Industry-Scale Duplicate Detection

Melanie Weis[1] Felix Naumann[1]
Ulrich Jehle[2] Jens Lufter[2] Holger Schuster[2]
[1]Hasso-Plattner-Institut
Potsdam, Germany
firstname.lastname
@hpi.uni-potsdam.de
[2]SCHUFA Holding AG
Wiesbaden, Germany
firstname.lastname
@schufa.de

ABSTRACT

Duplicate detection is the process of identifying multiple representations of a same real-world object in a data source. Duplicate detection is a problem of critical importance in many applications, including customer relationship management, personal information management, or data mining.

In this paper, we present how a research prototype, namely DogmatiX, which was designed to detect duplicates in hierarchical XML data, was successfully extended and applied on a large scale industrial relational database in cooperation with Schufa Holding AG. Schufa's main business line is to store and retrieve credit histories of over 60 million individuals. Here, correctly identifying duplicates is critical both for individuals and companies: On the one hand, an incorrectly identified duplicate potentially results in a false negative credit history for an individual, who will then not be granted credit anymore. On the other hand, it is essential for companies that Schufa detects duplicates of a person that deliberately tries to create a new identity in the database in order to have a clean credit history.

Besides the quality of duplicate detection, i.e., its effectiveness, scalability cannot be neglected, because of the considerable size of the database. We describe our solution to coping with both problems and present a comprehensive evaluation based on large volumes of real-world data.

1. DUPLICATE DETECTION AT SCHUFA

Duplicate detection is the process of detecting all cases of multiple representations of same real-world objects, i.e., duplicates in a data source. This problem has been widely examined in the research literature (see survey [9]), and several specialized commercial tools [1, 2, 3] perform duplicate detection especially on customer data. The efforts placed into duplicate detection are motivated by the impact duplicates have in many applications. A typical example is customer relationship management, where a company looses money by sending multiple catalogs to the same person, who in turn is annoyed, lowering customer satisfaction. Another application is data mining, where correct input data is essential to produce useful reports that form the basis of decision mechanisms.

The duplicate detection problem has two aspects: First, the multiple representations are usually not identical but contain differences, such as misspellings, changed addresses, or missing values. This makes it difficult to detect these duplicates, so methods *effectively* detecting duplicates are not trivial to design. Usually, some similarity measure and a threshold are used to compare two records and make a decision. Second, duplicate detection is a very expensive operation, as it requires the comparison of every possible pair of duplicates using the typically complex similarity measure. Clearly, *scalability* is crucial for deduplication of large volumes of data, i.e., for industry-scale duplicate detection.

In this paper, we present how we addressed the problems of effectiveness and scalability for an industry scenario in cooperation with Schufa Holding AG. Schufa Holding AG, as an innovative credit service provider, enables and accelerates credit reporting. To its contract partners, such as banks, savings institutions, trade companies, and related industries, Schufa grants a safe loan extension. To the consumer Schufa means a comfortable and cost-saving way of borrowing. As the market leader for credit information services in Germany, Schufa Holding AG has the largest nationwide data pool with 497 million data sets for the evaluation of the current payment behavior of 64 million adult individuals. The information stored about individuals is sparse and for instance includes information on a person's name, current address, birth date, and credit history. Due to extensive privacy protection laws, German citizens do not have a global ID, such as a social security number, to ease identification. Hence, duplicate detection is necessary.

Currently, duplicates in the Schufa database are detected in two different processes. The first process controls data entry by checking whether similar tuples already exist in the database before inserting it as a new person. The second process is triggered by incoming queries. When for instance a bank issues a query for a new customer, the queried entity is compared to persons in the database. If multiple similar persons are found in the database, an expert at Schufa decides which person in the search result corresponds to the query. If the user decides that several results match the query, the different representations, i.e., the duplicates, are fused into a single representation.

In contrast to Schufa's current duplicate detection approaches based on ad-hoc search of duplicates at runtime, the goal addressed in this paper is to perform batch dupli-

PVLDB '08, August 23-28, 2008, Auckland, New Zealand

cate detection on the entire database, independent of other, often time critical business processes. This proactive and systematic search of duplicates aims at further improving data quality, so effectiveness is the primary focus.

Contribution. We present our solution to the batch duplicate detection problem on the Schufa database. Our solution for effective duplicate detection is based on DogmatiX [14], a domain-independent algorithm initially proposed for XML duplicate detection. We adapted DogmatiX to the relational scenario and extended it to further include domain knowledge in the form of rules. To efficiently detect duplicates, we employed the multi-pass Sorted Neighborhood Method (SNM) [10] with the improvement of a key definition process with effectiveness guarantees (compared to the effectiveness obtained when performing all pairwise comparisons). We evaluated our methods on large real-world data sets, consisting of samples of the Schufa database ranging from 100,000 to 10 million customer records. The results show that our methods are very effective in detecting duplicate persons, while being scalable to large amounts of data.

Structure. We first present an overview of the duplicate detection process we devised for Schufa in Sec. 2. In Sec. 3 and Sec. 4, we show how we extended two components of DogmatiX, namely description selection and duplicate classification. Sec. 5 describes how we adapted the sorted neighborhood method to improve scalability and effectiveness. An extensive evaluation is presented in Sec. 6. We briefly discuss related work in Sec. 7 and conclude in Sec. 8.

2. A 10,000 KM PERSPECTIVE

Essentially, our goal is to detect duplicates in the database at Schufa. The system is capable of detecting duplicates in several types of objects including persons, addresses, contracts, birth locations, etc. However, the work presented here primarily focuses on detecting duplicate persons, due to the extensive use of domain-knowledge. In order to detect duplicate persons, we follow the steps below:

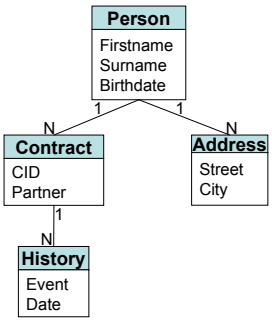
1. **Description selection**: Among 162 attributes available for a given person, we select attributes that are representative of a person, e.g., surname and first name and dismiss irrelevant attributes such as the tuple time stamp. We employ schema-based and instance-based heuristics to automate the process before refining the selection using domain knowledge. Details are described in Sec. 3.

2. **Comparison profile definition**: Using the representative attributes, we define a classifier to classify pairs of persons as duplicates or non-duplicates. In fact, our classifier performs a more fine-grained distinction of duplicates, yielding a total of 11 classes (10 duplicate classes and 1 non-duplicate class). We call this classifier a comparison profile that itself consists of several classifiers. Sec. 4 covers comparison profile definition.

3. **Key definitions**: The next step is to define keys that serve as input to the next phase, i.e., the SNM. Our key definitions are deducted from the comparison profile and actually allow to provide effectiveness guarantees for the SNM, a problem previous research did not address. We provide details in Sec. 5.2.

4. **SNM**: Using our key definitions, we apply multiple passes of the SNM. Because the comparison profile consists of several classifiers, different strategies defining when which classifier is applied are conceivable. We present two strategies in Sec. 5.3.

Example. The process described above is in principal applicable to any domain, however, in the context of the Schufa project, we focus on the domain of detecting duplicate persons in the Schufa database. To illustrate this domain and to support the discussion throughout this paper, consider the following example.



