000	0.090408
deathdate	3.50	0.020837	0.500000	0.009416
gender	4.00	0.011023	0.000000	0.000000

Table 4.12: Example: Attribute Importance for Decision Trees - Topic *Author*

Interpretation

When analysing the importance of a matching rule part, it is important to keep in mind that a preliminary filtering is done during the indexing of the silver standard creation. All hints found in this analysis depend on the attribute chosen for indexing.

In general, a low standard deviation of an importance score hints at the transferability of the matching rule within the topic. The attribute work is not used by any Decision Tree, so the standard deviation is 0. Hence, it seems that the attribute work is not useful for the Decision Tree in the topic *Author*. This conjecture is supported by the overall low density and high heterogeneity scores found in the data profiling section.

Chapter 5

Data Profiling

The topics are needed to conduct the experiments as described in Chapter 4. The goal of the experiments is to generate various insights about transferring matching rules. Hence, the topics have to be as diverse as possible. To show this diversity, this chapter conducts a data profiling of the different topics.

Therefore, this chapter provides an overview of the topics. This includes the data acquisition process and a data profiling for each topic.

This chapter is organised as follows. Section 5.1 introduces the foundations for the topics. This includes the topics components and the different data profiling dimensions. Section 5.2 gives a general overview of all topics. The sections 5.3, 5.4, 5.5 and 5.6 presents the topics *Author*, *Citation*, *City* and *Hotel* in detail.

5.1 Foundations

5.1.1 Topic Components

Every topic consists of a central KB table, several data set tables, as well as a table with positive examples. These positive examples are matches between instances from the knowledge base table and the corresponding data set tables. All components of a topic are the input for the silver standard creation and the experiments as specified in sections 4.3 and 4.6. Figure 5.1 illustrates the components of a topic.

5.1.2 Data Profiling Dimensions

Each section includes a general overview of the topic's structure. Therefore, the number of instances and missing attributes per data set table are provided. For the

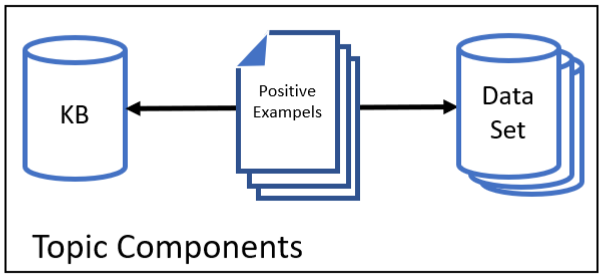


Figure 5.1: Components of a Topic

purposes of this thesis, all tables (KB table and data set tables) of a topic share the same schema. Technically, this enables the transfer of matching rules across matching tasks. If an attribute is not included in a table's schema after the data collection, the table is enhanced by an empty column representing the attribute.

Furthermore, the data profiling section covers characteristics on density and uniqueness per attribute in a topic's table. The density of an attribute is determined by the percentage of filled values in the corresponding column. Consequently, the uniqueness of an attribute describes the percentage of unique values in the respective column.

For the data set tables, the data profiling includes the average heterogeneity per attribute. To calculate the average heterogeneity, all values of a column from the data set table are matched to the values of the corresponding KB column via the positive examples of the matching task. Only matching values from the KB table and the data set table are compared. Compared values are tokenized as character bi-grams. On these tokens, the Jaccard distance is calculated as follows [12]:

$$dist_{jaccard}(s_1, s_2) = 1 - \frac{c_{common\_tokens}}{c_{tokens\_s_1} + c_{tokens\_s_2} - c_{common\_tokens}} \tag{5.1}$$

Based on the calculated distance score, the heterogeneity for an attribute A_i is calculated as follows:

$$\text{Heterogeneity}(A_i) = \frac{1}{|M|} \sum_{(r_1,r_2)\in M} dist_{jaccard}(r_1[A_i], r_2[A_i]), \tag{5.2}$$

with M representing all positive pairs (r_i, r_j). The heterogeneity characteristic is determined for all attributes even if the attribute's data type is not string. Those attribute values are parsed to a string value before the heterogeneity is calculated. Since the heterogeneity is an average across multiple Jaccard distances, the value range between 0.0 and 1.0 coincides with the range of the Jaccard distance. A value close to 0.0 indicates a low heterogeneity, whereas a value close to 1.0 is a sign for high heterogeneity.

5.2 Topics

Table 5.1 provides an overview of all used topics.

Topics	**Domains**	**Attributes (Data Type)**
Authors	DBpedia - *(KB)*	birthdate (date)
	DNB	deathdate (date)
	VIAF	gender (string)
	Wikidata	label (string) - *(Silver Standard)*
		work (list)
Citation	DBLP - *(KB)*	authors (list)
	ACM	freqTitle (string)
	Scholar	infreqTitle (string) - *(Silver Standard)*
		title (string)
		venue (string)
		year (date)
City	DBpedia - *(KB)*	country (string)
	GeoNames	label (string) - *(Silver Standard)*
	WebTable	latitude (coordinate)
	Wikidata	longitude (coordinate)
		population (numeric)
Hotel	revngo - *(KB)*	country (string)
	ihg	locality (string)
	nighttours	freqName (string)
	touristlink	infreqName (string) - *(Silver Standard)*
		postalcode (string)
		street (string)

Table 5.1: Overview Topics

The KB table, against which the matching takes place, is marked as *KB*. For each topic the schema is presented in form of the attributes and the attributes' data type.

The selection of the attributes aims for a variety with different characteristics that allows for various conclusions. Additionally, an attribute is chosen for the silver standard creation as explained in section 4.3. The chosen attribute is marked as the *Silver Standard* attribute.

5.3 Author

5.3.1 Data Sources

The four data sources for the topic *Author* are:

- **DBpedia** - KB: DBpedia facilitates data from Wikipedia to provide structured information[3].
- **DNB**: The catalogue of the Deutsche Nationalbibliothek (DNB) collects and provides access to any German literature in a structured manner[1].
- **VIAF**: Virtual International Authority File (VIAF) links authority files of different libraries like DNB and the Library of Congress. So, VIAF provides access to the records contained in the distinct libraries on a global scale [46].
- **Wikidata**: Wikidata is a collaborative KB that is used by Wikipedia and other sites to access its structured data [77].

5.3.2 Data Selection

All instances classified as *person*[2] and *Writer*[3] are acquired from DBpedia with all available attributes. The attribute *owl:sameAs* is exploited to identify instances of the same real world entity, which are contained in other KBs. A value of the attribute *owl:sameAs* contains a link to another record of another KB that describes the same real world instance [4]. Even though many quality issues with the *owl:sameAs* have been identified in the literature [59, 23], the *owl:sameAs* can be easily exploited for the purposes of this thesis to generate linked data sets.

From these *owl:sameAs links*, the three related KBs DNB, VIAF and Wikidata are identified as data set tables for the topic *Author*. These three KBs are chosen, because a sufficient amount of valid *owl:sameAs links* is available in the acquired DBpedia data.

[1]https://www.dnb.de/wir
[2]http://xmlns.com/foaf/spec/person
[3]http://DBpedia.org/ontology/Writer
[4]https://www.w3.org/TR/owl-ref/#sameAs-def

After acquiring all linked instances from the four KBs with all attributes, an attribute selection is done to unify the schema of the different tables. Based on the criteria density and variety of data types, the attributes *birthdate*, *deathdate*, *gender*, *label* and *work* are chosen. The attribute *label* is used to create the silver standards, because it is assumed that the name of an author is used to identify this author in a real-world scenario.

5.3.3 Data Collection

Knowledge Base Table DBpedia

With the decision for a set of topic attributes, the KB table *Author* is retrieved from the DBpedia Resource Description Framework (RDF) graph via SPARQL. In this query, only values with the language tag 'en' for English are considered in order to fetch only values containing characters from the Latin alphabet.

Data Set Tables

From the *owl:sameAs links* contained in DBpedia, the links to the corresponding instances in the KBs DNB, VIAF and Wikidata are collected. If a valid link is identified, the requested attribute values are extracted from the respective KB. Only the first valid link is taken, so that each DBpedia instance is linked to one DNB instance, one VIAF instance and one Wikidata instance at most. The identified instances are used to populate the respective data set table. The following data set specific processing is applied:

- Wikidata: Only values with the language tag 'en' are considered.
- VIAF: All values containing characters not included in the latin alphabet are excluded.
- DNB: The attribute *work* is not contained in the provided instances. To keep the schema consistent for the topic, the empty attribute *work* is added to all DNB instances.

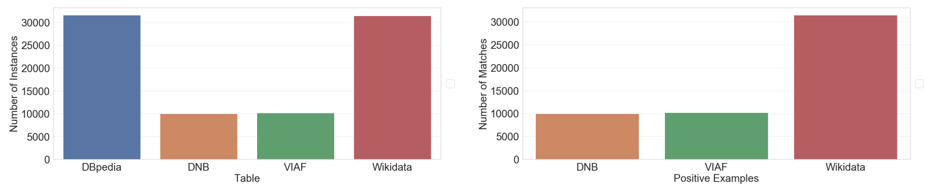
Positive Examples

From the valid *owl:sameAs links* found in the KB table, the positive examples for the topic *Author* are created. These positive links are not manually checked due to the enormous effort required to verify all pairs.

5.3.4 Data Profiling

Topic Overview

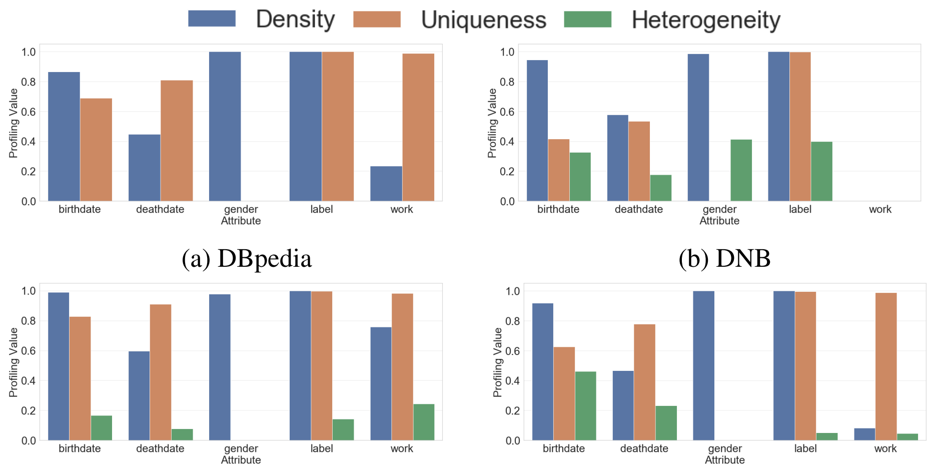
Figure 5.2 shows the sizes of the data set tables and the corresponding tables with positive examples. These numbers show how well DBpedia and Wikidata are interlinked. A corresponding instance for almost every author instance in DBpedia is found in Wikidata The numbers for DNB and VIAF are a lot lower, indicating that these KBs are not as good as Wikidata connected to DBpedia with regards to the selected instances.



(a) Table Sizes (b) Positive Examples

Figure 5.2: Topic Overview - *Author*

Figure 5.10 gives insights into the characteristics of the data sets in the topic *Author*. The following sections describe this profiling in detail.



(a) DBpedia (b) DNB

(c) VIAF (d) Wikidata

Figure 5.3: Data Profiling - *Author*

Knowledge Base Table DBpedia

The characteristics of DBpedia reveal that the label is maintained for all instances and has a very high uniqueness. So, the label is a good choice for creating the silver standard. The gender is mostly 'female' or 'male', which explains the low uniqueness score of the attribute gender. As every author has a unique list of works, the uniqueness of the attribute *work* is low. Simultaneously, a list of notable works is not maintained for every author, resulting in a low density score. As not all authors are dead yet, the density score of the attribute *deathdate* is naturally lower than the density score of the attribute *birthdate*. Consequently, the uniqueness score of the attribute *birthdate* is higher than the uniqueness score of the attribute *deathdate*.

