Statistical Modeling vs. Machine Learning for Deduplication of Customer Records (industrial paper)

Witold Andrzejewski[1], Bartosz Bębel[1], Paweł Boiński[1], Justyna Kowalewska[3], Agnieszka Marszałek[3] and Robert Wrembel[1,2]

[1]*Poznan University of Technology, Poznań, Poland*
[2]*Interdisciplinary Centre for Artificial Intelligence and Cybersecurity, Poznań, Poland*
[3]*PKO BP, Warsaw, Poland*

Abstract
Large companies typically face a problem of multiple database records describing the same physical object (a.k.a. duplicates). There are multiple sources of duplicates, e.g., using multiple but not synchronized data repositories, applications that do not check for duplicates before inserting data into a repository, and data errors. This problem is of particular importance while dealing with personal data, e.g., in healthcare, banking, insurance. To handle the problem of duplicates, the state-of-the-art data deduplication pipeline was developed. The pipeline is equipped with multiple complex algorithms. A promising direction in data deduplication is machine learning. In this paper, we report our experience in researching and developing two approaches to deduplication of customer records in a financial institution. These approaches are based on: (1) statistical modeling and (2) machine learning. In particular, this paper summarizes our findings from comparing these two approaches. The reported research was done within a R&D project for the biggest Polish bank - PKO BP.

Keywords
data quality, data deduplication, statistical modeling, machine learning, classification

1. Introduction

Large organizations/companies face the problem of duplicated data. This problem is frequent for companies that store customer data. Duplicated and outdated data cause economic losess, increase customer dissatisfaction, and deteriorate the reputation of a company. For these reasons, data integration, cleaning, and deduplication of customer records are one of the core processes in data governance.

Data deduplication has been extensively researched in multiple research centers worldwide. The research resulted in a **base-line data deduplication pipeline**, e.g., [1, 2, 3]. It has become a standard pipeline for multiple data deduplication projects in various application domains. The pipeline includes four basic tasks, namely:

- blocking (a.k.a. indexing), which arranges records into groups, such that each group is likely to include duplicates,
- block processing (a.k.a. filtering), which eliminates records that do not have to be compared,
- entity matching (a.k.a. similarity computation), which computes similarity values between record pairs, and
- entity clustering, which creates larger clusters of similar records.

Each of these tasks is supported by multiple algorithms. Some of them apply very complex **statistical models** (SM), whereas others are based on standard **machine learning** (ML), including deep learning (DL). The ML techniques typically apply various classification models to divide records into classes of matches, probably matches, and non-matches, e.g., [4, 5, 6]. The DL techniques apply language models for both record blocking and record matching, e.g., [7, 8, 9, 10, 11].

Since there are two families of approaches to data deduplication, a question is which family offers more accurate deduplication models. In this paper, we outline our experience and findings on designing a deduplication pipeline for customer data. We designed two versions of the pipeline. The first one uses statistical modeling for discovering duplicates. The second one uses ML techniques. Both pipelines were developed within a R&D project for the biggest Polish Bank PKO BP (https://pkobp.pl). The pipelines were tested on a real data set of over 5 million of customer records. To the best of our knowledge, this is the biggest deduplication experiment comparing SM and ML approaches, reported in the research literature.

2. Tasks in the SM pipeline

We built the SM pipeline as the extension of the base-line data deduplication pipeline (BLDDP), to serve the par-

Published in the Proceedings of the Workshops of the EDBT/ICDT 2024 Joint Conference (March 25-28, 2024), Paestum, Italy
robert.wrembel@cs.put.poznan.pl (R. Wrembel)
0000-0001-9486-929X (W. Andrzejewski); 0000-0002-6426-3809 (B. Bębel); 0000-0003-4914-9394 (P. Boiński); 0000-0001-6037-5718 (R. Wrembel)