(a) UML diagram of entities and relationships

Person

PID	Firstname	Surname	Birthdate
1	John	Doe	29.03.1970
2	Jonathan	Doe	29.03.1907
3	Michael	Mustermann	

Contract

CID	Partner	PID
123	NZ Bank	1
456	National Bank of New Zealand	2
789	Foreign Bank	3

Address

Street	City	PID
Main Street 1	Auckland	1
Large Place 2	Christchurch	1
Large Place 2	Christchurch	2
Dorfstrasse 567	Kleinstadt	3

History

Event	Date	CID
Open	29.03.1988	123
Update Student to Pro	01.01.1993	123

(b) Example of relational data

Figure 1: Example showing relationships and attributes of four object types Person, Address, Contract, and History (a) as well as a corresponding relational data (b).

EXAMPLE 1. *Fig. 1(a) shows four types of objects, their attributes, and their hierarchical relationships: a person has attributes firstname, surname, and birthdate. A person may have addresses and contracts as children. Addresses have attributes street and city whereas contracts consist of a partner and a contract ID (CID). Contracts have a children type that stores the history of a contract, e.g., when a contract is created or dissolved. Fig. 1(b) shows how three persons and related objects may be represented in a relational database. Please note that both the schema and the data do not correspond to the actual Schufa database. Nevertheless, it reflects the main types of objects that our solution focuses on.*

3. DESCRIPTION SELECTION

Description selection deals with the problem of identifying information relevant to duplicate detection. We refer to object representations that are subject to duplicate detection as *candidates*. Attributes representing informative data about a candidate are called *descriptions*. For description selection, we first used schema-based heuristics adapted from DogmatiX, before we additionally applied instance based heuristics and domain-knowledge.

3.1 Description Selection with DogmatiX

For description selection at Schufa, we used and adapted several schema-based *heuristics* and *conditions*, which we proposed for DogmatiX [14], an algorithm and framework to detect duplicates in XML. We briefly summarize them here.

DogmatiX semi-automatically selects descriptions for a given candidate type using heuristics and conditions based on an XML Schema. The heuristics use the hierarchical XML structure to select close descendants or ancestors as descriptions, based on the following intuition: When detecting duplicates for instance among persons, considering their name and addresses is potentially useful whereas their contract history does not clearly distinguish person candidates because it is very usual that a contract is created, changed, and closed. We observed that the farther from the candidate elements are defined in the hierarchy, the less relevant they are to that candidate. Therefore, the heuristics select only close elements as descriptions. Conditions are used to refine the selection of a heuristic, pruning XML elements based on data type, content model, and cardinality as defined by XML Schema attributes minOccurs and maxOccurs.

3.2 Schema-Based Description Selection

Because the Schufa database is a relational database, not all methods for XML description selection described in [14] apply. Whereas the heuristics on descendants are still valid, the content model condition and the cardinality condition are no longer necessary. On the other hand, we can prune attributes that represent a foreign key, because the same information is already stored in the referenced attribute. Consequently, when considering the relational schema of the Schufa database for description selection, our methods reduce to the proximity heuristic, information on data type constraints, and information about foreign key constraints.

EXAMPLE 2. *For person candidates, assume we apply the following schema-based description selection:*

1. *The proximity heuristic prunes descendants not being children, e.g., all attributes of the credit history.*
2. *The data type condition prunes all attributes not having string data type, e.g., the Birthdate.*
3. *Foreign key attributes are pruned, e.g., Contract.PID.*

The resulting set of descriptions for persons in Fig. 1(c) includes attributes Person.Firstname, Person.Surname, Contract.Partner, Address.Street, and Address.City.

However, using the data type condition (2) actually prunes significant attributes. Indeed, if we consider our example, both Birthdate and CID, which for instance corresponds to a bank account number, are pruned although they represent very informative information about a person. On the other hand, the Schufa database contains numerous fields for internal or legal purposes that are of string data type and do not describe the actual candidate.

Clearly, the data type condition is not applicable in the Schufa scenario, where numerical data represents a significant amount of descriptive information. However, by not applying any condition 135 of 162 attributes are selected by the heuristic. These include many non-descriptive attributes and the amount of attributes is too high to be efficiently processed when classifying candidate pairs as duplicates or non-duplicates as described in Sec. 4.4.1. To avoid these problems, we consider data characteristics to further distinguish descriptive and non-descriptive attributes.

3.3 Instance-Based Description Selection

For every attribute of every table, we collect both the frequency distribution of attribute values and NULL values and the relevance of each attribute value according to its IDF. We prune attributes based on the intuition that descriptive attributes contain values that discern the majority of tuples, i.e., that contain many attribute values all having a low frequency. From the 162 initial attributes, the following techniques pruned 62%. Formally, an attribute was pruned from the description selection if:

- Only one distinct value exists.
- More than 95% of all values are equal (e.g., NULL, country = "Germany").
- The IDF distribution shows at least one of the following characteristics:
  - The average IDF is low, i.e., less than half of the maximum possible IDF.
  - The IDF strongly varies depending on the attribute value, i.e., the standard deviation exceeds 20% of the average IDF value.

3.4 Domain-Knowledge

We presented this selection to experts who modified it based on their domain knowledge: Of the pruned attributes, only very few were added back to the description selection based on domain knowledge. For instance, the gender attribute has two possible values, so the average IDF is low and the attribute was pruned. However, the gender attribute allows to specify that any male person cannot be a duplicate of a female person. Therefore, we included gender in a person's description.

A common case of attributes not automatically removed from the description selection are attributes storing dates that are used for internal processes, e.g., the date when a tuple must be deleted according to German law. These attributes have characteristics of a descriptive attribute, i.e., no NULL values, a large number of distinct values that do not occur often and therefore have a high IDF. However, these attributes are not useful in identifying a person, so all attributes storing dates, except the birth-date, were manually pruned from the description selection.

The final description selection for person candidates of the Schufa database consists of 9 attributes from 4 tables. Due to confidentiality concerns, more details on the final selection cannot be disclosed.

4. DUPLICATE CLASSIFICATION

Using our selection of descriptions, we can now compare candidates and decide whether they are duplicates or not. We refer to this process as *duplicate classification*, or classification for short.

4.1 Comparison Profiles

Within the Schufa project, we classify pairs of candidates as duplicates or non-duplicates based on a profile. A *profile* is defined as a global classifier that is in turn based on a sequence of k base classifiers. These base-classifiers include *similarity-based classifiers*, denoted as $\delta^{\approx}(c, c')$, and *rule-based classifiers*. We distinguish two types of rule-based classifiers: *Negative classifiers*, denoted as $\delta^{-}(c, c')$, classify a pair of candidates as non-duplicates if a predicate returns true, and unknown otherwise. Unknown is represented by the class identifier -1. On the other hand, *positive classifiers*, denoted as $\delta^{+}(c, c')$, classify a pair of candidates as duplicates if a predicate returns true, and unknown otherwise.

Applying the base-classifiers in the order defined by the profile to a pair of candidates results in a final classification of that pair. To each of the k base-classifiers there is a corresponding class, identified by an integer $c, 0 \leq i \leq k$. This range is split up as follows between classifiers: If the predicate of a negative classifier holds true, the pair is always assigned to belong to Class 0. That is, Class 0 represents non-duplicates. If the predicate of a positive or similarity-based classifier at position $0 \leq i < k$ holds true, the candidate pair is assigned to belong to that Class $c = i + 1$. Profiles are defined in such a way that the higher position i and thus c is, the more likely it is that the pairs classified into that class are actually duplicates. That is, Class $c = k$, or Class k for short, is the class of sure duplicates.

Fig. 2 depicts how a comparison profile classifies a candidate pair. It takes as input two candidates c and c' and applies these candidates to a sequence of classifiers. If the first classifier, which is a positive classifier in this example, determines that c and c' are duplicates, the duplicate pair is assigned to Class k, which is the highest value and reflects that c and c' are most likely duplicates. If the positive classifier does not determine that c and c' are duplicates, it returns unknown (-1) and gives the pair to the next classifier in the sequence. The final result of a comparison profile is therefore the classification of the first classifier not returning unknown. If after the last base-classifier the classification is still unknown, the candidate pair is classified as non-duplicates (Class 0).