Data Set Table DNB

When comparing the characteristics of the DNB data table with the other tables of the topic *Author*, the attributes seem to be comparable in terms of density and uniqueness. An indication that DNB and DBpedia have diverse characteristics is given by the comparably high heterogeneity value of the attribute label. Part of the explanation can be given by the different formats used to represent the label. In DBpedia, the label names are more accurate compared to DNB. For example the label 'Joan Francesc Mira i Caster' in DBpedia is maintained as 'Mira, Joan F.' in DNB. Additionally, the attribute *work* is missing und the *gender* values are maintained in German and not in English. Consequently, matching DBpedia and DNB is difficult.

Data Set Table VIAF

Overall, the density and uniqueness scores are comparable and the heterogeneity values are low. From the characteristic's point of view, DBpedia and VIAF seem to be similar, promising a comparably easy matching task.

Data Set Table Wikidata

Apart from the attribute *work*, all other attributes have high density values. The heterogeneity values for *label* and *gender* are extremely low indicating a high similarity. *Birthdate* and *deathdate* have higher heterogeneity values resulting from different format patterns. DBpedia provides date values in a 'YYYY-MM-DD' format. Whereas, Wikidata provides date values in a 'YYYY-MM-DDTHH:MM:SSZ'. Hence, pre-processing for values of data type date is necessary.

5.4 Citation

5.4.1 Data Sources

The four data sources for the topic *Citation* are:

- **DBLP** - KB: *Digital Bibliography & Library Project (DBLP)*[5] is a well-structured data source for publication entities, which is at least manually curated [36].
- **ACM**: *ACM digital library (ACM)*[6] is a well-structured data source for publication entities, which is at least manually curated [36].
- **Scholar**: *Google Scholar (Scholar)*[7] is a publication entity search engine, which automatically extracts its publication entities from web documents [36].

5.4.2 Data Selection

Köpcke et al. provide the two benchmark data sets[8] DBLP-ACM and DBLP-Scholar to evaluate real-world match approaches [36]. Several researchers use these benchmarks to evaluate their Identity Resolution approaches [36, 34, 39, 48, 53]. Both data sets contain structured data matching tasks based on publication instances. The instances contained in the first data set (DBLP-ACM) are retrieved from the two well-structured and partially manually curated bibliographic data sources DBLP and ACM. The second data set (DBLP-Scholar) contains publication instances from DBLP and publication instances from the entity search engine Scholar [36]. Both data sets contain a manually created gold standard, which is used to derive the positive examples for the topic *Citation*.

5.4.3 Data Pre-processing

All publication instances in the two data sets share the same schema by having the attributes *authors*, *title*, *venue* and *year*. From these attributes, the attributes *title* is initially chosen for the silver standard creation process. The assumption is made that a publication is first referred to using its title in a real world scenario

When indexing instances by *title*, as explained in section 4.3, the number of blocked

[5]http://dblp.org/

[6]https://dl.acm.org/

[7]https://scholar.google.de/

[8]https://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution

record-pairs is huge. An analysis of the attribute *title* reveals that many values contain stop words and frequently used terms such as 'data', 'database', or 'system' from the domain of computer science. These terms inflate the number of blocked pairs significantly, but do not help to distinguish publications. A solution to overcome this problem is a stop word removal before indexing the records [49]. Consequently, two new attributes *frequent part (of) title* and *infrequent part (of) title* are introduced.

For both attributes, all values from the attributes *title* across all data sets are tokenised using non-character values. All tokens, except for english stop words according to the Natural Language Toolkit (NLTK)[9] library, are ordered by their relative term frequency. A threshold is applied to split the tokens into two lists. Tokens above the threshold are assigned to the attribute *Frequent part title*. Tokens below the threshold are handed over to the attribute *Infrequent part title*. The attribute *Infrequent part title* is used for indexing and the silver standard creation. After several indexing iterations, the threshold to split the tokens is set to 0.4. In the end, all publication instances share the same schema by having the attributes *authors*,*frequent part title*, *infrequent part title title*, *venue* and *year*.

5.4.4 Data Collection

Benchmark DBLP-ACM

All instances from DBLP and ACM deal with the same set of computer science conferences and journals [36]. Furthermore, no duplicates are contained in the data according to the provided gold standard. So, the instances for the KB table DBLP and the data set table ACM are taken over from the (DBLP-ACM) data set.

Benchmark DBLP-Scholar

Before adding the DBLP instances to the corresponding KB table and the Scholar instances to the corresponding data set table, duplicates have to be identified and removed. This is due to the fact that the data from Scholar possesses some data quality issues, such as misspellings and duplicate publication instances [36]. Duplicate instances contained in the list of positive examples undermine the assumptions of the silver standard creation.

To identify duplicates in the data set (DBLP-Scholar), a connected graph is created. The publication instances serve as nodes in the graph and the positive links

[9]https://github.com/nltk/nltk

provided by the gold standard are the edges. An edge signals that these two nodes describe the same real-world entity. Consequently, multiple nodes connected to the same node are duplicates of each other. This identifies sets of duplicates in the DBLP-Scholar data set is identified. Only the first publication instances from each set is kept. All other instances are marked as duplicates and neglected for further processing. Figure 5.4 shows an extract of this graph. The kept nodes are filled with ink. All other nodes are neglected.

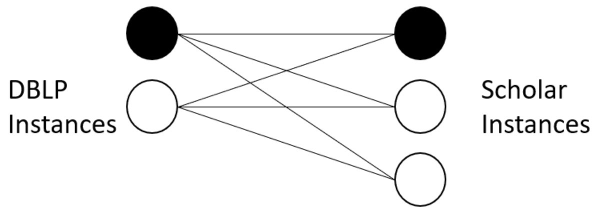


Figure 5.4: Elimination of Duplicates in the Data Set DBLP-Scholar

After removing these duplicates, the respective instances from DBLP are added to the KB table. From the remaining Scholar instances, the data set table Scholar is derived.

Positive Examples

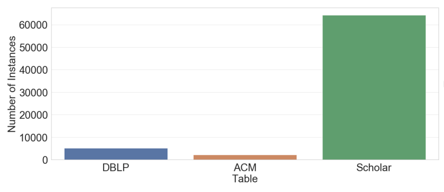
The positive examples of the topic contain matches from the gold standard provided by the data sets DBLP-ACM and DBLP-Scholar. Matching links from the data set DBLP-Scholar are removed, which refer to instances that are eliminated in an effort to remove duplicates from the data set. The remaining links are all used as positive examples for the topic *Citation*, which is used as input for the conducted experiments.

5.4.5 Data Profiling

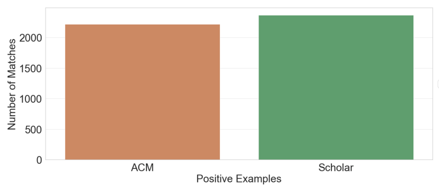
Topic Overview

Figure 5.5 reveals the imbalanced number of instances contained in the different tables. Table Scholar is roughly 20-times greater than the other tables. Even though the absolute number of instances in the Scholar table is a lot higher, the numbers of positive examples are comparable. A reason may be the automated data extraction leading to duplicated instances [36].

Figure 5.6 gives insights into the characteristics of the data sets in the topic *Citation*. The following sections describe this profiling in detail.

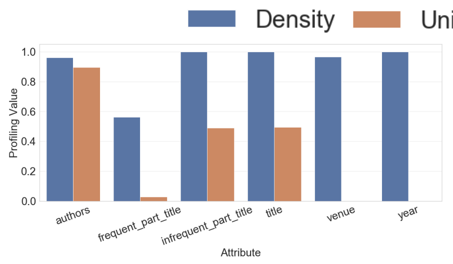


(a) Table Sizes

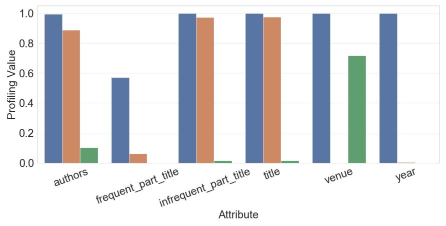


(b) Positive Examples

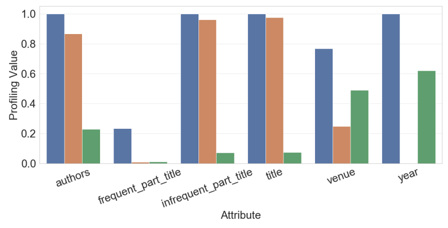
Figure 5.5: Topic Overview - *Citation*



(a) Data Profiling DBLP - *Citation*



(b) Data Profiling ACM - *Citation*



(c) Data Profiling Scholar - *Citation*

Figure 5.6: Data Profiling - *Citation*

Knowledge Base Table DBLP

All attributes apart from the newly generated attribute *frequent part title* are densely filled. The explanation of this lies in the observation that not all values of the attribute *title* contain frequent tokens. The uniqueness values for the attributes venue and year are very low, which is due to the low absolute number of different values. Only publications from five different venues within in nine years are covered. The uniqueness for the attribute *authors* is high. This may be helpful for the disambiguation of publication instances.

Data Set Table ACM

One can see that the density and uniqueness scores are high and that only the attribute *venue* has a high heterogeneity score. DBLP uses different names for the same venue, hence explaining the high heterogeneity score. The venue 'sigmod conference' in DBLP is covered as 'acm sigmod record' in ACM. All other attributes seem to be quite similar after the first data profiling.

Data Set Table Scholar

The characteristics of data set table Scholar are comparable to the characteristics of ACM. However, the attributes *venue* and *year* have high heterogeneity scores. This time the high heterogeneity can be explained by quality issues. Consequently, the attributes *venue* and *year* seem to be unsuitable for disambiguation purposes.

5.5 City

5.5.1 Data Sources

The four data sources for the topic *City* are:

- **DBpedia** - KB: DBpedia facilitates data from Wikipedia to provide structured information [3].
- **GeoNames**: The community driven database GeoNames[10] provides geographical data for countries and place names.
- **WebTable**: The web table *City* taken from the Web Data Commons table corpus [42]. The Web Data Commons table corpus is a large public corpus of Web tables, which contains over 233 million tables[11] and has been extracted from the July 2015 version of the CommonCrawl.
- **Wikidata**: Wikidata is a collaborative KB that is used by Wikipedia and other sites to access its structured data [77].

5.5.2 Data Selection

All instances classified as *city*[12] are acquired from DBpedia with all available attributes. The attribute *owl:sameAs* is exploited to identify instances of the same real world entity, which are contained in other KBs. A value of the attribute *owl:sameAs*

[10]http://www.geonames.org/
[11]http://webdatacommons.org/webtables/#results-2015
[12]http://DBpedia.org/ontology/City

contains a link to another record of another KB that describes the same real world instance [13]. Even though many quality issues with the *owl:sameAs* have been identified in the literature [59, 23], the *owl:sameAs* can be easily exploited for the purposes of this thesis to generate linked data sets.

Via the *owl:sameAs links* in the DBpedia KB table, the two related KBs GeoNames and WikiData are identified as data set tables for the topic *City*. These two KBs are chosen, because a sufficient number of *owl:sameAs links* is available, which points to existing instances in the target KBs.

The Web table City from the Web Data Commons table corpus is completely taken over, as data table *WebTable City* of the topic *City*. To link the instances from the Web Data Commons table corpus to the DBpedia KB table, the T2Dv2 Gold Standard for Matching Web Tables to DBpedia[14] is used.

After acquiring all linked instances from the three KBs with all attributes, an attributes selection is done to unify the schema of the different tables. Based on the criteria density and variety of data types, the attributes *country*, *label*, *latitude*, *longitude* and *population* are chosen. The attribute *label* is used to create the silver standards, because it is assumed that in a real-world scenario the name of city is used to identify a city in the first place.