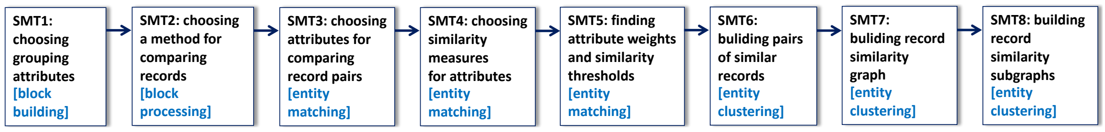
SMT1: choosing grouping attributes [block building]
SMT2: choosing a method for comparing records [block processing]
SMT3: choosing attributes for comparing record pairs [entity matching]
SMT4: choosing similarity measures for attributes [entity matching]
SMT5: finding attribute weights and similarity thresholds [entity matching]
SMT6: buliding pairs of similar records [entity clustering]
SMT7: buliding record similarity graph [entity clustering]
SMT8: building record similarity subgraphs [entity clustering]

Figure 1: The tasks in the deduplication pipeline based on statistical modeling

ticular goals of the project. First, our pipeline explicitly includes all steps that we found to be crucial for the deduplication process (whereas in the BLDDP, some steps are implicit). Second, the last two tasks in our pipeline extend the BLDDP and they allow to further merge groups of similar records into subgraphs. Third, our pipeline accepts partially dirty data, as in practice it is impossible to perfectly clean all data. The SM pipeline applied in the project is shown in Figure 1.

SMT1: In this task, grouping attributes are selected, which allow to co-locate similar records in the same group. In our project, initially 20 candidate grouping attributes were selected by domain experts. The attributes allowed either to identify customers, or they were error-free and not null, or their values haven't changed over time. The candidate attributes were further verified by means of a method that we developed. For each attribute, the method computed a value representing its fit as a grouping attribute [12]. Finally, the obtained ranking of attributes was verified by domain experts again. Based on their input, the final set of grouping attributes was selected.

SMT2: Using grouping attributes obtained from SMT1, records are arranged into groups via sorting. The method we used is very similar to the one described in [13], however multiple sortings were used.

SMT3: The goal of this task is to select attributes that will contribute to assessing a similarity value between records in each pair. Attributes that: (1) represent record identifiers, (2) do not include nulls, (3) include cleaned values, (4) include unified (homogenized) values, (5) do not change values over time are good candidates. Unfortunately, in real scenarios, attributes exposing all these characteristics are often unavailable. In our project, the set of attributes selected for comparing record pairs is based on the aforementioned preferable attribute characteristics and on expert knowledge. The set includes 18 attributes describing individual customers (including personal data and address).

SMT4: Similarities between corresponding attributes (selected in SMT3) in pairs of records are compared by means of similarity measures. The literature lists well over 30 different similarity measures for text data (e.g., [14, 15]). Unfortunately, there are no rules that would guide a data scientist for selecting the right similarity measure for an attribute having a given characteristic of its values. For this reason, in our project, the selection of the most suitable similarity measures for our problem was based on excessive experimental evaluation [16, 17].

SMT5: Measures selected in SMT4 are applied to computing similarities of corresponding attribute values, constituting customer pairs of records. An adequate similarity measure is applied to every pair of attributes. Based on attribute similarities, an overall *record similarity* value is computed. In a simple scenario, all compared attributes are equally important. In practice, some attributes may contribute more to records similarity than others. For example, a last name is more important (usually more clean and not null) than an email address (frequently null and changing in time). For this reason, the compared attributes must be weighted, resulting in an overall weighted similarity of records. The first challenge in task SMT5 is to *define adequate weights* for attributes. In practice, these weights are defined by a try and error procedure and by applying expert knowledge. Based on the similarity of a pair of records, the pair is classified either as *duplicates* (class T), or *probably duplicates* (class P), or *non-duplicates* (class N). For this kind of classification, the so-called *similarity thresholds* have to be defined for each class. These thresholds impact the number of true positives and false negatives. Setting adequate values of the thresholds is another challenge in SMT5. In practice, the thresholds are defined experimentally with the support of domain experts [14]. In our project, to find attribute weights and similarity thresholds we apply mathematical programming [17].

SMT6: To compare records in a group, a popular technique, called *sorted neighborhood* is used. It accepts one parameter that is the size of a sliding window in which records are compared. This parameter impacts the computational performance and the number of discovered (potential) duplicates. Defining the *size of the sliding window* is challenging. A window that is too narrow may prevent from discovering all potential duplicates, whereas a window that is too wide will allow to discover more potential duplicates at a cost of unnecessary time overhead caused by comparing more records - some of them will be false positives. In our project, the window size was selected by means of excessive experiments [13].