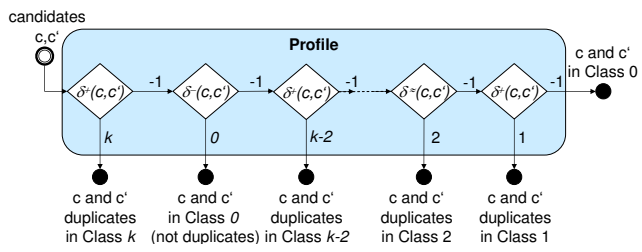


Figure 2: Sample duplicate detection profile.

Clearly, the order in which base-classifiers are set in a profile influences the classification result. This observation will become relevant when discussing methods to improve efficiency in Sec. 5.

In the following sections we describe the principles underlying the classifiers that constitute the *standard profile* developed for Schufa duplicate detection. This profile classifies a pair of candidates into one of 11 classes. It aims solely at obtaining high effectiveness of batch-duplicate detection, and does not consider the functional background that may have caused the duplicate and that potentially requires different processing of the duplicate. For instance, duplicates due to a change of address and duplicates due to a change of last name are potentially classified into the same class by the standard profile, as long as the likelihood that they are duplicates is similar. Opposed to that, we plan to define *specialized profiles* that serve a specific purpose, e.g., that are specifically developed to detect duplicates caused by a change in the surname. The standard profile is very effective for batch duplicate detection, whereas specialized profiles are useful for instance to pipe the detected duplicates to different processes for consolidation.

4.2 Negative Classifiers

A negative classifier classifies pairs of candidates as non-duplicates if a given predicate p holds true. The definition of p is based on domain-knowledge. Formally, if p is the given predicate, then

$$\delta^{-}(c, c') = \begin{cases} 0 & \text{if } p \\ -1 & \text{otherwise} \end{cases}$$

In total, the standard profile includes five negative classifiers. Negative classifiers can improve both effectiveness and efficiency. To improve effectiveness a negative classifier prunes pairs that are similar enough to likely be classified as duplicates by a positive classifier later in the profile, but that cannot be duplicates based on domain-knowledge.

EXAMPLE 3. *When the address of a newly inserted person is unknown, a dummy address is used:* `street = 'UNKNOWN ADDRESS'` and `city = 'UNKNOWN'`. *When comparing two persons where the address is unknown, the addresses seem equal, which adds to the similarity of the two persons. If the persons have some other data in common, e.g., the common surname* `'Smith'`, *their similarity easily exceeds the similarity threshold and the pair is falsely classified as a duplicate. To avoid such cases, we add a negative classifier to the profile, where predicate p holds if two candidates have an unknown address as their only address.*

Negative classifiers improve efficiency by pruning candidate pairs that are unlikely to result in a duplicate. Not computing more complex similarities of such pairs significantly improves performance.

EXAMPLE 4. *It is highly unlikely that two persons with different gender are duplicates, so we define a negative classifier that classifies candidate pairs as non-duplicates if the gender values of the two candidates differ. Because roughly half of all persons are male, and the other half female, this reduces the number of more complex classifications by half.*

Note that the examples above are used for illustration and do not exactly correspond to the real-word classifiers of the actual Schufa standard profile.

4.3 Positive Classifiers

Another constituent of a profile are positive classifiers, which classify candidate pairs as duplicates based on a predicate p:

$$\delta^+(c,c') = \begin{cases} c & \text{if } p \quad (c \text{ is position } i+1 \text{ in profile}) \\ -1 & \text{otherwise} \end{cases}$$

Many positive classifiers are modeled after typical transitions in a person's life, similar to the following example.

EXAMPLE 5. *Jane Smith, born on December, 12th, 1978 and living in Queenstown marries John Doe represented in Fig. 1. She changes her surname to Doe, and moves in with her husband to their new home in Christchurch. She reports the address change under her new surname to her bank, which in turn reports the change to Schufa. However, the address change notification does not include her date of birth, so the following new person is added to the Schufa database: Jane Doe, birthdate unknown, has a former address in Queenstown and a current address at Large Place 2 in Christchurch. For a similarity-based classifier, it is almost impossible to detect this duplicate, because the most distinctive attributes, surname and birthdate are either contradictory or missing. To detect such duplicates, we can define a predicate that classifies two candidates as duplicates if (i) their first name matches, (ii) the data originates from the office reporting address changes, and (iii) some former address of the person matches the current address of the "old" tuple. Note that Schufa maintains a history of all reported addresses of a person.*

In total we defined nine positive classifiers for the standard profile. Each of these classifiers returns a different integer c, $0 < c \leq k$, reflecting how confident we are that the classifier correctly identifies duplicates. Ideally, the higher c is, the more likely the pairs classified by that positive classifier are duplicates. Experiments show that the order we chose for our classifiers is adequate.

4.4 Similarity-Based Classifiers

Finally, the standard profile includes similarity-based classifiers whose definition is based on the domain-independent DogmatiX classifier we briefly summarize next.

4.4.1 DogmatiX Classifier

To detect duplicates, DogmatiX computes a similarity score for a pair of candidates. Generally, if the similarity of two candidates c and c', given by $sim(c,c')$ is above a predefined threshold θ, the pair is classified as a duplicate, otherwise it is classified as a non-duplicate. Formally,

$$\delta(c,c') = \begin{cases} 1 & \text{if } sim(c,c') > \theta \\ 0 & \text{otherwise} \end{cases}$$

where 1 represents a duplicate and 0 a non-duplicate.

To compute the similarity $sim(c,c')$ of two candidates, the DogmatiX similarity measure compares descriptions of same type and aggregates the results of every description type to a total similarity. When comparing descriptions, the following aspects are considered:

- **Data similarity:** A secondary similarity measure calculates the textual similarity of description values and thus identifies descriptions that are considered equal.
- **Data relevance:** To reflect the fact that descriptions do not all have the same importance in describing a candidate, different description values are assigned different weights.
- **Contradictory vs. missing data:** When two candidates both have a description of same type with non-similar values, this value is considered as contradictory and reduces the similarity, whereas a missing description of that type or an empty value does not reduce the two candidates' similarity.

These three aspects are combined in the DogmatiX similarity measures as follows: Intuitively, $sim(c,c')$ captures the weight of similar descriptions relative to the weight of similar plus contradictory descriptions. Similar descriptions are determined based on edit distance; weights are computed using a soft version of the inverse document frequency (IDF) [5], which considers similar strings as equal when computing the IDF of a value.

4.4.2 Similarity Measure Template

The DogmatiX similarity measure uses special functions to implement the different aspects that the similarity measure considers. To define a similarity measure for Schufa, we vary these functions so in this section, we provide a generic similarity measure template of which the DogmatiX similarity measure is a particular instance of.