5.5.3 Data Collection

Knowledge Base Table DBpedia

With the decision for a set of domain attributes, the KB table *City* is retrieved from DBpedia RDF graph via SPARQL. In this query only values with the language tag 'en' for English are considered to retrieve only characters from the Latin alphabet.

Data Set Table GeoNames and Wikidata

From the *owl:sameAs links* contained in DBpedia, the links to the corresponding instances in the KBs GeoNames and Wikidata are collected. If a valid link is identified, the requested attribute values are extracted from the respective KB. Only the first valid link is taken, so that each DBpedia instance is linked to one GeoNames instance and one Wikidata instance at most. The identified instances are used to

[13]https://www.w3.org/TR/owl-ref/#sameAs-def

[14]http://webdatacommons.org/webtables/goldstandardV2.html

populate the respective data set table. For Wikidata instances, only values with the language tag ’en’ are considered. Furthermore, as often multiple population values are provided for some Wikidata instances, the latest value in time is taken.

Data Set Table WebTable

The Web table City from the Web Data Commons table corpus is completely taken over as data table *WebTable City* of the topic *City*. To link the instances from the Web Data Commons table corpus to the DBpedia KB table, the T2Dv2 Gold Standard for Matching Web Tables to DBpedia[15] is used. These WebTable city instances cover all attributes except for the attributes *latitude* and *longitude*. To keep the schema consistent across all tables in the topic, the attributes *latitude* and *longitude* are added to the WebTable instances but left empty.

Positive Examples

Via the valid *owl:sameAs links* found in DBpedia, the positive examples for the matching tasks DBpedia-GeoNames and DBpedia-Wikidata are created. Due to the enormous effort, these positive links between instances are not manually checked.

To derive the positive examples for the matching task DBpedia-WebTable, the T2Dv2 Gold Standard for Matching Web Tables to DBpedia[16] is used. All matches in the T2Dv2 Gold Standard, which contain instances from the *WebTable City*, are moved to the table of positive examples.

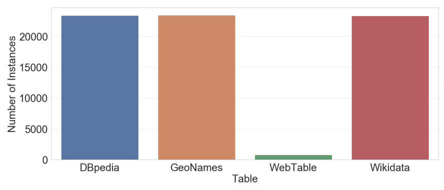
5.5.4 Data Profiling

Topic Overview

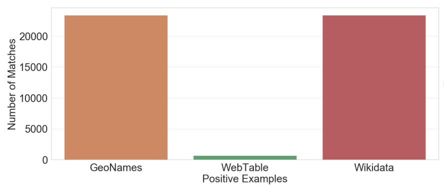
Figure 5.7 shows the table sizes and the numbers of positive examples. DBpedia, GeoNames and Wikidata are of a comparable size, showing how well the original data sets are linked. For almost every city instance in DBpedia a corresponding matching in GeoNames and WikiData is found. As the WebTable is a lot smaller than the other tables, the number of matching pairs is lower, as well. These different sizes may lead to different silver standard sizes, as well.

Figure 5.8 gives insights into the characteristics of the data sets in the topic *City*. The following sections describe this profiling in detail.

[15]http://webdatacommons.org/webtables/goldstandardV2.html
[16]http://webdatacommons.org/webtables/goldstandardV2.html

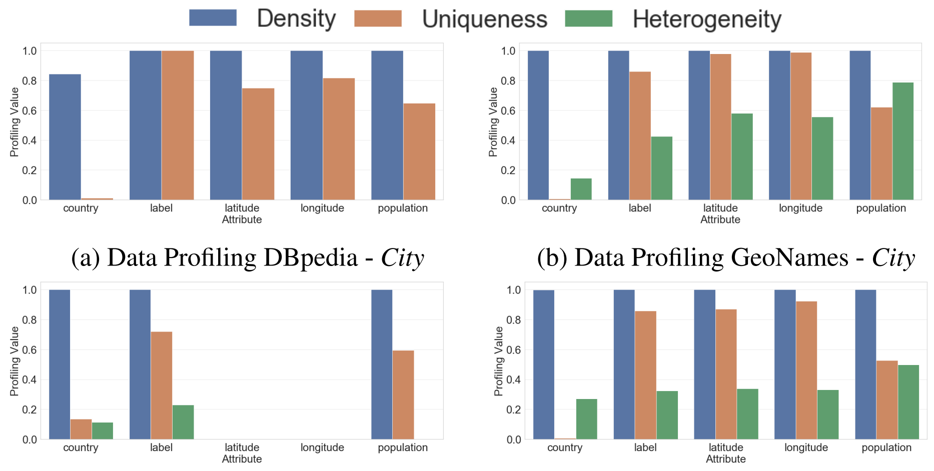


(a) Table Sizes



(b) Positive Examples

Figure 5.7: Topic Overview - *City*



(a) Data Profiling DBpedia - *City* (b) Data Profiling GeoNames - *City*

(c) Data Profiling WebTable - *City* (d) Data Profiling Wikidata - *City*

Figure 5.8: Data Profiling - *City*

Knowledge Base Table DBpedia

All attributes are densely filled. Apart from the attribute *country* all other attributes have comparably high uniqueness values explained by the low absolute number of mentioned countries. The high uniqueness value of the attribute *label* is interesting. Cities with the name 'Paris' or 'London' occur multiple times across the globe. DBpedia adds some detail like 'London, Arkansas' to make the values of the attribute *label* unique. Here the additional token 'Arkansas' may lead to decreased similarity values, which may distract learned rules. Fortunately, all matching rules have to deal with this issue, because DBpedia is the KB table. The lower uniqueness score of the attributes *latitude* compared to the attribute *longitude* can be explained by the bigger value range of latitude values.

Data Set Table GeoNames

In general, the density and uniqueness scores of the DBpedia instances and the GeoNames instances are comparable. All heterogeneity values are high except for the attribute *country*. Checking the values of the attribute *label* in detail reveals that a lot of values occur multiple times. Unlike in DBpedia, this *label* attribute provides only the city's name without an attempt at making this name unique. Hence, the comparably high heterogeneity value can be explained. The high heterogeneity of coordinate and numeric values may be a result of a different rounding.

Data Set Table WebTable

As the coordinate attributes are missing in the table WebTable, only the attributes *country*, *label* and *population* are useful. The low uniqueness score of the attributes *country* indicates that it is not useful for disambiguation purposes. Again, the attribute *label* is not unique, such as in DBpedia. If a disambiguation based on the attributes *label* and *country* is not possible, only the attribute *population* of a city instance can help. Matching DBpedia and WebTable is assumed to be the most difficult task in this topic.

Data Set Table Wikidata

Similar to the table GeoNames, the table WikiData the attributes *label* does not have unique values. WikiData provides the coordinate values in an extremely accurate way. This results in a high uniqueness score, which can help matching record-pairs. The accuracy of the coordinate values and the *population* values explains the high heterogeneity as well.

5.6 Hotel

5.6.1 Data Sources

The data source for the topic *Hotel* is the micro data corpus from the Web Data Commons Microdata corpus extract in November 2017[17] [51]. From the mirco data corpus, the following four hosts are chosen to provide the hotel instances for this topic:

[17]http://www.webdatacommons.org/structureddata/2017-12/stats/schema_org_subsets.html

- **revngo** - KB: revngo[18] is a web page to book hotels.
- **ihg**: ihg[19] is a web page to book hotels.
- **nighttours**: nighttours[20] is a web page to find hotels suitable for homosexuals.
- **touristlink**: touristlink[21] is a social network that enables an exchange of experiences among travelers and locals.

5.6.2 Data Selection

In the Web Data Commons Microdata corpus a subject of type hotel is identified via the rdf-type hotel from schema.org[22]. Via the hotel's n-quad subject, attributes like the hotel's name or address can be assigned to the identified hotel instance.

In a real-world scenario hotels are mostly identified via their name, so only hotel instances that have a name are extracted from the micro data corpus. All extracted hotels are grouped by host. If the hotel name is the only attribute mentioned in a group, the complete group is not further considered. Hence, the kept hotel instances have a variety of attributes.

From the list of remaining hotel instances, the most frequent hosts are determined. Additionally, the schema of the instances from the most frequent hosts is analysed. In the end, the instances from the host 'revngo' are assigned to KB table of the topic *Hotel*. The hosts 'ihg', 'nighttours' and 'touristlink' provide the instances of the three linked data set tables.

To unify the schema of the different tables, an attributes selection is done. Via the criteria density and variety of attribute types, the attributes *locality*, *name*, *postal_code* and *street* are chosen.

5.6.3 Data Pre-processing

The hotel's name is initially chosen for the silver standard creation process. When indexing instances by hotel name as explained in section 4.3, the amount of blocked pairs is huge. A reason is that many hotel names contain frequently used terms like 'pension', 'hotel' or 'comfort'. These terms inflate the number of blocked pairs

[18]https://en.revngo.com/
[19]https://www.ihg.com/
[20]https://www.nighttours.com/
[21]https://www.touristlink.com/
[22]https://schema.org/Hotel

significantly, but do not help to distinguish hotels. A solution to overcome this problem is a stop word removal during indexing [49]. Consequently, two new attributes regarding the frequent part of name (*frequent part name*) and infrequent part of name (*infrequent part name*) are introduced.

The procedure applied to fill these two attributes is the same as in the pre-processing section of the topic *Citation*. The attribute *Infrequent part title* is used for indexing and the silver standard creation. Additionally, tokens describing cities like 'London', 'Paris' or 'Beijing' are kept in the infrequent part to facilitate these tokens during indexing. After several indexing iterations, the threshold to split the tokens is set to 0.15. This pre-processing step finalises the topic's schema. The schema contains the attributes *frequent_part_name*, *infrequent_part_name locality*, *name*, *postal_code* and *street*.

5.6.4 Data Collection

Hotel Instances

All hotel instances assigned to the selected hosts 'revngo', 'ihg', 'nighttours' and 'touristlink' are extracted from the Web Data Commons Microdata corpus and assigned to a dedicated data set table.

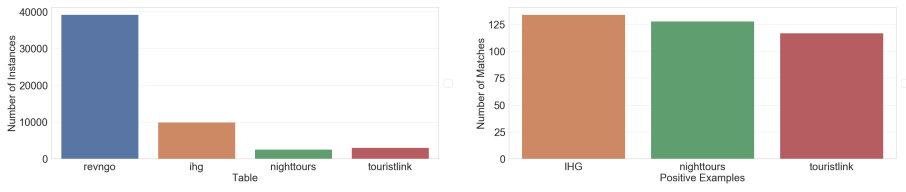
Positive Examples

The positive examples for the topic *Hotel* are annotated by a human annotator. Therefore, hotel instances from the KB table and the data set tables are indexed in accordance with the silver standard creation. The blocked pairs are manually annotated. Based on these annotations, three tables with positive examples are generated. Each table contains matches for instances from the KB table and one of the data set tables.