SMT7: Since similar records may form groups larger than pairs, in order to find such groups, all similar pairs of records have to be combined. To this end, we cre-

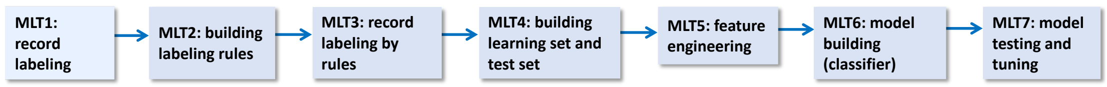
MLT1: record labeling
MLT2: building labeling rules
MLT3: record labeling by rules
MLT4: building learning set and test set
MLT5: feature engineering
MLT6: model building (classifier)
MLT7: model testing and tuning

Figure 2: The tasks in the deduplication pipeline based on machine learning

ated a *record similarity graph*. In this graph, nodes represent records and edges represent similarity links between records (labeled with similarity values).

SMT8: Similar records in the record similarity graph can be extracted by cutting the graph into sub-graphs. In our project, we applied clustering of records by using similarity values of pairs of records computed in SMT5. For this purpose, we modified the Highly Connected Subgraphs (HCS) clustering algorithm [18]. HCS recursively applies the minimum cut to the record similarity graph (separately for each connected component) until the obtained sub-graphs are highly connected, i.e. the size of the minimum cut is larger than the number of nodes divided by two. The resulting sub-graphs form the groups of similar records. The modifications include: using a minimum weighted cut instead of a minimum cut, searching for additional edges in sufficiently small subgraphs and assigning resulting singletons to the most similar groups after the clustering ends. The performance of the modified HCS is close to linear w.r.t. the number of pairs, since larger connected components in the similarity graph are rare.

Notice that the pipeline we proposed, is mapped to the BLDDP as follows: SMT1 realizes block building in the BLDDP; SMT2 realizes block processing; SMT3, SMT4, and SMT5 realize entity matching; SMT6, SMT7, and SMT8 realize entity clustering. A more detailed description of our pipeline is available in [17].

3. Tasks in the ML pipeline

The machine learning pipeline applied in the project is shown in Figure 2.

MLT1: in this task, 1000 pairs of records were manually assigned labels by domain experts. A label indicated one of the three classes, namely *T*, *P*, or *N*. All the labels were roughly equally represented. Since the Bank database included over 20 millions of customer records, it was not possible to label manually a sufficiently large sample of record pairs. For this reason, in **MLT2** the set of rules was built manually with the support of domain experts. It is important to stress that high quality of these rules is crucial for subsequent tasks. The rules were next applied in task **MLT3** to automatically label a set of 50 000 pairs of customer records.

It was assumed that the effect of applying the rules was to determine decisions consistent with those of the domain experts. Even for a small set of attributes being compared in pairs (18 attributes in our case), attempts to create simple rules of type *if X then Y else Z* turned out to be challenging. It was due to the very different scenarios present in the data, in particular, the lack of 100% similarity between the compared attribute values. It turned out that rules that only confirmed whether a given pair could be a representative of a given class T, P, or N were much easier to interpret. This led to a set of m rules under the following assumptions:

- each rule can return only one answer: T, P or N;
- a rule for a given pair of customer records may return no answer;
- for a given pair, successive rules are applied; an answer can be returned by a set of 0 to 89 rules;
- rules have a certain priority - the strongest are rules that assign class N, followed by rules that assign class T, and the weakest rules are those assigning P.

The above assumptions cause that the final decision made by the rules is determined as described in Tab. 1. It shows the number of N, P, and T responses generated by a set of rules. Symbol $*$ denotes any number of rules returning a given answer. In general, the following four cases are possible. *Case 1*: when at least one rule returns N, then the final decision is N. This is due to the highest priority of rules returning N. *Case 2*: decision P is taken when no rule returns N and no rule returns T and at least one rule returns P. *Case 3*: decision T is taken when no rule returns N and at least one rule returns T. *Case 4*: no rule returns an answer. In such a case, given the bank's conservative policy, the final decision is N, since a possible false negative is much less costly than a false positive.