Formally, let $D(c) = \{d_1, d_2, \ldots, d_n\}$ be the set of descriptions of candidate c, $w(\cdot)$ be a weight function, $descSim(\cdot)$ be a similarity function used to compute the similarity of description values, $type(d)$ be a function returning the description type of description d, and θ_{desc} be a threshold for description similarity. Then, the set $D^{\approx}$ of similar descriptions and the set $D^{\neq}$ of contradictory descriptions are defined as

$$\begin{aligned} D^{\approx}(D(c),D(c')) &= \{(d,d') | descSim(d,d') > \theta_{desc} \\ &\quad \wedge type(d) = type(d')\} \\ D^{\neq}(D(c),D(c')) &= \{(d,d') | d,d' \text{ contradictory} \\ &\quad \wedge type(d) = type(d')\} \end{aligned}$$

The definition of contradictory descriptions requires both descriptions to not be similar to any other description, to be of same type, and to optionally comply to a given predicate p_{contra}, which for DogmatiX for instance requires both description values to not be null. Formally, if

$$\begin{aligned} & \neg\exists d_i | sim(d_i,d') > \theta \wedge \neg\exists d'_j | sim(d,d'_j) > \theta \\ \wedge \quad & type(d_i) = type(d) = type(d'_j) = type(d') \\ \wedge \quad & p_{contra} \\ \Rightarrow \quad & d,d' \text{ contradictory} \end{aligned}$$

Using these definitions, the final similarity is defined as

$$sim(c,c') = \frac{w(D^{\approx}(c,c'))}{w(D^{\approx}(c,c')) + w(D^{\neq})(c,c')} \qquad (1)$$

EXAMPLE 6. *In the example of Fig. 1(c), assume our candidates are the persons John Doe and Jonathan Doe, which we label p_1 and p_2. Throughout this paper, we assume these are duplicates. After description selection as described in Example 2, we obtain*

$$\begin{aligned} D(p_1) &= \{\textit{John, Doe, NZ Bank, Main Street 1,} \\ &\quad \textit{Auckland, Large Place 2, Christchurch}\} \end{aligned}$$

Measure	Similarity	Relevance	Missing descriptions	Children	F-measure
SchufaSim1	equality	unit weight	no influence	no	45%
SchufaSim2	equality	unit weight	reduce similarity	no	61%
SchufaSim3	edit distance for descriptions, containment for children	unit weight	reduce similarity	yes	63%
SchufaSim4	edit distance for descriptions, containment for children	IDF	reduce similarity	yes	64%
DogmatiX	edit distance for all descriptions	soft IDF	no influence	yes	n.a.

Table 1: Different similarity measures characterized along four dimensions and their f-measure on D_{small}.

$$D(p_2) = \{\textit{Jonathan, Doe, National Bank of New Zealand, Large Place 2, Christchurch}\}$$

From these, we obtain

$$D^{\approx}(D(p_1), D(p_2)) = \{\textit{(Doe, Doe), (Large Place 2, Large Place 2), (Christchurch, Christchurch)}\},$$

That is, surname, street, and city attributes are similar between p_1 and p_2. This leaves us with the following contradictory information:

$$D^{\neq}(D(p_1), D(p_2)) = \{\textit{(John, Jonathan), (NZ Bank, National Bank of New Zealand)}\}$$

The remaining street and city provided for p_1 are not considered as contradictory because no non-matched streets and cities exist between $D(p_1)$ and $D(p_2)$. That is, we consider these as information missing in the representation of p_2, which is not considered during similarity computation. The final similarity equals $sim(p_1, p_2) = \frac{3}{3+2} = 0.6$, assuming unit weight for all descriptions.

4.4.3 Similarity Measurement for Schufa

In principal, we can use a classifier based on the DogmatiX similarity measure for duplicate detection at Schufa. However, preliminary experiments showed that it is too slow for the large volume of data we consider, mainly due to the computation of the edit distance for all descriptions and due to the use of the expensive soft-IDF. In consequence, we define several different similarity measures and test these on Schufa data. In devising the similarity measures, we follow the three dimensions presented in Sec. 4.4.1, i.e., data similarity, data relevance, and missing vs. contradictory data. We additionally consider whether a similarity measure considers descriptions from child tables or not. Tab. 1 summarizes how selected similarity measures consider these dimensions.

We applied the different similarity measures on a sample of the Schufa database where all duplicates have been detected manually (see D_{Small} in Sec. 6.1) and measured their effectiveness. Tab. 1 shows the maximum f-measure obtained by the different similarity measures. Intuitively, the higher the f-measure, the better the effectiveness obtained by a similarity measure. Note that whenever the edit distance is used as secondary similarity measure, the threshold is set to $\theta_{desc} = 0.7$, based on experience.

Among all similarity measures we tested and of which SchufaSim1 to SchufaSim4 are examples, SchufaSim4 performs best. Consequently, we choose SchufaSim4 as similarity measure $sim(\cdot)$ for similarity-based classification at Schufa, which is formally defined as

$$\delta^{\approx}(c, c') = \begin{cases} c & \text{if } sim(c, c') > \theta,\ c \text{ is position } i+1 \\ -1 & \text{otherwise} \end{cases}$$

Note that opposed to the DogmatiX classifier, above classifier returns -1 for pairs not classified as duplicates, i.e., it classifies them as unknown and not as non-duplicates. Using this definition, similarity-based classifiers can be put at any position in the classifier sequence defining a comparison profile.

5. SCALABILITY

Duplicate classification, as discussed in the previous section, is in principle applied to every possible pair of candidates. However, this is practically infeasible for large data sets: N tuples in the person table would require $\frac{N \times (N-1)}{2}$ pairwise comparisons. This means roughly 2048×10^{12} pairwise comparisons for the 64 million persons stored in the Schufa database, which does not terminate in reasonable time. Therefore, we devise a strategy to prune a significant number of classifications.

Our method is based on the Sorted Neighborhood Method (SNM) [10], which prunes a large number of comparisons and thus potentially misses some true duplicates. However, as effectiveness remains the primary concern at Schufa, we propose a new method to determine the keys for SNM to minimize this effect.

5.1 Sorted Neighborhood Method

The SNM consists of three phases:

1. Generate a key for every candidate.
2. Sort candidates according to the lexicographical order of their key values.
3. Perform pairwise classifications: Given a window of fixed size w that slides over the sorted sequence of candidates, compare candidates that fall in the same window, i.e., compare the last candidate entering the window with all other candidates already in the window.

Clearly, the smaller the window size w is, the less pairwise comparisons are performed. However, maintaining high effectiveness is critical and essentially depends on the key definition and the window size. The key is ideally defined in such a way that duplicates are sorted close to each other. The window size defines how distant duplicates are allowed to be to still be detected. These problems have been considered before, yielding the multi-pass SNM and the use of transitive closure over duplicate pairs [11]. The multi-pass SNM uses different keys in each pass. This way, different orders

allow comparisons of different pairs over multiple runs, and more duplicates are potentially detected. Using the transitive closure of pairs detected in one run and pairs detected in several runs allows to reduce the window size, because even candidates not lying in the same window can be classified as duplicates based on transitivity.

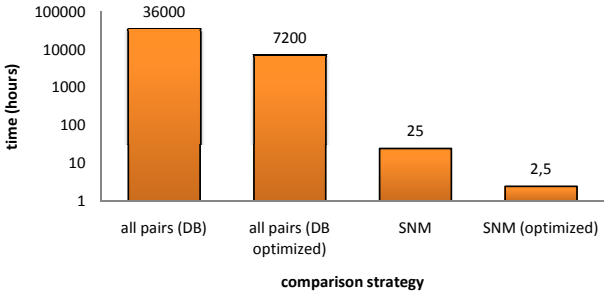


Figure 3: Runtime improvement.