5.6.5 Data Profiling

Topic Overview

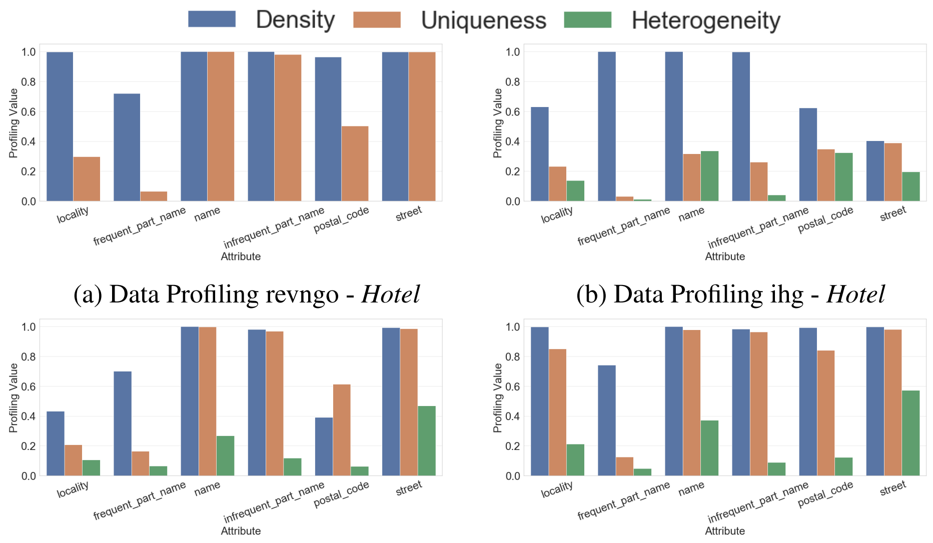
Figure 5.9 shows the table sizes and the numbers of positive examples. The figure reveals the difference in the number of hotel instances per table. In fact, nighttours and tourristlink contain only a fraction of the number of hotel instances contained in revngo and ihg. Nevertheless, the numbers of positive examples are equally low. Compared to all other topics, the numbers of positive examples are very low. This is due to the high manual annotating effort, as well as the high variety of hotel instances mentioned in the different tables. Hence, it is assumed that only a small intersection of hotel instances exists across the whole topic. Since no other linkage between the hotel instances is found, this is the only approach to retrieve positive examples for the topic hotel. An advantage of this approach is that the link quality is higher compared to the other topics, because these links are manually checked.



(a) Table Sizes (b) Positive Examples

Figure 5.9: Topic Overview - *Hotel*

Figure 5.8 gives insights into the characteristics of the data sets in the topic *City*. The following sections describe this profiling in detail.



(a) Data Profiling revngo - *Hotel* (b) Data Profiling ihg - *Hotel*

(c) Data Profiling nighttours - *Hotel* (d) Data Profiling touristlink - *Hotel*

Figure 5.10: Data Profiling - *Hotel*

Knowledge Base Table Revngo

All initially determined attributes have high density values in the table revngo, which is helpful for Identity Resolution. This is supported by high uniqueness scores of the attributes *name* and *street*.

Data Set Table ihg

Except for the attribute *name*, the attributes are only densely filled. Only the attribute *name* is filled for all instances, but the attribute has a comparably low uniqueness value. The reason for this low uniqueness is that many hotel instances belong to a hotel chain like 'Holiday Inn Express & Suites'. As the hotel chain's name is contained in the hotel name and the uniqueness is calculated based on tokens, the overall uniqueness of the hotel name decreases. Hence, it is assumed that the matching task revngo-ihg is the most difficult task in this topic.

Data Set Table nighttours

All attributes are densely filled apart from the attributes *postalcode* and *locality*. As the heterogeneity of the attributes *locality*, *name*, *freq part name* and *infreq part name* is low, these attributes seem to be helpful for Identity Resolution. This is underlined for the attributes *locality*, *name* and *infreq part name* by a low uniqueness value. The attribute *street* can be useful in combination with other attributes, because of its high uniqueness and heterogeneity scores. Street values are represented in different formats explaining the high heterogeneity. '33306 2648 N.E. 32nd St' in revngo is represented as '2648 NE 32 Street' in nighttours.

Data Set Table touristlink

The characteristics of the table nighttours are very similar to those of the table touristlink. A main difference is that the attributes *locality* and *postalcode* have higher density values, which can be beneficial for Identity Resolution. Another minor difference is that the heterogeneity scores for the attributes *locality*, *name*, *postalcode*2 and *stree* are slightly higher. This can be an indication that the matching task revngo-touristlink is more difficult than the matching task revngo-nighttours.

Chapter 6

Experiments

To answer the asked research questions, this chapter summarises the results of all conducted experiments for Identity Resolution using Transfer Learning.

The chapter is organised as follows. Each of the sections evaluates the experiments related to one research question. The foundations for this evaluation can be found in Section 4.7. At the end of each section, the key findings are summarised.

6.1 Research Question 1

RQ1: What is the performance of the learned matching rules?

To answer this question, each of the following sections analyses the learned matching rules of one topic.

6.1.1 Author

Table 6.1 presents the performance scores of the comparison between the learned matching rules and the best baseline for the respective matching task.

The majority of matching rules outperforms the best baseline in terms of precision, recall and F score. On the matching task DBpedia-DNB (DNB), only the Decision Tree fails to outperform the baseline in terms of the recall score. An explanation can be found in the data profiling of DNB showing the obstacles of this matching task.

The 'easy' silver standard leads to a precision score below the best baseline's pre-

MT	α	MR	Pr	Rec	F	BBL	Pr-△	Rec-△	F-△
DNB	0.45	DT	0.896	0.968	0.930	BoW	0.133	-0.016	0.071
DNB	0.45	LR	0.824	0.984	0.897	BoW	0.062	0.000	0.038
DNB	0.7	DT	0.967	0.935	0.951	BoW	0.204	-0.048	0.092
DNB	0.7	LR	0.924	0.984	0.953	BoW	0.162	0.000	0.094
VIAF	0.45	DT	0.676	0.990	0.803	BoW	-0.178	0.222	-0.005
VIAF	0.45	LR	0.742	0.990	0.848	BoW	-0.112	0.222	0.040
VIAF	0.7	DT	0.970	0.980	0.975	BoW	0.116	0.212	0.166
VIAF	0.7	LR	0.970	0.980	0.975	BoW	0.116	0.212	0.166
WD	0.45	DT	0.698	1.000	0.822	Label	0.028	0.020	0.026
WD	0.45	LR	0.710	1.000	0.830	Label	0.040	0.020	0.034
WD	0.7	DT	0.976	0.995	0.985	Label	0.306	0.015	0.189
WD	0.7	LR	0.935	0.985	0.959	Label	0.265	0.005	0.163
All	0.45	DT	0.753	0.984	0.853	BoW	0.077	0.065	0.074
All	0.45	LR	0.702	0.989	0.821	BoW	0.026	0.070	0.042
All	0.7	DT	0.963	0.973	0.968	BoW	0.287	0.054	0.189
All	0.7	LR	0.943	0.978	0.960	BoW	0.267	0.059	0.181

Table 6.1: Comparison Learned Matching Rule & Best Baseline - Topic *Author*

cision score for the both learned matching rules of the matching task DBpedia-VIAF. Increasing the difficulty of the silver standard improves the precision scores by more than 0.2. Such precision score improvements are also observed for the matching tasks DBpedia-WikiData (WD) and DBpedia-All (All), when increasing the difficulty of the silver standard. These results are indications of the positive influence of a selected training set with an increased difficulty.

6.1.2 Citation

Table 6.2 presents the results of the comparison between the learned matching rules and the best baseline for the respective matching task.

In the topic Citations, a clear difference between the matching task DBLP-ACM (ACM) and DBLP-Scholar (SL) is observed with regards to the performance scores. The performance scores of the learned matching are lower for SL than for ACM. An explanation may be the quality issues reported by the providers of the initial data set [35], as well the findings of the data profiling. However, in terms of F score, all matching rules outperform the baseline or have the comparable performance to the baseline.

MT	α	**MR**	**Pr**	**Rec**	**F**	**BBL**	**Pr-△**	**Rec-△**	**F-△**
ACM	0.45	DT	0.879	0.935	0.906	LC	0.204	0.000	0.122
ACM	0.45	LR	0.853	0.935	0.892	LC	0.179	0.000	0.109
ACM	0.7	DT	0.879	0.935	0.906	LC	0.204	0.000	0.122
ACM	0.7	LR	0.879	0.935	0.906	LC	0.204	0.000	0.122
SL	0.45	DT	0.469	0.778	0.585	Label	0.022	-0.071	0.000
SL	0.45	LR	0.459	0.921	0.613	Label	0.013	0.071	0.028
SL	0.7	DT	0.470	0.824	0.599	Label	0.024	-0.025	0.014
SL	0.7	LR	0.466	0.925	0.620	Label	0.020	0.075	0.035
All	0.45	DT	0.628	0.790	0.700	Label	0.118	-0.048	0.066
All	0.45	LR	0.545	0.871	0.671	Label	0.036	0.032	0.037
All	0.7	DT	0.685	0.806	0.741	Label	0.175	-0.032	0.107
All	0.7	LR	0.626	0.919	0.745	Label	0.117	0.081	0.111

Table 6.2: Comparison Learned Matching Rule & Best Baseline - Topic *Citation*

Increasing the 'difficulty' of the silver standard increases the performance scores of the learned matching rules compared to the matching rules learned on the 'easy' silver standard. Only the Decision Tree on the matching task ACM produces the same performance scores on both silver standards. A positive impact of the increased difficulty of the training set is observed.

6.1.3 City

Table 6.3 presents the results of the comparison between the learned matching rules and the best baseline for the respective matching task.

In terms of the F score, all learned matching rules outperform the best baseline. However, a clear difference between the matching tasks DBpedia-GeoNames (GN) and DBpedia-WikiData (WD), as well as DBpedia-WebTable (WT) is observed with regards to the performance scores. Furthermore, the precision score is comparably low for the matching task WT and contrary to all other matching tasks, the best baseline for WT is the label baseline. This can be interpreted as an indication of the importance of the two missing coordinate attributes, as well as for the disambiguation limitations of the other attributes as explained in the data profiling. Furthermore, the performance scores of the matching task WT are only slightly worse, when the matching rules are trained on the difficult silver standard. An additional explanation may be the comparably small number of positive examples

MT	α	MR	Pr	Rec	F	BBL	Pr-△	Rec-△	F-△
GN	0.45	DT	0.978	0.996	0.987	BoW	0.212	0.001	0.121
GN	0.45	LR	0.963	0.995	0.979	BoW	0.196	-0.001	0.112
GN	0.7	DT	0.995	0.991	0.993	BoW	0.228	-0.005	0.126
GN	0.7	LR	0.977	0.995	0.986	BoW	0.210	-0.001	0.119
WT	0.45	DT	0.584	0.945	0.722	Label	-0.002	0.145	0.045
WT	0.45	LR	0.546	0.964	0.697	Label	-0.040	0.164	0.020
WT	0.7	DT	0.571	0.945	0.712	Label	-0.015	0.145	0.035
WT	0.7	LR	0.536	0.945	0.684	Label	-0.051	0.145	0.007
WD	0.45	DT	0.992	0.996	0.994	BoW	0.268	-0.004	0.154
WD	0.45	LR	0.973	0.999	0.986	BoW	0.249	-0.001	0.146
WD	0.7	DT	0.996	0.996	0.996	BoW	0.271	-0.004	0.156
WD	0.7	LR	0.990	0.997	0.994	BoW	0.266	-0.003	0.153
All	0.45	DT	0.800	0.970	0.877	BoW	0.001	0.297	0.146
All	0.45	LR	0.767	0.976	0.859	BoW	-0.032	0.303	0.128
All	0.7	DT	0.818	0.952	0.880	BoW	0.019	0.279	0.149
All	0.7	LR	0.782	0.976	0.868	BoW	-0.017	0.303	0.138

Table 6.3: Comparison Learned Matching Rule & Best Baseline - Topic *City*

provided for the WebTable data set table resulting in a small training set.

All other matching rules benefit from the difficulty training data set. In the case of the matching tasks GN, WD and DBpedia-All (All), the difficulty training data helps to produce more precise matching rules. In general, a positive impact of the 'difficult' training set is observed. At the same time, the matching task WT seems to show the limitation of this approach.

6.1.4 Hotel

Table 6.4 presents the results of the comparison between the learned matching rules and the best baseline for the respective matching task.
In terms of recall and F score all learned matching rules outperform the best baseline. It is striking that the precision score of the baseline performs at least as well as the learned matching rules, even slightly outperforming them. Furthermore, the 'difficult' silver standard only barely improves the learned matching rule's performance on the matching tasks Revngo-Nighttours (NT) and Revngo-Touristlink (TL). No clear tendency is observable for the other two matching tasks Revngo-Ihg (IHG) and Revengo-All (All).