Table 1
Determination of the final rule decision

No.	N	P	T	Decision
1	>0	*	*	N
2	0	>0	0	P
3	0	*	>0	T
4	0	0	0	? (N by default)

Despite tremendous effort, designing rules that matched exactly the experts decisions proved to be impossible in a reasonable amount of time. The confusion matrix for the set of 89 rules created is shown in Tab. 2. In addition to the T, P, and N classes, it is easy to see that for a total of 70 pairs, the rules made no decision (column *None*). This means that for 1 000 pairs, the *coverage* of pairs by rules is equal to 93%. It is noteworthy that the assessments made by the rules were more conservative than those made by the experts, i.e., in the case of confusion, they assigned class N to pairs considered to be T and class P to pairs considered to be T. It is notable that never a pair of class N was considered either P or T.

Table 2
Confusion matrix for the final set of rules

		Rules decision			
		N	**P**	**T**	**None**
Expert decision	**N**	310	0	0	38
	P	8	373	0	30
	T	0	7	232	2

In **MLT3**, from the test set of over 5 million of customer records, 2.5 million of pairs were created and then they were labeled by means of the rules. The set of pairs to be labeled was not random, and a good portion of it contained common as well as problematic examples. Therefore, it can be considered that it was somehow biased and unrepresentative for the coverage assessment measure. It turned out to be much more interesting to assess coverage by experimenting with the set of pairs obtained by applying step SMT2 (from the SM approach). For the set of 2.5 million of pairs analyzed, in only 23 352 cases the rules were not able to make a final decision. This means the rule coverage of 99.987%. It should be noted here that of the 89 rules, the most popular rules (i.e., those that produced a result for the largest number of pairs) assigned class N. The most popular rule produced N for more than 83% of pairs. When considering the entire population of pairs (i.e., any pairs from the set), the result would be even higher, as the chance to find duplicates decreases significantly.

In, **MLT4**, from the set of pairs built in MLT3, a training and a testing data sets were created by stratified sampling. Having created these two subsets, we have added new features inspired by the conditions of the expert rules (task **MLT5**).

In **MLT6**, we run several preparatory experiments to get the "feel" of the data. In one of them, we used PCA method to obtain two principal components and rendered each example in the dataset of 1000 pairs manually labeled by domain experts on a two-dimensional scatter plot, with classes T, P, and N marked in colors. The result was that classes T and N were separated quite well, while P was scattered all over the plot. This observation meant that class P would be problematic.

Having run the preparatory experiments, four different classification models were created based on the training set. The models included: decision tree, random forest, SVM, and feed forward neural network. These models were built using Python package *sklearn*. Hyperparameters of the models were adjusted manually or (if needed) with the help of the *optuna* package, based on the dataset labeled by domain experts. We also attempted to use an auto-ML approach by means of the *tpot* library, but it was not able to produce any model that would be more accurate than the models mentioned above, within the given time frame.

Finally, in **MLT7**, the quality of all the produced models was tested based on the testing set obtained from the automatically labeled data as well as based on the set of 1000 pairs manually labeled by domain experts. The model quality was measured by means of precision, recall, measure F1, and accuracy.

4. Development

Pipeline implementation environment: Both the SM and ML pipelines were implemented in a typical data science environment, which included: (1) a relational database management system to store customer data, (2) csv files to store temporary data produced by the tasks in the pipelines, (3) spreadsheet files to store groups of duplicated records, and (4) Python programs and standard packages to implement tasks in both pipelines.

Data size: In the reported project, in the development and testing phases we used 2.5 million of pairs of customer records. In the production system, the pipelines run on the Bank customer database, which includes over 20 million of records.

5. Results

Tab. 3 presents precision, recall, measure F1, and accuracy achieved by the ML models on the dataset of 1000 pairs manually labeled by the domain experts. We also add results achieved by the Statistical Modeling approach, for comparison. The numbers in brackets represent the ranking of the values (best-1, worst-5). As it can be observed, the best performance on this dataset was obtained by the Statistical Model. The worst result was obtained by the decision tree. In the rest of the cases the ranking is not so obvious, as not all measures point to the same rank. However, since F1 is an aggregation of precision and recall, and the ranking of F1 is consistent with the accuracy, we can assume that the second best is random forest, followed by SVM and feed-forward neural network (FFNN).