To see how runtime improved when using the SNM on a large sample of the Schufa database (D_{Large} in Sec. 6.1), consider Fig. 3 that shows runtimes of different implementations using a logarithmic scale. Our first attempt to process D_{Large} was to actually compare all candidate pairs. The first implementation basically used a DBMS as a data store and we added indexes to improve query processing. We started to run duplicate detection and based on the number of processed pairs, we extrapolated that comparing all candidate pairs would take 1500 days, which is clearly no acceptable runtime. By optimizing our programm, e.g., by reducing the communication with the database and the amount of exchanged data and by materializing views, we were able to reduce expected runtime to 300 days. By using the SNM with a window size $w = 50$, our first implementation ran for 25 hours and could be improved to 2.5 hours by adequately configuring the database and optimizing the interaction between our prototype and the database. Clearly, the use of the SNM significantly improves scalability but adequate database configuration is also necessary.

5.2 Keys to Maximize Effectiveness

Using the SNM improves efficiency, but it potentially reduces effectiveness. Indeed, when defining a key for the SNM it is essential that this key sorts duplicates close to each other. In general, we do not know in advance which candidates are duplicates, so the key definition is at best an approximation of the ideal key. However, using a profile as defined in Sec. 4, we actually *can* predict most candidates that are potentially duplicates.

- Positive classifiers: When defining a key that sorts duplicates according to the predicate of a positive classifier, it is guaranteed that duplicates are sorted next to each other.
- Similarity-based classifiers: Depending on the similarity measure, we can derive positive classifiers that identify candidate pairs with similarity > 0. These pairs are potential duplicates (depending on threshold θ). We can then define keys for the derived positive classifiers as described above.
- Negative classifiers: A negative rule defines when two candidates cannot be a duplicate, so by negating that rule, we obtain a rule that identifies potential duplicates. Because this rule cannot be used to directly classify a pair as a duplicate (e.g., based only on equal gender) we combine it with positive classifiers or with similarity-based classifiers subsequently applied in the comparison profile.

EXAMPLE 7. *Consider a positive classifier that is based on the following predicate P1: If the surname and the day and month of the birthdate of two candidates are exactly equal, they are classified to be duplicates. To guarantee that duplicates are sorted next to each other, we define a key consisting of the surname, the day of birth, and the month of birth. All candidates that are duplicates according to that predicate are then sorted next to each other. P1 for instance detects the duplicates shown in Fig. 1(c), which have the same key Doe2903.*

In the example above, all candidates with the same key are actually duplicates. In practice, such an ideal key cannot be defined for every predicate. In general, candidates with equal key are only potentially duplicates and, vice versa, duplicates do not necessarily have equal keys. For duplicate detection at Schufa we defined 11 distinct key definitions. As depicted in Fig. 4, these definitions include (i) Keys that correspond to exactly one predicate (key_{n-1} and key_n),(ii) Keys that correspond to more than one predicate (key_3),and (iii) Keys that correspond to only a part of one predicate, the other parts being covered by other keys (key_1 and key_2).

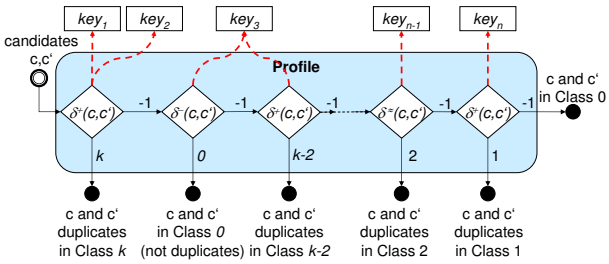


Figure 4: Variants of rule-based key definitions.

EXAMPLE 8. *In addition to the predicate given in Example 7, assume further positive base-classifiers that use the following predicates:*

- *P2: Two persons are duplicates if their last names are equal and if one first name is equal to the first name or the middle name of the other person.*
- *P3: Two persons are duplicates if their surname and their month of birth are equal.*

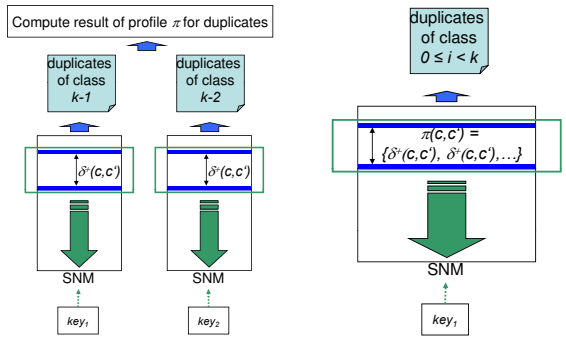
The key provided in Example 7, which concatenates surname, year of birth, and month of birth neither corresponds to P2 nor to P3, because it is too specialized. However, a key consisting of the last name fits all three predicates. Finally, for P2, we reflect the containment of the first name by defining two keys for the predicate. The first key always includes the first token of a first name, and the other one the second token of a first name.

Based on the fact that (sets of) keys are defined for (sets of) base classifiers, some obvious opportunities to improve comparison runtime in one pass of multi-pass SNM arise. We describe these in the next section.

5.3 Multi-Pass Variants

The multi-pass SNM applies the SNM method once for every key. We defined 11 keys for the standard profile, so our multi-pass SNM performs 11 runs. Because keys are defined for specific base classifiers, it may be beneficial to not compare candidates using the complete profile in every run, but to apply only those base classifiers that correspond to the key used in a run. Based on this observation, we distinguish two strategies: a *rule-based strategy* and a *profile-based strategy*. Both strategies are illustrated in Fig. 5.

The rule-based strategy (Fig. 5(a) shows two runs) applies only part of the profile for duplicate classification in one run, and combines the result in a post-processing step. More specifically, the run using key_i uses only the base classifiers for which key_i has been defined. For instance, the run using the order given by key_n of Fig. 4 compares persons based only on the positive classifier returning 1. This comparison strategy potentially detects a pair more than once over several runs, but with different classification results, or finds a pair once but with a non-maximal classification result. As a consequence, all detected pairs need to be reevaluated using the complete profile in order to obtain the correct classification.



(a) Rule-based strategy (b) Profile-based strategy

Figure 5: Comparison strategies.

EXAMPLE 9. *The duplicate of John Doe represented in Fig. 1 can be detected both by a positive classifier δ_1 based on P1 and that classifies duplicates into class i , and a positive classifier δ_2 based on P2 that classifies duplicates into class j. Because P1 is stricter in its classification than P2, we assume $i > j$. If we perform two runs where only δ_1 and δ_2 are applied, respectively, we obtain two possible classifications of the John Doe duplicate and have to determine in a post-processing step that the final result is i.*

Another overhead of this method is that the configuration of duplicate detection becomes more difficult: not only do we have to specify keys, but for every key, we need to provide a different classifier.

The profile-bases strategy (Fig. 5(b) shows one run) does not require the post-processing and the configuration overhead described for the rule-based strategy. Indeed, it compares candidate pairs using the entire standard profile, no matter the key being used. This way, the correct classification result is directly computed. However, the classification step is more complex because 11 instead of one classifier are applied, which may compromise runtime.

Determining which strategy is best is not trivial. The runtime of the profile-based strategy depends on the distribution of duplicates over the different base classifiers. For instance, if the majority of duplicates is detected by the first base classifier, all remaining base classifier are rarely used, which represents an advantage compared to the rule-based classifier that uses all classifiers in different runs. On the other hand, if the majority of duplicates is detected by base classifiers at the end of the sequence, the rule-based strategy potentially terminates faster. Experiments show that the rule-based strategy is more efficient and does not compromise effectiveness in our Schufa scenario.

6. EVALUATION

We now evaluate the methods presented throughout this paper both in terms of effectiveness and in terms of scalability, based on large volumes of real-world data.

6.1 Data Sets and Setup

To evaluate our methods, we use two samples of data collected at Schufa.

1. D_{Small} consists of data about 100,000 randomly selected persons, including their addresses, contracts, etc. In this sample, all duplicate pairs[1] have been identified manually through the existing search processes at Schufa. Therefore, we can measure both recall and precision on this sample, which we use to evaluate the different classifiers.