MT	α	MR	Pr	Rec	F	BBL	Pr-△	Rec-△	F-△
IHG	0.45	DT	0.676	0.926	0.781	Label	-0.021	0.074	0.015
IHG	0.45	LR	0.694	0.926	0.794	Label	-0.003	0.074	0.027
IHG	0.7	DT	0.643	1.000	0.783	Label	-0.054	0.148	0.016
IHG	0.7	LR	0.667	0.963	0.788	Label	-0.030	0.111	0.021
NT	0.45	DT	0.952	1.000	0.976	Label	-0.048	0.200	0.087
NT	0.45	LR	0.952	1.000	0.976	Label	-0.048	0.200	0.087
NT	0.7	DT	0.952	1.000	0.976	Label	-0.048	0.200	0.087
NT	0.7	LR	1.000	1.000	1.000	Label	0.000	0.200	0.111
TL	0.45	DT	0.810	1.000	0.895	BoW	-0.073	0.118	0.012
TL	0.45	LR	0.850	1.000	0.919	BoW	-0.032	0.118	0.037
TL	0.7	DT	0.850	1.000	0.919	BoW	-0.032	0.118	0.037
TL	0.7	LR	0.850	1.000	0.919	BoW	-0.032	0.118	0.037
All	0.45	DT	0.877	0.980	0.926	Label	-0.056	0.157	0.051
All	0.45	LR	0.860	0.961	0.907	Label	-0.074	0.137	0.032
All	0.7	DT	0.860	0.961	0.907	Label	-0.074	0.137	0.032
All	0.7	LR	0.877	0.980	0.926	Label	-0.056	0.157	0.051

Table 6.4: Comparison Learned Matching Rule & Best Baseline - Topic *Hotel*

An explanation for this behaviour may be the comparably small number of positive links initially labelled for this topic. Consequently, the diversity of the tasks may not be sufficiently covered in the training data sets. Furthermore, the data profiling of this topic already reveals difficulties. Especially, ihg has comparably high heterogeneity scores and at the same time comparably low density scores. Thus, from this topic's performance scores, no indication about the impact of the silver standard can be given.

6.1.5 Summary

To give a general answer about the performance of the learned rules, the mean F deltas of the matching rules are compared across the different topics. Therefore, the matching rules of a topic are grouped by indexing threshold of the silver standard to analyse the impact of the silver standard on the learned matching rule. Furthermore, all matching rules of a topic belonging to the matching task 'All' are put into the group 'All'. The other matching rules of a topic are put into the group 'Other'. Hence, it is possible to compare the performance of the matching rules learned on the 'All' matching tasks with the other matching rules. Consequently, Figure 6.1

illustrates four mean F scores representing the four generated groups.

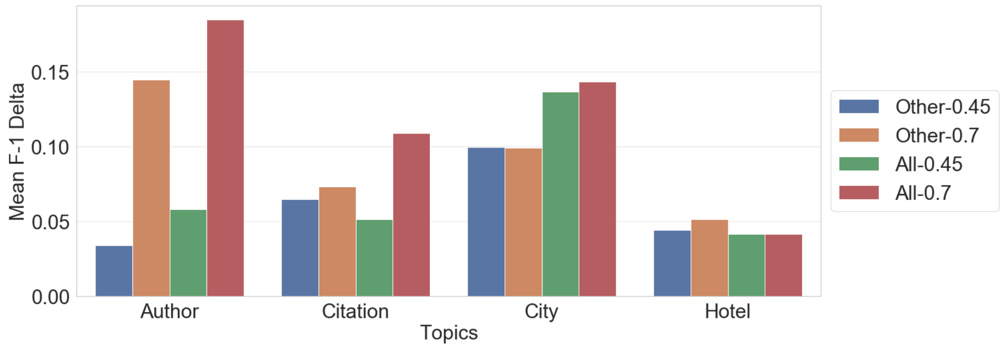


Figure 6.1: Average Comparison Learned Matching Rule & Best Baseline per Topic

In general, Figure 6.1 sums up the general findings of RQ1 regarding the matching rule's general performance:

- On average, the matching rules outperform the best baseline in terms of the F score.
- Apart from the topic hotel, a positive impact of the different silver standard can be observed.
- Apart from the topic hotel, the average performance of the matching rules learned on the 'All' matching task is better than the average performance of the other matching rules.

The main consequence drawn from these findings is that the transfer of the matching rules learned on the difficult silver standards is analysed in depth.

6.2 Research Question 2

RQ2: To which extent can entire matching rules be transferred?

To answer this question, each of the following sections analyses the transferred matching rules of one topic.

6.2.1 Author

Table 6.5 presents the performance scores of the comparison between the transferred matching rules and the best baseline for the respective matching task.

MT_T	**MT_S**	**MR**	**Pr**	**Rec**	**F**	**BBL**	**Pr-△**	**Rec-△**	**F-△**
DNB	VIAF	DT	0.881	0.952	0.915	BoW	0.118	-0.032	0.056
DNB	VIAF	LR	0.950	0.919	0.934	BoW	0.188	-0.065	0.075
DNB	WD	DT	0.962	0.806	0.877	BoW	0.199	-0.177	0.018
DNB	WD	LR	0.978	0.726	0.833	BoW	0.216	-0.258	-0.026
DNB	All	DT	0.967	0.952	0.959	BoW	0.205	-0.032	0.100
DNB	All	LR	0.937	0.952	0.944	BoW	0.174	-0.032	0.085
VIAF	DNB	DT	0.984	0.606	0.750	BoW	0.130	-0.162	-0.059
VIAF	DNB	LR	1.000	0.424	0.596	BoW	0.146	-0.343	-0.213
VIAF	WD	DT	1.000	0.626	0.770	BoW	0.146	-0.141	-0.038
VIAF	WD	LR	1.000	0.869	0.930	BoW	0.146	0.101	0.121
VIAF	All	DT	1.000	0.960	0.979	BoW	0.146	0.192	0.171
VIAF	All	LR	1.000	0.980	0.990	BoW	0.146	0.212	0.181
WD	DNB	DT	0.809	0.961	0.878	Label	0.139	-0.020	0.082
WD	DNB	LR	0.789	0.700	0.742	Label	0.119	-0.281	-0.054
WD	VIAF	DT	0.870	0.985	0.924	Label	0.200	0.005	0.128
WD	VIAF	LR	0.778	1.000	0.875	Label	0.108	0.020	0.079
WD	All	DT	0.935	0.995	0.964	Label	0.265	0.015	0.168
WD	All	LR	0.863	0.990	0.922	Label	0.193	0.010	0.126
All	DNB	DT	0.918	0.839	0.876	BoW	0.242	-0.081	0.097
All	DNB	LR	0.912	0.667	0.770	BoW	0.236	-0.253	-0.009
All	VIAF	DT	0.910	0.973	0.940	BoW	0.234	0.054	0.161
All	VIAF	LR	0.905	0.973	0.938	BoW	0.229	0.054	0.159
All	WD	DT	0.974	0.806	0.882	BoW	0.298	-0.113	0.103
All	WD	LR	0.982	0.860	0.917	BoW	0.306	-0.059	0.138

Table 6.5: Comparison Transferred Matching Rule & Best Baseline - Topic *Author*

All transferred matching rules outperform the baseline on the target matching tasks in terms of the percision-△ score between 0.108 and 0.308. At the same time, especially the matching rules learned on the matching task DNB end up with a negative recall-△. Hence, the matching task DNB seems to be different to all other matching tasks. An explanation can be found in the data profiling, which reveals that the label is maintained differently and the gender values are maintained in German.

The matching tasks DBpedia-VIAF (VIAF) and DBpedia-Wikidata WD seem to be related, because 3 of the 4 transferred matching rules lead to a positive F-△. Only the Decision Tree transferred from WD to VIAF ends up with a slightly negative F-△ of 0.038. The higher number of initially provided positive examples by WD is not necessarily a benefit for the transfer of matching rules.

The matching rule learned on the matching task DBpedia-All has a positive F-△ on all target matching tasks. This matching rule seems to generalise best within the topic *Author*.

6.2.2 Citation

Table 6.6 presents the performance scores of the comparison between the transferred matching rules and the best baseline for the respective matching task.

MT$_T$	**MT$_S$**	**MR**	**Pr**	**Rec**	**F**	**BBL**	**Pr-△**	**Rec-△**	**F-△**
ACM	SL	DT	0.694	0.806	0.746	LC	0.020	-0.129	-0.038
ACM	SL	LR	0.571	0.903	0.700	LC	-0.103	-0.032	-0.084
ACM	All	DT	0.800	0.903	0.848	LC	0.126	-0.032	0.065
ACM	All	LR	0.659	0.935	0.773	LC	-0.015	0.000	-0.010
SL	ACM	DT	0.447	0.410	0.428	Label	0.001	-0.439	-0.157
SL	ACM	LR	0.457	0.427	0.442	Label	0.011	-0.423	-0.143
SL	All	DT	0.458	0.699	0.553	Label	0.011	-0.151	-0.032
SL	All	LR	0.466	0.900	0.614	Label	0.020	0.050	0.029
All	ACM	DT	0.689	0.677	0.683	Label	0.179	-0.161	0.049
All	ACM	LR	0.677	0.677	0.677	Label	0.168	-0.161	0.043
All	SL	DT	0.529	0.726	0.612	Label	0.020	-0.113	-0.022
All	SL	LR	0.500	0.871	0.635	Label	-0.010	0.032	0.001

Table 6.6: Comparison Transferred Matching Rule & Best Baseline - Topic *Citation*

The performance scores reveal that the matching tasks DBLP-ACM (ACM) and DBLP-Scholar (SL) are dissimilar. As a first hint for this dissimilarity, the matching rules learned on the matching task ACM perform badly on the matching task SL with F-△ scores around -0.15. Vice-versa the matching rules learned on the matching task SL yield negative F-△ scores on the target matching task ACM as well. The difference is that the F-△ scores are only 0.038 and 0.084, even though the absolute scores of the baseline are comparably high on the target matching task ACM. A partial explanation can be found in the analysis of the matching rules learned on ACM. This analysis shows that the attribute *year* is the most important attribute (DT: 0.246, LR: 2.677). At the same time, the heterogeneity score of the attributes *year* of the data set Scholar is above 0.6. Consequently, the transferred matching rule fails on the target table Scholar. The high heterogeneity score on the target table combined with a high attribute importance seems may signal a transfer boundary.

A side effect of the above mentioned transfer problems is that the matching rule learned on the matching task DBLP-All (All) seems to be only comparable to the baselines. The average F-△ scores of the learned matching are close to 0.0.

6.2.3 City

Table 6.7 presents the performance scores of the comparison between the transferred matching rules and the best baseline for the respective matching task.

The transfer results for matching rules show a clear difference among the matching tasks. Matching rules learned on the matching task DBpedia - WebTable (WT) perform poorly when transferred to any other matching task. Even on the matching task DBpedia-All, the matching rules learned for the matching task WT yield Precision-△ scores of -0.174 and -0.284. Vice versa the matching rules learned on the matching tasks DBpedia - GeoNames (GN) and DBpedia - Wikidata (WD) yield negative F-△ scores of up to -0.516. This is a hint for the dissimilarity of WT and the matching task GN and WD.
GN and WD seem to be similar. All transferred matching rules yield Precision-△ scores around 0.2 and F-△ scores of around 0.1. It seems that both matching tasks benefit from the transferred matching rules. Hence, the Logistic Regression learned on GN and transferred to WD yields the highest F-△ (0.131) of all transferred matching rules in this thesis.