The best results obtained via the Statistical Model stem most probably from the fact that all parameters of the

Table 3
Comparison of models on the expert labeled dataset

	Decision Tree	Random forest	FFNN	SVM	Statistical model
Precision	[5] 0.7620	[3] 0.8369	[4] 0.8176	[2] 0.8422	**[1] 0.8607**
Recall	[5] 0.7801	[2] 0.8498	[3] 0.8391	[4] 0.8208	**[1] 0.8799**
F1	[5] 0.7631	[2] 0.8407	[4] 0.8241	[3] 0.8296	**[1] 0.8673**
Accuracy	[5] 0.7612	[2] 0.8383	[4] 0.8201	[3] 0.8276	**[1] 0.8640**

Table 4
Comparison of models on the testing dataset

	Decision Tree	Random Forest	FFNN	SVM	Statistical model
Precision	[2] 0.9577	**[1] 0.9629**	[3] 0.9444	[4] 0.8898	[5] 0.8051
Recall	[3] 0.9399	[2] 0.9551	[4] 0.9368	**[1] 0.9619**	[5] 0.8182
F1	[2] 0.9484	**[1] 0.9590**	[3] 0.9405	[4] 0.9201	[5] 0.8065
Accuracy	[2] 0.9958	**[1] 0.9967**	[3] 0.9953	[4] 0.9926	[5] 0.9859

model (mainly attribute weights and similarity thresholds) were optimized directly based on the expert labeled dataset also used for testing. The same dataset was only used for optimizing hyperparameters for other models (random forest, SVM, FFNN). Decision trees were adjusted manually.

Tab. 4 presents precision, recall, F1, and accuracy achieved by the ML models on the testing dataset, obtained by means of 89 rules. It is crucial to underline that the results presented in the table compare **how close are the results of the ML model to the results obtained by applying the rules, and not to the ground truth (which is not available)**. The structure of the table is the same one as Tab. 3.

There are a few interesting observations to be made here. First, notice that the orders of precision, F1, and accuracy are consistent, while recall is different. However, similarly as before we can ignore precision and recall in favor of their aggregation, i.e., F1, to obtain the ranking of the models. In this approach, the best model is random forest, followed by decision tree, FFNN, SVM, and the Statistical Model. Thus, the obtained order is completely different. We explain the results as follows.

Since training and testing datasets had labels generated by the same set of rules, the classifiers that were trained on the training dataset tried to generalize the set of these rules. Thus, their performance on the testing dataset is quite good. The Statistical Model was not built based on the set of rules at all, and thus, its performance on the testing dataset is worse. It is also interesting that tree-based classifiers, i.e., decision tree and random forest perform better than mathematical models such as FFNN and SVM, which come down to division of attribute space by hyperplanes. We suspect that this stems from the fact, that trees are just representations of sets of rules. Thus, the tree model structure fits better the underlying source of the labels than the mathematical models. One can also observe that a single test in a node of a tree is also a hyperplane dividing the attribute space, but such planes are orthogonal to one of the axis, while in mathematical models, the hyperplanes are aligned arbitrarily. Larger degree of freedom makes it harder for the mathematical models to find the original model that produced the labels, which leads to worse results.

Regardless of the testing dataset, the best results were achieved by the random forest model. Its confusion matrix for the dataset labeled by the domain experts is given in Tab. 5. One of the most important observations here is the fact that the number of erroneous classifications involving class P is an order of magnitude greater than the number of erroneous classifications involving classes T and N. This observation confirms the result mentioned in Section 3.

Table 5
Confusion matrix for random forest

		Random forest decision		
		N	**P**	**T**
Expert decision	**N**	318	24	6
	P	24	319	68
	T	1	28	212

6. Summary

In this paper, we reported our experience from a R&D project on deduplicating customers data. The project was realized for the biggest Polish Bank PKO BP. While presented solutions were designed explicitly for the Bank dataset, the findings are general and the approach can be fitted to other, similar problems.

In the project, we designed two deduplication pipelines: (1) based on statistical modeling and (2) based on machine learning. The SM pipeline extends the standard deduplication pipeline from the literature to the particular characteristics of data being deduplicated and to the project requirements. The ML pipeline was designed from scratch within the project. Both pipelines were implemented, tested on a data set of 2.5 million of pairs of customer records, and verified by domain experts. The obtained results were accepted by the Bank. The project ended with the decision to run the SM pipeline in the production system on the Bank database, which includes over 20 million of customer records. This pipeline has already been deployed in the Bank.