2. D_{Large} consists of data about 10,000,000 randomly selected persons. In this sample, many duplicate pairs have already been identified manually, but some duplicates remain undiscovered because the corresponding person is not queried. Hence, an evaluation of effectiveness on this data set requires a manual evaluation of the result to determine precision, and recall can only be calculated based on the number of known duplicates. We use this data set to validate the standard profile and to evaluate scalability.

Prior to actual duplicate detection, we performed data standardization, which consisted in removing special characters from string attributes.

We wrote our prototype in Java 1.5 and used IBM DB2 v.8.2 as DBMS. The Java heap size was set to 1.3 GB. We ran our experiments on a PC with 4GB main memory and an Intel Core Duo 6400 CPU with 2.13 GHz. The operating system installed is Windows XP Pro version 2002 with SP2.

6.2 Effectiveness

Our first series of experiments concentrates on effectiveness. We measure effectiveness using recall, precision, and f-measure. Recall measures the fraction of correctly found duplicates, relative to all known duplicates in the data set. Precision, on the other hand, measures the fraction of correctly found duplicates, relative to all found duplicates. The f-measure is the harmonic mean of recall and precision. Ideally, all metrics return 100%, but in practice, an algorithm

[1]actual number cannot be disclosed

finds some false duplicates, and further does not find all duplicates in the data source. Therefore, the goal is to maximize recall and precision (and thus f-measure) to obtain high effectiveness overall.

Experiment 1. From our incremental similarity measure design described in Sec. 4.4, we concluded that SchufaSim4 is the best similarity measure. We now show how adding positive and negative classifiers improves effectiveness. The selection of the best positive and negative classifiers results from a similar strategy than the determination of the best similarity measure.

Methodology. We compare the effectiveness of the following profiles on D_{Small}: First, we apply a profile solely consisting of SchufaSim4 and output duplicate pairs, their similarity as computed by the similarity measure, and whether they are a known duplicate or not. We then evaluate a profile consisting of SchufaSim4 and negative classifiers, called NC. The generated output is similar to the output of SchufaSim4. Finally, we combine SchufaSim4, negative classifiers, and positive classifiers to a profile called PC. Duplicate pairs detected by positive classifiers are assigned the result of the positive classifier. For each of the three profiles, we sort the detected duplicate pairs in descending order of their similarity / classification result and compute recall and precision over the sorted list of pairs. We set the similarity threshold to $\theta = 0.3$ and the edit distance threshold to $\theta_{desc} = 0.7$ (based on experience). Fig. 6 shows the resulting recall-precision curves. The maximum f-measure of the different profiles is 64%, 76%, and 86%, respectively.

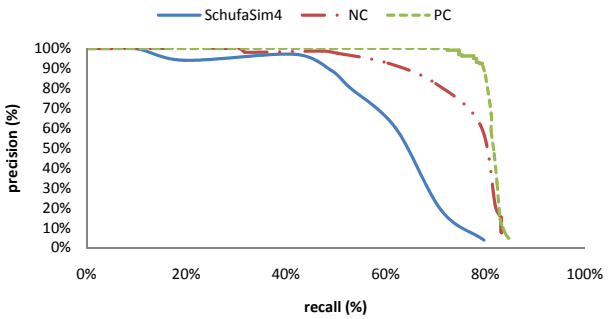


Figure 6: Effectiveness on D_{Small} of different comparison profiles.

Discussion. In Fig. 6, we see that SchufaSim4 reaches a recall of 80%. However, precision steadily drops for recall values higher than 40%. When using negative classifiers, we see that precision is significantly improved compared to SchufaSim4. Recall is slightly improved an now reaches 83% for the following reason: When applying a negative classifier that prunes pairs with different genders, we exclude the gender attribute from the description processed by the similarity measure, because it will be equal for any pair anyway. This alters the result of SchufaSim4, and in the depicted case yields a slightly higher recall. When additionally adding positive classifiers to the profile, the precision is again significantly improved, but recall is comparable to the experiment using SchufaSim4 and negative classifiers because no additional duplicates are found. This essentially means that the positive classifiers reorder pairs in such a way that false positives are sorted after most true positives. As a consequence, the standard profile has a precision of 100% for recall values of up to 72%. Essentially, we steadily improved overall effectiveness by subsequently adding negative and positive classifiers to the classifier using SchufaSim4. The final profile, which we use as standard profile at Schufa, reaches a maximum f-measure of 86% on D_{Small}.

Experiment 2. We now evaluate the effectiveness of the standard profile determined in the previous experiment on D_{Large}.

Methodology. On D_{Small}, we actually performed all pairwise comparisons to evaluate the different profiles. For the larger data set D_{Large}, this is no longer feasible (see Fig. 3), so we use multi-pass SNM with all 11 keys that we defined to fit the standard profile. The window size is set to 50, and we measure effectiveness for both the profile-based strategy and the rule-based strategy. Because not all duplicates are known in D_{Large}, we compute recall based on the number of known duplicates. To evaluate precision, we randomly sampled 100 duplicates of each duplicate class, which we label from 1 to 10, 10 being the class with sure duplicates[2]. Three experts then decided for each pair in the sample whether it is an actual duplicate or not. For the classes with high duplicate confidence ($c \geq 6$), the experts were already satisfied with the corresponding classifier after 20 to 40 pairs. For all remaining classes, all 100 pairs were manually processed. Because the experts used the actual DB at Schufa, and not the snapshot we used for our experiments, some person tuples were already deleted, either because they were recognized as duplicates and merged in the three months lying between our database snapshot and manual evaluation, or because deletion constraints due to Schufa policy applied. Because pairs where one person was deleted could not be clearly classified as duplicates or non-duplicates by the experts, we removed them from the sample and computed the precision on the reduced sample. The recall and precision obtained for each class is depicted in Fig. 7 and the recall-precision curves for both comparison strategies are depicted in Fig. 8. Cumulated over all classes, our standard profile achieves a maximum f-measure of 66% with 53% recall and 90% precision.

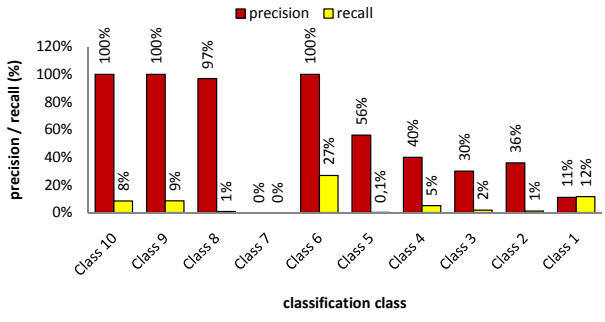


Figure 7: Effectiveness per class on D_{Large}.

Discussion. In Fig. 7, we observe that the profile behaves as expected: classes with high classification result ($c \geq 6$) have a very high precision, which means that we can trust their classification and we can consider the pairs they classify

[2]We do not report the actual classification result ranging from 1 to 15, because this would disclose confidential information on the order of positive and negative base classifiers.

as sure duplicates. Actually, these classifiers alone already detect 47% of all detected pairs. Class 7 does not contain any pair because the pairs it considers as duplicates are already detected by the classes applied first (10, 9, and 8). Consequently, we remove the corresponding classifier from the standard profile. For $c < 6$, we observe that the precision of each class (except for Class 2) drops. From this behavior, we conclude that the order in which the classifiers are specified in the standard profile is adequate. In future work, an interesting question is whether we can find an order of base-classifiers that orders the classifiers in decreasing order of their precision. This is not as simple as switching the classifiers for Class 2 and Class 3, and Class 6 and Class 7. Indeed, classifiers serve as filters to classifiers coming later in the sequence, in the sense that they potentially classify pairs as false positives (reducing their own precision) that would otherwise be found by a subsequent classifier, which would lower its precision if the order was changed.