The matching rules learned on the matching task DBpedia-All (All) draw a different picture. Both only slightly outperform the baseline on the matching task WT, which may be a sign for the dissimilarity of this matching task compared to the other matching tasks. For the other two matching tasks GN and WD, the transferred Decision Tree outperforms the Logistic Regression by 0.06 (GN) and 0.13 (WD) in terms of the precision-△. Thereby, the F-△ score of the Decision Tree is comparable to the F-△ score of the respective matching rules learned on GN

MT_T	MT_S	MR	Pr	Rec	F	BBL	Pr-△	Rec-△	F-△
GN	WT	DT	0.631	0.913	0.746	BoW	-0.136	-0.083	-0.120
GN	WT	LR	0.359	0.493	0.416	BoW	-0.408	-0.502	-0.451
GN	WD	DT	0.998	0.941	0.969	BoW	0.231	-0.054	0.102
GN	WD	LR	0.998	0.956	0.976	BoW	0.231	-0.039	0.110
GN	All	DT	0.979	0.972	0.976	BoW	0.212	-0.023	0.109
GN	All	LR	0.918	0.996	0.955	BoW	0.151	0.001	0.089
WT	GN	DT	0.557	0.800	0.657	Label	-0.030	0.000	-0.020
WT	GN	LR	0.574	0.491	0.529	Label	-0.012	-0.309	-0.148
WT	WD	DT	0.625	0.182	0.282	Label	0.038	-0.618	-0.395
WT	WD	LR	0.714	0.091	0.161	Label	0.128	-0.709	-0.516
WT	All	DT	0.605	0.891	0.721	Label	0.018	0.091	0.044
WT	All	LR	0.593	0.927	0.723	Label	0.006	0.127	0.046
WD	GN	DT	0.922	0.999	0.959	BoW	0.198	-0.001	0.119
WD	GN	LR	0.945	0.999	0.971	BoW	0.220	-0.001	0.131
WD	WT	DT	0.663	0.927	0.773	BoW	-0.061	-0.073	-0.067
WD	WT	LR	0.508	0.814	0.626	BoW	-0.216	-0.186	-0.214
WD	All	DT	0.974	0.995	0.984	BoW	0.249	-0.005	0.144
WD	All	LR	0.842	1.000	0.914	BoW	0.117	0.000	0.074
All	GN	DT	0.811	0.933	0.868	BoW	0.012	0.261	0.137
All	GN	LR	0.862	0.830	0.846	BoW	0.063	0.158	0.115
All	WT	DT	0.624	0.927	0.746	BoW	-0.174	0.255	0.016
All	WT	LR	0.515	0.745	0.609	BoW	-0.284	0.073	-0.121
All	WD	DT	0.951	0.703	0.808	BoW	0.152	0.030	0.078
All	WD	LR	0.983	0.685	0.807	BoW	0.184	0.012	0.077

Table 6.7: Comparison Transferred Matching Rule & Best Baseline - Topic *City*

and WD. This could indicate that the Decision Tree is better able to separate the instances from WT than the Logistic Regression.

6.2.4 Hotel

Table 6.8 presents the performance scores of the comparison between the transferred matching rules and the best baseline for the respective matching task.

Using the performance scores, it is possible to identify the two similar matching tasks revngo-nighttours (NT) and revngo-tourristlink (TL). Both matching tasks benefit in terms of the recall-△ score of around 0.2 by the transfer of the matching

MT_T	MT_S	MR	Pr	Rec	F	BBL	Pr-△	Rec-△	F-△
IHG	NT	DT	0.556	0.926	0.694	Label	-0.141	0.074	-0.072
IHG	NT	LR	0.587	1.000	0.740	Label	-0.110	0.148	-0.027
IHG	TL	DT	0.512	0.815	0.629	Label	-0.185	-0.037	-0.138
IHG	TL	LR	0.636	0.778	0.700	Label	-0.061	-0.074	-0.067
IHG	All	DT	0.641	0.926	0.758	Label	-0.056	0.074	-0.009
IHG	All	LR	0.650	0.963	0.776	Label	-0.047	0.111	0.009
NT	IHG	DT	0.769	0.500	0.606	Label	-0.231	-0.300	-0.283
NT	IHG	LR	0.938	0.750	0.833	Label	-0.062	-0.050	-0.056
NT	TL	DT	0.952	1.000	0.976	Label	-0.048	0.200	0.087
NT	TL	LR	0.952	1.000	0.976	Label	-0.048	0.200	0.087
NT	All	DT	1.000	1.000	1.000	Label	0.000	0.200	0.111
NT	All	LR	1.000	1.000	1.000	Label	0.000	0.200	0.111
TL	IHG	DT	0.833	0.882	0.857	BoW	-0.049	0.000	-0.025
TL	IHG	LR	0.833	0.882	0.857	BoW	-0.049	0.000	-0.025
TL	NT	DT	0.850	1.000	0.919	BoW	-0.032	0.118	0.037
TL	NT	LR	0.850	1.000	0.919	BoW	-0.032	0.118	0.037
TL	All	DT	0.850	1.000	0.919	BoW	-0.032	0.118	0.037
TL	All	LR	0.850	1.000	0.919	BoW	-0.032	0.118	0.037
All	IHG	DT	0.833	0.784	0.808	Label	-0.100	-0.039	-0.067
All	IHG	LR	0.865	0.882	0.874	Label	-0.068	0.059	-0.001
All	NT	DT	0.847	0.980	0.909	Label	-0.086	0.157	0.034
All	NT	LR	0.850	1.000	0.919	Label	-0.083	0.176	0.044
All	TL	DT	0.825	0.922	0.870	Label	-0.109	0.098	-0.005
All	TL	LR	0.855	0.922	0.887	Label	-0.079	0.098	0.012

Table 6.8: Comparison Transferred Matching Rule & Best Baseline - Topic *Hotel*

rules.

The matching rules learned on the matching task revngo-ihg (IHG) generate negative F-△ scores on all target tasks. Interestingly, when comparing the F score on the target matching tasks, the absolute F score is higher in most cases than in the source matching task ihg. Only the Decision Tree transferred to the matching task NT performs worse in terms of the F score. It is possible to see, how difficult this matching task is.

The matching rule learned on the matching task revngo-all (All) yields positive

F-△ scores on the matching tasks NT and TL. The Decision Tree transferred to the matching task NT is especially important to mention, as it outperforms even the Decision Tree learned for the source matching task NT in terms of the F-△ by 0.023. Hence, the transferred rule outperforms the matching rule learned based on the given matching task. This indicates that the matching NT benefits from the transfer. On the matching task IHG, the performance of the 'All' matching rule is comparable to the baseline. These observations suggest that the 'All' matching rules generalise best for the topic *Hotel*.

6.2.5 Summary

To summarise all results of this section, all matching rules are grouped by topic and source matching task. Afterwards, the mean F-△ is calculated for each group. Figure 6.2 displays the corresponding aggregation sorted by F-△ per topic.

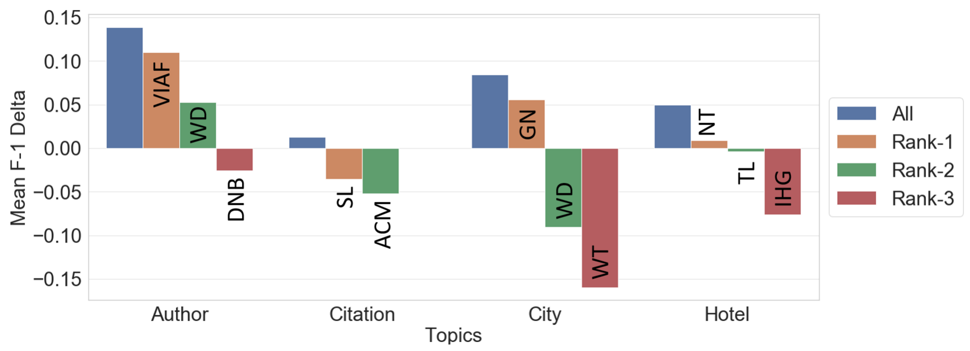


Figure 6.2: Average Comparison Transfer Matching Rule & Best Baseline per Topic

Figure 6.2 underlines the following key findings of RQ2:

- For the topics author, city and hotel two similar matching tasks are identified. These similar matching tasks support each other, which leads to higher average F-△ score. The best transferred rule outperforms the best baseline by a F-△ of 0.131.
- The matching rule learned on instances from all matching tasks seems to generalise best across the given matching tasks.
- The most dissimilar matching tasks in terms of data profiling seem to lead to the highest negative F-△ score.

6.3 Research Question 3

RQ3: Is it more beneficial to transfer only parts of the matching rule instead of the entire rule?

To answer this question, a focus is set on interesting findings. Due to their high number, not all importance scores of all matching rule parts can be presented.

6.3.1 Missing Attributes

In some Identity Resolution scenarios, in which Transfer Learning may be beneficial, the feature spaces are different. This scenario is found in the matching tasks DNB of the topic *Author* and in the matching task WT of the topic *City*. In both matching tasks, one or two attributes are missing. If an attribute is missing in a table, an empty attribute is added to the table to align the schemata. The influence on the transferability of a matching rule with an empty can be seen in the tables 6.9 and 6.10.

Author

Table 6.9 shows the Attribute Importance of Decision Trees in the topic *Author*.

	$\mathbf{R}_{mean}$	$\mathbf{I}_{mean}$	$\mathbf{R}_{std}$	$\mathbf{I}_{std}$
Attribute				
record	2.00	0.424	1.225	0.288
birthdate	1.75	0.407	0.829	0.277
label	2.50	0.130	0.500	0.090
deathdate	3.50	0.021	0.500	0.009
gender	4.00	0.011	0.000	0.000

Table 6.9: Attribute Importance for Decision Trees - Topic *Author*

The attribute *work* is not used by any Decision Tree, so the standard deviation is 0. It seems like the attribute *work* is not useful for the Decision Tree in the topic *Author*. This hint is supported by the overall low average density score (0.358) found in the data profiling section for the attribute work. Furthermore, this finding explains why the performance of the Decision Tree models transferred to the matching task DNB seems not to suffer from a negative transfer as mentioned in section 6.2. The attribute *work* is empty in the domain DNB, because it is not acquired during the initial data collection.

Consequently, this hints that adding a dummy attribute to fulfil technical requirements in cases where the importance of an attribute is low is in fact a valid solution.

City

Table 6.9 shows the Attribute Importance of Decision Trees in the topic *City*.

	$\mathbf{R}_{mean}$	$\mathbf{I}_{mean}$	$\mathbf{R}_{std}$	$\mathbf{I}_{std}$
Attribute				
longitude	2.000	0.469	0.816	0.355
record	2.333	0.413	1.247	0.312
label	2.250	0.159	0.829	0.113
latitude	4.000	0.141	2.160	0.192
population	3.750	0.028	1.090	0.009
country	5.667	0.005	0.471	0.000

Table 6.10: Attribute Importance for Decision Trees - Topic *City*

The coordinate attribute *latitude* has the highest attribute importance in the topic *City* for Decision Trees. This can be explained by the overall high density across all data set tables containing the attribute *latitude*. At the same time, the attribute *latitude* has the lowest uniqueness score compared to the other attributes and especially the longitude in the KB table.

The attributes *latitude* and *longitude* are empty in the WebTable data table. As these attributes are important for the learned Decision Tree, a performance drop is observed when transferring any matching rule to the matching task WebTable as explained in section 6.2. This is a hint for a transfer boundary. If an important attribute of the transferred matching rule is missing, this can lead to a performance drop of the transferred matching rule.

6.3.2 Thresholds and Similarity Scores

When analysing the impact of thresholds on the importance of a matching rule parts, a difference between the Logistic Regression and the Decision Tree can be observed. As an example, the thresholds applied to the similarity score of coordinate values are analysed in depth.

Decision Tree

Table 6.11 shows an importance aggregation of matching rule parts grouped by *[attribute, similarity measure, threshold and split value]*. The Decision Tree's split value is used to accurately identify the learned thresholds. Additionally, the list is filtered for the two coordinate attributes *latitude* and *longitude*.