It must be stressed that the real settings of the project differ from the ones assumed in the research literature (the ideal ones) among others w.r.t.: (1) the quality of data being deduplicated, (2) the sizes of deduplicated data, (3) the availability of tagged data for ML algorithms, (4) available development environments, (5) the performance of a deduplication pipeline (time and quality). The real settings, which are far from the ideal ones, made the project very challenging.

Acknowledgements. The project is supported by the grant from the National Center for Research and Development no. POIR.01.01.01-00-0287/19.

References

[1] A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios, Duplicate record detection: A survey, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 1–16.

[2] H. Köpcke, E. Rahm, Frameworks for entity matching: A comparison, Data & Knowledge Engineering 69 (2010) 197–210.

[3] G. Papadakis, L. Tsekouras, E. Thanos, G. Giannakopoulos, T. Palpanas, M. Koubarakis, Domain- and structure-agnostic end-to-end entity resolution with jedai, SIGMOD Record 48 (2019) 30–36.

[4] X. Chen, Y. Xu, D. Broneske, G. C. Durand, R. Zoun, G. Saake, Heterogeneous committee-based active learning for entity resolution (healer), in: European Conf. on Advances in Databases and Information Systems ADBIS, volume 11695 of *LNCS*, Springer, 2019, pp. 69–85.

[5] S. Sarawagi, A. Bhamidipaty, Interactive deduplication using active learning, in: ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, ACM, 2002, pp. 269–278.

[6] J. A. Silva, D. A. Pereira, A multiclass classification approach for incremental entity resolution on short textual data, Int. Journal of Business Intelligence and Data Mining 18 (2021) 218–245.

[7] U. Brunner, K. Stockinger, Entity matching with transformer architectures - A step forward in data integration, in: Int. Conf. on Extending Database Technology (EDBT), OpenProceedings.org, 2020, pp. 463–473.

[8] A. Jain, S. Sarawagi, P. Sen, Deep indexed active learning for matching heterogeneous entity representations, VLDB Endowment 15 (2021) 31–45.

[9] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Deep learning for entity matching: A design space exploration, in: SIGMOD Int. Conf. on Management of Data, ACM, 2018, pp. 19–34.

[10] S. Thirumuruganathan, H. Li, N. Tang, M. Ouzzani, Y. Govind, D. Paulsen, G. Fung, A. Doan, Deep learning for blocking in entity matching: A design space exploration, Proc. VLDB Endowment 14 (2021) 2459–2472.

[11] A. Zeakis, G. Papadakis, D. Skoutas, M. Koubarakis, Pre-trained embeddings for entity resolution: An experimental analysis, Proc. VLDB Endowment 16 (2023) 2225–2238.

[12] P. Boiński, M. Sienkiewicz, B. Bębel, R. Wrembel, D. Gałęzowski, W. Graniszewski, On customer data deduplication: Lessons learned from a r&d project in the financial sector, in: Workshops of the EDBT/ICDT 2022 Joint Conference, volume 3135 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.

[13] P. Boinski, W. Andrzejewski, B. Bebel, R. Wrembel, On tuning the sorted neighborhood method for record comparisons in a data deduplication pipeline - industrial experience report, in: Int. Conf. Database and Expert Systems Applications (DEXA), volume 14146 of *LNCS*, Springer, 2023, pp. 164–178.

[14] P. Christen, Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection, Data-Centric Systems and Applications, Springer, 2012.

[15] F. Naumann, Similarity measures, Hasso Plattner Institut, 2013.

[16] W. Andrzejewski, B. Bebel, P. Boinski, M. Sienkiewicz, R. Wrembel, Text similarity measures in a data deduplication pipeline for customers records, in: Int. Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP), volume 3369 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 33–42.

[17] W. Andrzejewski, B. Bębel, P. Boiński, R. Wrembel, On tuning parameters guiding similarity computations in a data deduplication pipeline for customers records: Experience from a r&d project, Information Systems 121 (2024) 102323.

[18] F. Hüffner, C. Komusiewicz, A. Liebtrau, R. Niedermeier, Partitioning biological networks into highly connected clusters with maximum edge coverage, IEEE/ACM Transactions on Computational Biology and Bioinformatics 11 (2014) 455–467.