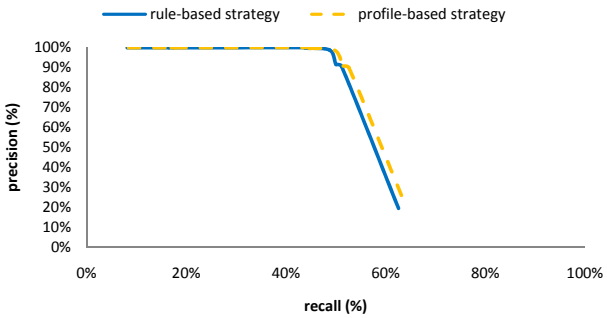


Figure 8: Comp. strategy effectiveness on D_{Large}.

In Fig. 8, we see that both comparison strategies obtain comparable effectiveness. The recall reached by the rule-based strategy, which applies only the base-classifiers for which the key of the corresponding run was designed, is only slightly lower than the recall of the profile-based strategy. In Experiment 5, we see that the rule-based strategy is more efficient than the profile-based strategy, so we can conclude that the profile-based strategy is the most suited.

Compared to the effectiveness on D_{Small}, we obtain a lower recall on D_{Large}. We evaluated our standard profile on a sample of D_{Large} consisting only of known duplicates, and observed that the profile is in principle capable of finding 85% of known duplicates. The lower recall measured for both comparison strategies is explained by the fact that we do not perform all pairwise comparisons using the SNM and we therefore miss some duplicates when using a window size of 50, although we defined suited keys. To counter this effect, we plan to use variable window size in the future.

Experiment 3. Despite a moderate recall, the overall effectiveness is acceptable due to the high precision, which is the basis for subsequent automatic processing of duplicates. We now evaluate how beneficial our approach is for Schufa.
Methodology. Using the same methodology as in the previous experiment, we compare the number of known duplicates that we found per class to the number of additional duplicates found per class, as depicted in Fig. 9.
Discussion. For every class, some duplicates are known, so current duplicate detection methods at Schufa are able to find these duplicates. However, these methods require a

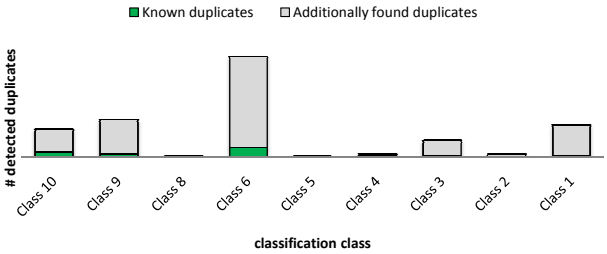


Figure 9: Number of known duplicates found vs. number of all duplicates found on D_{Large}.

query for a duplicated person to detect the duplicate. Our batch duplicate detection approach allows to find these duplicates systematically without necessitating a query. This way, a substantial amount of duplicates can additionally be detected per class, as Fig. 9 shows (also for Class 5, although the absolute number of duplicates in that class is small). Clearly, using such a systematic duplicate detection approach helps to further improve the already high data quality of the Schufa database.

Experiment 4. Having evaluated the different classifiers of the standard profile, we now evaluate the key definitions we devised for the base classifiers, and study how duplicates are distributed over the different SNM runs.
Methodology. For each of the 11 keys we defined for the standard profile, we perform a SNM run with window size 50. Fig. 10 shows the distribution of classification results over each run and thus key using a profile-based comparison strategy. We obtain a similar distribution when using the rule-based strategy, omitted here for brevity.

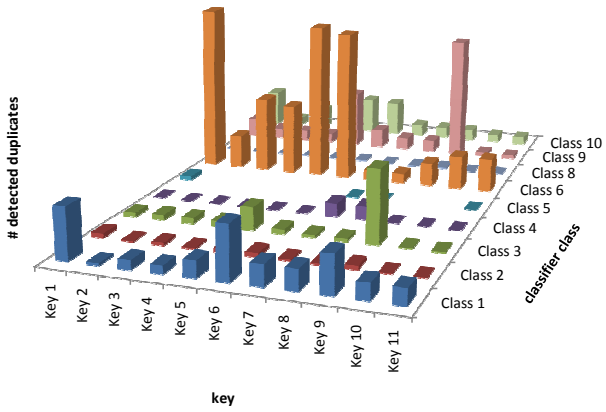


Figure 10: #duplicates per key and class on D_{Large}.

Discussion. The results for the profile-based strategy in Fig. 10 show that each class, except for Class 5, finds duplicates in each run. However, the number of duplicates found per class varies with the key being used. This is explained by the fact that keys are defined for specific classifiers. Essentially, keys with low ID (e.g, Key 1 and Key 2) match classifiers returning a high class (e.g., Class 10 and Class 9)[3]. This trend appears in Fig. 10, where the back left corner and the front right corner cumulate high frequencies for the respective keys and classes. The two exceptions to this trend

[3]The true matching between classes and keys is confidential.

are the number of pairs found for Key 9, Class 9 and for Key 1, Class 1. Because the distribution of the rule-based strategy is almost equal to the distribution of the profile-based strategy, we explain these exceptions by the fact that the detected duplicates can be detected by both the classifier for which the key was designed (the only one being applied by the rule based strategy) and the classifier yielding the final classification.

In summary, experiments show that the standard profile is effective in detecting duplicates at Schufa. The individual classifiers of the standard profile are defined in an adequate order to satisfy the assumption that classifiers applied first are more sure of their classification. Finally, the rule-based strategy obtains almost the same result as the profile-based strategy but requires less time, as we see next.

6.3 Scalability

Effectiveness was the primary concern at Schufa, however, scalability is relevant, too.

Experiment 5. In Sec. 5, we presented two comparison strategies. In this experiment, we compare the overall runtime of both strategies.

Methodology. We measure the runtime of the multi-pass SNM over D_{Large} using all 11 keys for both comparison strategies. The window size is set to $w = 50$. Fig. 11 reports the results.

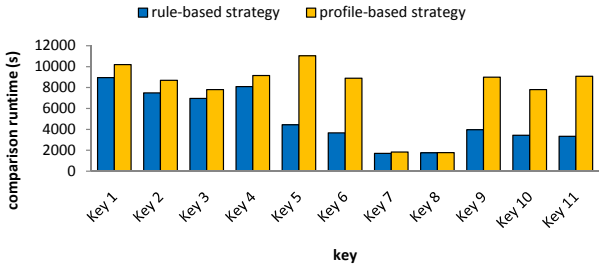


Figure 11: Comp. strategy runtime on D_{Large}.

Discussion. Fig. 11 shows that every run of the rule-based strategy is faster than the corresponding run using the profile-based strategy. Hence, the rule-base strategy is more efficient than the profile-based strategy. We explain this by the reduction of the classifier's complexity and by the fact that the overhead required to compute the correct classification result for a duplicate is negligible. We further observe that the runtime of runs may significantly differ. One reason for this behavior are negative classifiers, which we still consider in the rule-based strategy. These act as filters and reduce the input of base classifiers appearing later in the sequence. For instance, a negative filter applied between the classifiers for which Key 6 is designed and the classifiers for which Key 7 is designed reduces the number of pairs the classifiers corresponding to Key 7 have to consider, which improves the runtime of the run using Key 7 compared to the run using Key 6.