Split Threshold	$\mathbf{R}_{mean}$	$\mathbf{I}_{mean}$	$\mathbf{R}_{std}$	$\mathbf{I}_{std}$
absSim_long-T:0.8/S:true-0.9	2.500	0.421	1.500	0.363
absSim_lat-T:0.8/S:true-0.9	1.000	0.406	0.000	0.000
percSim_long-T:0.8/S:false-1.0	2.500	0.142	0.500	0.038
percSim_long-T:0.8/S:true-1.0	9.333	0.008	1.247	0.004
percSim_lat-T:0.8/S:false-0.9	13.000	0.003	1.000	0.000
absSim_lat-T:0.2/S:false-0.9	17.500	0.002	3.500	0.001
absSim_lat-T:0.6/S:true-0.9	19.000	0.002	4.082	0.001
absSim_long-T:0.2/S:false-0.9	20.000	0.001	0.000	0.000
percSim_long-T:0.8/S:false-0.9	31.000	0.000	0.000	0.000
absSim_lat-T:0.6/S:true-0.0	32.000	0.000	0.000	0.000
percSim_long-T:0.8/S:true-0.9	35.000	0.000	0.000	0.000
percSim_lat-T:0.0/S:true-0.9	38.000	0.000	0.000	0.000

Table 6.11: Split Importance of Coordinate Values for Decision Trees - Topic *City*

The majority of split values applied in the Decision Trees for the coordinates is 0.9 or higher. This hints for the application of a strict threshold to coordinate values when used within a Decision Tree. An explanation may be that a difference of 10% in terms of the absolute similarity of coordinate values is already a far distance on the globe. For the latitude, a difference of 10% is roughly 1100 kilometres depending on the distance to the equator. A hint for the transferability of this finding are the high number of occurrences. Another hint is the high mean importance score of the first three *[attribute, similarity measure, threshold and split value]* groups displayed in Table 6.11. This is slightly diminished by the relatively high the standard deviation score in context of the related mean score.

Logistic Regression

Table 6.12 shows an importance aggregation of matching rule parts grouped by the attribute, similarity measure and threshold. Furthermore, a filter is set on the two coordinate attributes *latitude* and *longitude*. The coordinates seem to be unimportant for the Logistic Regression, because all mean ranks are below or equal to rank 9. No conclusion about thresholds for the coordinate attributes in a Logistic Regression can be maid.

Comparator Threshold	$\mathbf{R}_{mean}$	$\mathbf{I}_{mean}$	$\mathbf{R}_{std}$	$\mathbf{I}_{std}$
percSim_lat-T:0.8	11.000	0.268	3.000	0.023
percSim_lat-T:0.4	16.000	-0.002	0.000	0.000
percSim_lat-T:0.0	9.000	-0.157	0.000	0.000

Table 6.12: Threshold Importance of Coordinate Values for Logistic Regression - Topic *City*

An explanation for the difference of Decision Tree and Logistic Regression lies in the way the features are evaluated. The Decision Tree decides based on one feature at a time. The Logistic Regression evaluates all features simultaneously.

6.3.3 Attribute Record

From the sections 6.1 and 6.2 it is known that the Bag of Words is a strong baseline for many Identity Resolution tasks, because it is often chosen as best baseline. The record attribute represents an aggregation of all attributes as well. Table 6.13 shows the average rank of the attribute record across all matching rules and topics.

The majority of mean ranks for the attribute *record* is low. Consequently, this suggest that the attribute *record* is important for the majority of matching rules. Simultaneously, the standard deviation of the rank is low when set in the context of the mean rank. This indicates that the comparison features derived from the attribute *record* can be transferred. At least this seems to hold for tables with up to seven attributes.

Attribute	Topic	MR	$\mathbf{R}_{mean}$	$\mathbf{R}_{std}$
record	Hotel	LR	1.333	0.471
record	Author	LR	1.500	0.500
record	City	LR	1.750	1.299
record	Author	DT	2.000	1.225
record	Hotel	DT	2.000	1.414
record	Citation	LR	2.000	0.816
record	City	DT	2.333	1.247
record	Citation	DT	4.000	1.000

Table 6.13: Rank of Attribute *Record* across Topics

6.3.4 String Similarity Measures

String is the most used data type in the experiments. Hence, it is worth to analysing the similarity measures used for attributes of data type string in depth. Table 6.14 shows an aggregated overview of string attributes and applied similarity measures from all conducted experiments. In this table a focus is set on *[attribute, similarity measure]* combinations with a mean rank below five.

The highest ranks are achieved by attributes from the topics *Citation* and *Hotel*. In these topics, the *Jaccard* similarity measure is applied in the top ranked attribute - similarity measure combinations. In fact, *Jaccard* and *JaccardOnBiGrams* are applied to the name of a hotel and title of a citation. The initial indexing's goal is to exploit the attributes *name* of the topic *Hotel* and *title* of the topic *Citation*. It seems that *Jaccard* and *JaccardOnBiGrams* are able to detect more similar instances than the overlap similarity applied during indexing. This explains the low heterogeneity scores of the hotel name and the publications title found during data profiling, as well. This is an indication that the silver standard creation can be further improved for these two attributes.

Even though the standard deviations of the top ranked attribute similarity measure combinations are low, it is difficult to derive a hint for the transferability of these matching rule parts. The reason is the bad performance of the learned and transferred matching rules within the topics *Citation* and *Hotel*. In fact, it is difficult to identify any combination of a string attribute and a similarity measure from the given results. It seems that the best similarity measure depends on the string attribute at hand.

AttributeSimilarityMeasure	**Topic**	$\mathbf{R}_{mean}$	$\mathbf{R}_{std}$
Jaccard-hotel_name	Hotel	1.0000	0.0000
JaccardOnBiGrams-title	Citation	1.2500	0.2500
JaccardOnBiGrams-hotel_name	Hotel	1.5000	0.5000
JaccardOnBiGrams-freq._part_name	Hotel	1.5000	0.0000
Jaccard-infreq._part_name	Hotel	2.0000	0.0000
Jaccard-streetaddress	Hotel	2.0000	0.0000
JaroWinkler-postalcode	Hotel	2.0000	0.0000
Equal-infreq._part_name	Hotel	2.0000	0.0000
Jaro-gender	Author	3.0000	0.0000
FirstCharacter-label	City	3.2915	1.0165
Levenshtein-title	Citation	3.5000	0.5000
JaccardOnBiGrams-infreq._part_name	Hotel	3.5000	0.0000
JaccardOnBiGrams-street	Hotel	4.0000	0.0000
Jaro-label	City	4.7500	0.2500

Table 6.14: String Similarity Measures across Topics

Summary

It depends on the context if it more beneficial to transfer only matching rule parts instead of the whole matching rule. This is underlined by the following key findings for RQ3:

- Missing attributes seem to not necessarily hinder the transfer of matching rules. If the missing attribute is not important for the transferred matching rule, an empty dummy attribute can be beneficial.
- When comparing coordinate values, high similarity thresholds seem to be helpful for Decision Trees. This finding can be transferred to other Identity Resolution tasks that have to deal with an attribute of the data type coordinate.
- Using the whole record as input for the comparison vector seems to be beneficial across almost all topics. Hence, other Identity Resolution task may benefit from integrating similarity measures based on the whole *Record*.
- Across topics the best and hence transferable similarity measure for the most used data type string cannot be identified. It seems like the similarity measure depends on the task at hand.

Chapter 7

Discussion

Transferring matching rules across matching task is a promising approach. The results of this thesis show that in cases where the matching tasks are similar, the naive transfer of an entire matching rules outperform the baselines. In the best case the baseline's F score is outperformed by 0.131.

These results can be improved when matching rules are learned across multiple related tasks. The performance results suggest that these matching rules are on average the most robust ones. This is backed by findings in the literature that use a few labelled instances from the target domain to improve matching rules [74].

Simultaneously, the naive transfer of matching rules entails the risk of negative transfer. If for the transferred rule important attributes are missing in the target domain or are too dissimilar, a negative transfer may occur. To prevent a negative transfer, it is useful to determine the attribute importance of a matching rule. If an attribute is important, it is helpful to analyse this attribute in the target domain in depth. If the attribute is too dissimilar in source and target domain according to the proposed heterogeneity measure, a negative transfer is probable. This finding can be aligned with the findings of Ngomo et al. [56].

The naive transfer done in this thesis does not fully consider the marginal probabilities of the target domain. Hence, it is worth considering an enhancement of the presented approaches by more sophisticated transfer learning approaches found in the literature [27, 85].

Another interesting finding is that comparing entire records without considering attribute boundaries, is transferable across matching tasks. This is a hint why deep learning approaches for Identity Resolution that neglect attribute boundaries are successful, as well [74].

Nevertheless, all findings of this thesis are indications, because only eleven matching tasks are analysed. Hence, a bigger study is necessary to statistically verify these findings. However, Neghban et al. [55] use one topic with six data sets to learn different matching rules. Then, all matching rules are evaluated on a single matching task.

Thirumuruganathan et al. [74] evaluate their approach on twelve matching tasks from four different domains. But as all records are mapped via word embeddings into a high dimensional space, it is difficult to determine the influence of the different aspects of the data. With the standardised experimental setup presented in this thesis, the importance of different parts of a matching rule can be determined. Thus, it is possible to draw conclusions about an attribute, a similarity measure or a threshold in a specific context. Consequently, these conclusion can be used to explain the behaviour of a certain matching rule in a defined situtation. For example, the influence of the threshold applied to the similarity score of the attributes *latitude* in the topic *City* can be analysed in detail.

A downside of these many different matching tasks is that not all positive examples used to learn matching tasks are manually checked. The bases for some of the matching tasks are positive examples derived from *owl:sameAs* links. Studies show that these links are not always accurate [59, 23]. Consequently, the ground truth used to evaluate the transferred rules may contain errors. On top a silver standard creation is automatically done introducing another source of errors. These errors are justifiable by the acquired performance scores. Furthermore, the performance results of the matching rules generated based on *owl:sameAs* links produce good and transferable matching rules in the topics *Author* and *City*.

Simultaneously, the manually annotated topics *Citation* and *Hotel* suffer from quality issues. And especially for the *Hotel* topic, the number of positive examples is very small. Consequently, it seems that the training sets do not cover the variety of data characteristics. Hence, the learned matching rules generalise badly. A solution may be to improve data pre-processing and to increase the number of labelled instances.

As the silver standards are automatically generated the training set's difficulty can be influenced. Hence, the findings of Köpcke et al. [34] are applied to increase the quality of learned matching rules by increasing the difficulty of the training set.

Chapter 8

Conclusion

8.1 Summary

Transfer Learning is a beneficial enhancement for Identity Resolution. The conducted experiments show that it is possible to naively transfer entire matching rules in some cases. In one case the transferred matching rule outperformed the best baseline's F score by 0.131.

This thesis contributes a standardised experimental setup for analysing the importance of different parts of a matching rule. Thereby, it is possible to find hints for characteristics of matching rules that can be transferred across different matching tasks. For example, the experiments show that it is beneficial to use similarity measures based on entire records in the context of Identity Resolution. The experimental setup includes an algorithm to generate configurable silver standards for specific matching tasks based on corresponding positive examples. The performance results suggest that more difficult training sets are beneficial when learning matching rules.

8.2 Future Work

Results of this thesis suggest that matching rules learned across multiple matching tasks generalise best. In the future this naive combination of training sets can be enhanced by existing approaches for transferring knowledge. Thereby, the pool of existing labelled instances in the source domain is better exploited to learn matching rules for a target domain. Furthermore, the findings of this thesis should be statistically verified on a bigger set of experiments. These experiments should be built on manually constructed gold standards to improve the data quality.

Bibliography

[1] Arvind Arasu, Michaela Götz, and Raghav Kaushik. On Active Learning of Record Matching Packages. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 783–794, New York, NY, USA, 2010. ACM.

[2] A. Arnold, R. Nallapati, and W. W. Cohen. A Comparative Study of Methods for Transductive Transfer Learning. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 77–82, October 2007.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, Lecture Notes in Computer Science, pages 722–735. Springer, Berlin, Heidelberg, November 2007.

[4] Mikhail Bilenko and Raymond J. Mooney. On Evaluation and Training-Set Construction for Duplicate Detection. In *Proceedings of the Kdd-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation, Washington Dc*, pages 7–12, 2003.

[5] Bilenko Mikhail, Basu Sugato, Sahami Mehran. Adaptive Product Normalization: Using Online Learning for Record Linkage in Comparison Shopping. In William J. MacKnight and Montgomery T. Shaw, editors, *Introduction to polymer viscoelasticity*, pages 1–6. Wiley-Interscience, Hoboken, N.J, 2005.

[6] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. pages 120–128. Association for Computational Linguistics, July 2006.

[7] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[8] Michael J. Cafarella, Alon Halevy, Zhe Daisy Wang, Eugene Wu, and Yang Zhang. *WebTables: Exploring the Power of Tables on the Web*. National Fire Protection Assoc, Quincy MA, 2008.

[9] P. Christen. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24(9):1537–1555, September 2012.

[10] Peter Christen. A Comparison of Personal Name Matching: Techniques and Practical Issues. In *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 290–294.

[11] Peter Christen, editor. *ICDM workshops 2006: Proceedings : 18 December, 2006, Hong Kong, China ; [Sixth IEEE International Conference on Data Mining - workshops]*. IEEE Computer Society, Los Alamitos, Calif., 2006.

[12] Peter Christen. *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection*. Data-centric systems and applications. Springer, Berlin and New York, 2012.

[13] Peter Christen and Karl Goiser. Quality and Complexity Measures for Data Linkage and Deduplication. In Fabrice Guillet and Howard J. Hamilton, editors, *Quality measures in data mining*, volume 43 of *Studies in computational intelligence*, pages 127–151. Springer, Berlin and London, 2007.

[14] Munir Cochinwala, Verghese Kurien, Gail Lalk, and Dennis Shasha. Efficient data reconciliation. Information Sciences 137.1, 2001.

[15] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Self-taught Clustering. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 200–207, New York, NY, USA, 2008. ACM.

[16] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Elsevier Science, Saint Louis, 2014.

[17] Uwe Draisbach and Felix Naumann. DuDe: The Duplicate Detection Toolkit. page 7, Singapore, September 2010.

[18] M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid. TAILOR: a record linkage toolbox. In *Proceedings 18th International Conference on Data Engineering*, pages 17–28, February 2002.

[19] Ivan P. Fellegi and Alan B. Sunter. A Theory For Record Linkage. 1969.

[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407, April 2000.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[22] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[23] Harry Halpin, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness, and Henry S. Thompson. When owl:sameAs Isnt the Same: An Analysis of Identity in Linked Data. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web ISWC 2010*, Lecture Notes in Computer Science, pages 305–320. Springer Berlin Heidelberg, 2010.

[24] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques: Concepts and Techniques*. Elsevier professional, s.l., 3. aufl. edition, 2011.

[25] Hannaneh Hajishirzi, Wen-tau Yih, and Aleksander Kolcz. *Adaptive Near-Duplicate Detection via Similarity Learning: SIGIR 2010 Geneva, Switzerland July 19-23, 2010*. Association for Computing Machinery, New York, 2010.

[26] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. *Data quality and record linkage techniques*. Springer, New York, NY, 2007.

[27] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Scholkopf. Correcting sample selection bias by unlabeled data. pages 601–608. MIT Press, December 2006.

[28] Robert Isele and Christian Bizer. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649, 2012.

[29] Robert Isele and Christian Bizer. Active learning of expressive linkage rules using genetic programming. *Web Semantics: Science, Services and Agents on the World Wide Web*, 23:2–15, 2013.

[30] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and Understanding Recurrent Networks. *arXiv:1506.02078 [cs]*, June 2015. arXiv: 1506.02078.

[31] Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44, 2013.

[32] Pradap Konda, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, Vijay Raghavendra, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, and Haojun Zhang. Magellan: toward building entity matching management systems [Technical Report]. *Proceedings of the VLDB Endowment*, 9(12):1197–1208, August 2016.

[33] Pradap Konda, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, Vijay Raghavendra, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, and Haojun Zhang. Magellan: toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208, August 2016.

[34] Hanna Köpcke and Erhard Rahm. Training Selection for Tuning Entity Matching. pages 3–12, January 2008.

[35] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Comparative evaluation of entity resolution approaches with FEVER. *Proceedings of the VLDB Endowment*, 2(2):1574–1577, August 2009.

[36] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of Entity Resolution Approaches on Real-world Match Problems. *Proc. VLDB Endow.*, 3(1-2):484–493, September 2010.

[37] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic Model Trees. *Machine Learning*, 59(1):161–205, May 2005.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[39] Sanghoon Lee, Jongwuk Lee, and Seung-won Hwang. Scalable Entity Matching Computation with Materialization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2353–2356, New York, NY, USA, 2011. ACM.

[40] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, and Christian Bizer. DBpedia A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. page 29, 2012.

[41] Oliver Lehmberg, Alexander Brinkmann, and Christian Bizer. WInte.r - A Web Data Integration Framework. page 4, 2017.

[42] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A Large Public Corpus of Web Tables containing Time and Context Metadata. pages 75–76. ACM Press, 2016.

[43] V. I. Levenshtein. Binary codes with correction for deletions and insertions of the symbol1. 1965.

[44] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding Neural Networks through Representation Erasure. *arXiv:1612.08220 [cs]*, December 2016. arXiv: 1612.08220.

[45] Ee-Peng Lim, Jaideep Srivastava, Satya Prabhakar, and James Richardson. Entity identification in database integration. *Information Sciences*, 89(1):1–38, February 1996.

[46] Martha Fallahay Loesch. VIAF (The Virtual International Authority File) http://viaf.org. *Technical Services Quarterly*, 28(2):255–256, February 2011.

[47] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 431–439. Curran Associates, Inc., 2013.

[48] Yongtao Ma and Thanh Tran. TYPiMatch: Type-specific Unsupervised Learning of Keys and Key Values for Heterogeneous Web Data Integration. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 325–334, New York, NY, USA, 2013. ACM.

[49] Christopher D. Manning. *Introduction to information retrieval*. Cambridge UnivPress, Cambridge [u.a.], repr. edition, 2009.

[50] Neil G. Marchant and Benjamin I. P. Rubinstein. In Search of an Entity Resolution OASIS: Optimal Asymptotic Sequential Importance Sampling. *arXiv:1703.00617 [cs, stat]*, March 2017. arXiv: 1703.00617.

[51] Robert Meusel, Petar Petrovski, and Christian Bizer. The WebDataCommons Microdata, RDFa and Microformat Dataset Series. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandei, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web ISWC 2014*, volume 8796, pages 277–292. Springer International Publishing, Cham, 2014.

[52] Monge A. Matching Algorithms within a Duplicate Detection System. 2000.

[53] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep Learning for Entity Matching: A Design Space Exploration. pages 19–34. ACM Press, 2018.

[54] Felix Naumann and Melanie Herschel. An Introduction to Duplicate Detection. *Synthesis Lectures on Data Management*, 2(1):1–87, 2010.

[55] Sahand N. Negahban, Benjamin I.P. Rubinstein, and Jim Gemmell Gemmell. Scaling Multiple-source Entity Resolution Using Statistically Efficient Transfer Learning. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2224–2228, New York, NY, USA, 2012. ACM.

[56] A. N. Ngomo, J. Lehmann, and M. Hassan. Towards Transfer Learning of Link Specifications. In *2013 IEEE Seventh International Conference on Semantic Computing*, pages 202–205, September 2013.

[57] Stefanie Nowak and Stefan Rger. How Reliable Are Annotations via Crowdsourcing: A Study About Inter-annotator Agreement for Multi-label Image Annotation. In *Proceedings of the International Conference on Multimedia Information Retrieval*, MIR '10, pages 557–566, New York, NY, USA, 2010. ACM.

[58] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[59] Pierre-Henri Paris. Assessing the Quality of owl:sameAs Links. In Aldo Gangemi, Anna Lisa Gentile, Andrea Giovanni Nuzzolese, Sebastian Rudolph, Maria Maleshkova, Heiko Paulheim, Jeff Z Pan, and Mehwish Alam, editors, *The Semantic Web: ESWC 2018 Satellite Events*, Lecture Notes in Computer Science, pages 304–313. Springer International Publishing, 2018.

[60] Petar Petrovski, Volha Bryl, and Christian Bizer. Integrating product data from websites offering microdata markup. In Chin-Wan Chung, Andrei Broder, Kyuseok Shim, and Torsten Suel, editors, *the 23rd International Conference*, pages 1299–1304, 2014.

[61] Joseph J. Pollock and Antonio Zamora. Automatic Spelling Correction in Scientific and Scholarly Text. *Commun. ACM*, 27(4):358–368, April 1984.

[62] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[63] Dietrich Rebholz-Schuhmann, Antonio Jos Jimeno Yepes, Erik M van Mulligen, Ning Kang, Jan Kors, David Milward, Peter Corbett, Ekaterina Buyko, Katrin Tomanek, Elena Beisswanger, and Udo Hahn. The CALBC Silver Standard Corpus for Biomedical Named Entities A Study in Harmonizing the Contributions from Four Independent Named Entity Taggers. page 6, May 2010.

[64] Ritze, Dominique and Christian Bizer. Matching Web Tables To DBpedia - A Feature Utility Study. In *Proc. 20th International Conference on Extending Database Technology (EDBT)*, 2017. OCLC: 986240053.

[65] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To Transfer or Not To Transfer. page 4, 2005.

[66] Jrgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[67] Burr Settles. Active Learning Literature Survey. 2009.

[68] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, October 2000.

[69] Kavitha Srinivas, Abraham Gale, and Julian Dolby. Merging datasets through deep learning. *arXiv:1809.01604 [cs, stat]*, September 2018. arXiv: 1809.01604.

[70] Michael Stonebraker and Ihab F Ilyas. Data Integration: The Current Status and the Way Forward. page 7, 2018.

[71] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A Survey on Deep Transfer Learning. *arXiv:1808.01974 [cs, stat]*, August 2018. arXiv: 1808.01974.

[72] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson Addison Wesley, 2006. Google-Books-ID: KZQ0jgEACAAJ.

[73] Ignacio Terrizzano, Peter Schwarz, Mary Roth, and John E Colino. Data Wrangling: The Challenging Journey from the Wild to the Lake. page 9, 2015.

[74] Saravanan Thirumuruganathan, Shameem A. Puthiya Parambath, Mourad Ouzzani, Nan Tang, and Shafiq Joty. Reuse and Adaptation for Entity Resolution through Transfer Learning. *arXiv:1809.11084 [cs, stat]*, September 2018. arXiv: 1809.11084.

[75] Helge Toutenburg. *Deskriptive Statistik: eine Einfhrung in Methoden und Anwendungen mit R und SPSS*. Springer-Lehrbuch. Springer, Dordrecht ; Heidelberg [u.a.], 7., aktualisierte und erw. aufl. edition, 2009.

[76] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk A Link Discovery Framework for the Web of Data. page 6, Madrid, Spain, April 2009.

[77] Denny Vrandei and Markus Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014.

[78] Zheng Wang, Yangqiu Song, and Changshui Zhang. Transferred Dimensionality Reduction. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 550–565. Springer Berlin Heidelberg, 2008.

[79] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1), December 2016.

[80] William E. Winkler. *String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage*. 1990.

[81] William E Winkler. Overview of Record Linkage and Current Research Directions. page 44, 2006.

[82] Lars Wissler, Mohammed Almashraee, Dagmar Monett, and Adrian Paschke. The Gold Standard in Corpus Annotation. page 4, June 2014.

[83] Su Yan, Dongwon Lee, Min-Yen Kan, and Giles C. Lee. Adaptive Sorted Neighborhood Methods for Efcient Record Linkage. 2007.

[84] William E Yancey. Evaluating String Comparator Performance for Record Linkage. page 42, 2005.

[85] Bianca Zadrozny. Learning and Evaluating Classifiers Under Sample Selection Bias. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 114–, New York, NY, USA, 2004. ACM.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Masterarbeit 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 25.01.2019 Unterschrift