Experiment 6. The last experiment aims at showing how our duplicate detection method scales and shows that it is not only applicable to D_{Small} or D_{Large}, but also applies to the complete Schufa database.

Methodology. We vary the data set size from information about 2 million persons to information about 10 million persons in increments of 2 million. To obtain these data sets, we sample D_{Large}. For each dataset size, we measure the runtime of preprocessing and the runtime of pairwise comparisons over one SNM run that uses Key 7. We report the results in Fig. 12. Note that both key generation and key sorting are decoupled from the actual duplicate detection process because the goal is to perform these asynchronously and incrementally. The preprocessing time shown in Fig. 12 therefore includes only the time to fetch all necessary data from the database into main-memory (which includes candidate IDs, descriptions, IDF weight) with the support of a DBMS. For 10 million persons, key generation takes 30 seconds and sorting takes between 3 and five minutes.

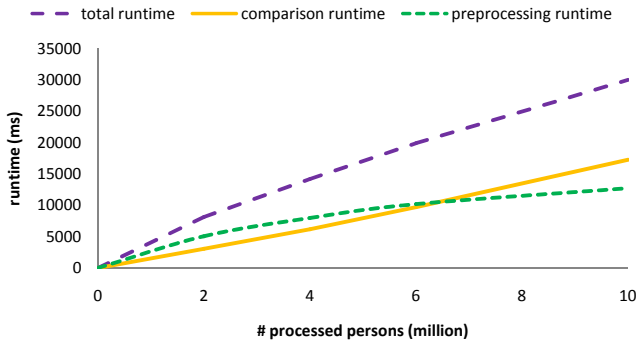


Figure 12: Scalability of Schufa duplicate detection.

Discussion. The comparison phase scales linearly with the number of processed candidates N. This is in accord with the complexity of the SNM comparison phase, which is $O(wN)$. We further observe that the preprocessing time per candidate decreases with increasing number of candidates. This phase makes heavy use of DBMS features and optimizations, so it is difficult to further interpret the result without knowing the internals of the DBMS. Nevertheless, the linear behavior together with the runtimes of Experiment 5 for processing 10 million persons allow us to estimate the runtime of processing 70 million persons (complete Schufa database) to 4.7 days on a one-processor PC, using $w = 50$ and a rule-based strategy. As hardware at Schufa is more advanced, runtime should further improve there.

7. RELATED WORK

Schufa has evaluated some commercial tools for duplicate detection and decided that these did not fit their requirement, either due to unsatisfactory effectiveness or scalability. Therefore, Schufa decided to build their own solution based on up to date research.

There is a large body of research literature on duplicate detection (see survey in [9]) but we only briefly discuss solutions for industry-scale duplicate detection here.

Our prototype extends techniques we developed for DogmatiX [14] and allows efficient and effective duplicate detection. Our prototype is extensible and may use alternative methods for classification [7, 8, 12] and comparison strategy [4, 6, 15]. Our choice for extending DogmatiX was motivated by previous research of ours that showed that DogmatiX is best suited in terms of effectiveness in scenarios similar to the Schufa scenario [13].

A very interesting approach that may be used to further improve our classifier is [8], where rules and similarity measure parameters are first learned and expressed in SQL queries and then combined to obtain a close to optimal rule operator tree, with respect to effectiveness. The drawback of this approach in our context is that the operator trees potentially result in queries that take a long time to execute over all pairs, and applying the queries on candidates selected for comparison by the SNM results in a large communication overhead with the database.

A complementary approach to our comparison strategy is proposed in [15], where the window size of the SNM varies at runtime. This solution is very appealing in our context, because key values occur with different frequencies. For instance, a key consisting of the surname and the firstname requires a large window size when persons with equal frequent names are compared (e.g., thousands of John Smith's are sorted next to each other and all have equal key). On the other hand, when the keys sorted next to each other strongly differ, the window size can be kept small.

Clearly, many techniques can further improve duplicate detection but nevertheless, the current system is already very effective and scalable.

8. CONCLUSION AND OUTLOOK

We presented a solution to duplicate detection, which was specifically developed for a batch duplicate detection process at Schufa Holding AG. This solution is complementary to the already existing duplicate detection processes based on duplicate search at runtime.

Our solution to effectively detect duplicates extends DogmatiX, initially developed to detect duplicates in XML, in two ways: First, we extended description selection by applying instance-based heuristics for automatic description selection and a subsequent manual refinement of the selection based on domain knowledge. The other extension to DogmatiX is a profile that consists of a sequence of domain-dependent rule-based classifiers and a similarity-based classifier. The standard profile classifies pairs of candidates in one of 11 classes. Experiments over a large fraction of the Schufa database show that the standard profile is effective in detecting duplicates, reaching 90% precision for 53% recall when time is critical, and reaching up to 85% recall with adequate window sizes and more time.

To improve efficiency with minimal impact on effectiveness, we devised key definitions tailored to the classification rules of the standard profile. These key definitions guarantee that if candidates are classified as duplicate by a positive rule, the key corresponding to that rule will sort them close together. By coupling key definitions with classifiers, we further reduced classification complexity by using only those classifiers that correspond to the key used in a run. This rule-based strategy significantly improves runtime whereas it does not significantly degrade effectiveness. Using a rule-based strategy, duplicate detection over 10 million persons represented in the Schufa database terminates after 15 hours on an off-the-shelf PC, which is deemed acceptable.

The prototypical implementation of the methods we described in this paper yields satisfactory effectiveness and scalability and is currently in the process of being integrated in the actual system at Schufa. At the same time, we are devising methods to further improve duplicate detection: To improve effectiveness and to facilitate post-processing, we will define specialized profiles that can be used in addition to the standard profile to classify duplicate pairs based on the reason this duplicate may have entered the database, e.g., an unreported address change or a change in surname. Different reasons require different post-processing, which may be more easily automated using specialized profiles.

Acknowledgements. This project is funded by the Schufa Holding AG and we thank the experts who manually classified duplicates and the Insiders employees who are migrating the prototype into the actual Schufa system.

9. REFERENCES

[1] FUZZY! Double by FUZZY! Informatik AG. Details at `http://www.fazi.de/`.

[2] IBM Entity Analytic Solutions (EAS). Details at `http://www-306.ibm.com/software/data/db2/eas/`.

[3] Trillium software system. Details at `http://www.trilliumsoftware.com/de/content/products/index.asp`.

[4] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Conference on Very Large Databases (VLDB)*, Hong Kong, China, 2002.

[5] R. A. Baeza-Yates and B. A. Ribeiro-Neto. Modern information retrieval. *ACM Press / Addison-Wesley*, 1999.

[6] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*, 2004.

[7] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Conference on Knowledge Discovery and Data Mining (KDD)*, Washington, DC, 2003.

[8] S. Chaudhuri, B.-C. Chen, V. Ganti, and R. Kaushik. Example-driven design of efficient record matching queries. In *Conference on Very Large Databases (VLDB)*, 2007.

[9] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1), 2007.

[10] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Conference on Management of Data (SIGMOD)*, San Jose, CA, 1995.

[11] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1), 1998.

[12] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Trans. Database Syst.*, 31(2), 2006.

[13] M. Weis. Duplicate detection in XML. *WiKu Verlag*, 2008.

[14] M. Weis and F. Naumann. DogmatiX tracks down duplicates in XML. In *Conference on the Management of Data (SIGMOD)*, Baltimore, MD, 2005.

[15] S. Yan, D. Lee, M.-Y. Kan, and C. L. Giles. Adaptive sorted neighborhood methods for efficient record linkage. In *Joint Conference on Digital Libraries (JCDL)*, 2007.