On Customer Data Deduplication: Lessons Learned from a R&D Project in the Financial Sector

Paweł Boiński[1], Mariusz Sienkiewicz[1], Bartosz Bębel[1], Robert Wrembel[1], Dariusz Gałęzowski[2] and Waldemar Graniszewski[3]

[1]*Poznan University of Technology, Poznań, Poland*
[2]*DAMA Poland, Warsaw, Poland*
[3]*Warsaw University of Technology, Warsaw, Poland*

Abstract

Despite the fact that financial institutions (FIs) apply data governance strategies and use the most advanced state-of-the-art data management and data engineering software and systems to support their day-to-day businesses, their databases are not free from some faulty data (dirty and duplicated). In this paper, we report some conclusions from an ongoing research and development project for a FI. The goal of this project is to integrate customers' data from multiple data sources - clean, homogenize, and deduplicate them. This paper, in particular, focuses on findings from developing customers' data deduplication process.

Keywords

data quality, data cleaning, data deduplication pipeline

1. Introduction

Financial institutions (FIs) apply data governance strategies and use the most advanced state-of-the-art data management and data engineering software to manage data collected by their day-to-day businesses. Unfortunately, the application of advanced technologies does not prevent from collecting and storing some faulty data - mainly erroneous, outdated, and duplicated, e.g., [1]. Such data mainly concern customers, both individuals and institutions.

Duplicated and outdated data cause economic loses, increase customer dissatisfaction, and deteriorate a reputation of a FI. For these reasons, data integration, cleaning, and deduplication of customers' is one of the processes in data governance.

In the research literature, a base-line data deduplication pipeline has been proposed, e.g., [2, 3, 4]. It has become a standard pipeline for multiple data deduplication projects. The pipeline includes four basic tasks, namely: (1) blocking (a.k.a. indexing), which arranges records into groups, such that each group is likely to include duplicates, (2) block processing (a.k.a. filtering), which eliminates records that do not have to be compared, (3) entity matching (a.k.a. similarity computation), which computes similarity values between record pairs, and (4) entity clustering, which creates larger clusters of similar records.

In this paper, we outline our experience and findings from designing a deduplication pipeline for customers' data (Section 2). We discuss approaches that are possible for each task in the pipeline and present particular solutions that were proven to be adequate to solve the addressed problem. Final conclusions are presented in Section 3. Notice that this paper presents findings from a real R&D project, and therefore, not all details can be revealed, as they are treated as the company know-how.

2. Deduplication pipeline in the project

Inspired by the aforementioned base-line data deduplication pipeline (BLDDP), in the described project, we apply an adjusted pipeline that suits the goals of our project. There are three basic differences between our pipeline and the BLDDP. First, in our pipeline, we explicitly included all steps that we found to be crucial for the deduplication process on customers' data, whereas in the BLDDP some steps are implicit. Second, the last task in our pipeline allows to further merge some groups of similar records (cliques), whereas the BLDDP, to the best of our knowledge, does not include this task. Third, our pipeline accepts dirty customers data, whereas the BLDDP assumes that input data were cleaned beforehand.

Our pipeline includes the following tasks (cf. Figure 1), which are outlined in the remainder of the paper: [T1] selecting grouping attributes, [T2] selecting attributes used to compare record pairs, [T3] choosing a method for comparing records, [T4] selecting similarity measures for comparing values of attribute pairs, [T5] defining weights

Published in the Workshop Proceedings of the EDBT/ICDT 2022 Joint Conference (March 29-April 1, 2022), Edinburgh, UK
robert.wrembel@cs.put.poznan.pl (R. Wrembel)
0000-0001-6037-5718 (R. Wrembel)

of attributes to compute records similarities and choosing similarity thresholds, [T6] building pairs of records having high similarity value, [T7] building cliques of similar records, [T8] further merging cliques of similar records. [T1] realizes blocking in BLDDP; [T3] realizes block processing and entity matching; [T2], [T4], and [T5] realize entity matching; [T6] and [T7] realize entity clustering.

2.1. Pipeline implementation environments

Tasks [T1] to [T6] were implemented in parallel in two alternative environments. The first one is a typical *data engineering environment*, based on the Oracle DBMS as a data storage and the PL/SQL programming language for implementing the deduplication pipeline; this environment is a standard one in the FI running the project. The second environment is a typical *data science environment*, based on csv files as a data storage and Python (Anaconda or Jupyter-lab, data science packages) for implementing the pipeline.

2.2. Tasks in the pipeline

2.2.1. T1: Grouping attributes

The main challenge in grouping records is to select such a set of grouping attributes that would allow to identify the highest number of potentially duplicate records. Even though, in the research literature there were proposed 14 different blocking methods [5], there is no single universal grouping method suitable for all application domains [6].

Inspired by [7], we proposed a method based on statistical characteristics of customers attributes: (1) the number of $nulls$ to the number of $notnull$ values of an attribute, (2) the number of $duplicated$ values to the number of $notnull$ values of an attribute, (3) the number of $notduplicated$ to the number of $notnull$ values of an attribute, (4) the number of $(notnull - distinct)/notnull$ values of an attribute. These characteristics are computed for every attribute being a candidate for grouping. Additionally, (1) a diversity of values of each attribute is modeled by means of the Gini Index and (2) the size of a record group is penalized by means of a quadratic function with a negative value of coefficient *a*.

Notice that the initial set of potential grouping attributes was selected based on expert knowledge. It included 20 attributes. Next, the candidate attributes were processed by means of our method and their statistical characteristics were computed. The obtained ranking of attributes was verified by domain experts. Based on their input, the final set of attributes was selected for arranging (grouping) records.

2.2.2. T2: Attributes for comparing record pairs

Having ordered records by the attributes from the ranking obtained from task [T1], the next task is to select attributes whose values will be compared in record pairs, to compute the similarity of records in each pair. Potential candidates for comparison are attributes that: (1) are record identifiers, (2) do not include nulls, (3) include cleaned values, e.g., no typos, no additional erroneous characters, (4) include unified (homogenized) values, e.g., no abbreviations, the same acronyms used throughout the whole data set.

Unfortunately, in real cases, such attributes often do not exist. As it concerns record identifiers, in FI applications, natural identifiers are typically used, but their values are frequently artificially generated (in cases when natural identifiers cannot be used, i.e., a customer is not able to provide it). Thus, in some cases, artificially generated IDs may have the same values as the natural ones. Notice that the financial sector is strictly regulated by means of European law, national law, and recommendations issued by institutions controlling the sector. As a consequence, procedures aiming at improving the quality of data in this sector are strictly controlled. For this reason, possibilities of applying data cleaning processes are limited.

In the described 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 (e.g., personal data and address components) and 24 attributes describing institutional customers (e.g., institution names, addresses, type of business run).

2.2.3. T3: A method for comparing records

Based on the ranking of grouping attributes obtained from task [T1] (cf. Section 2.2.1), records need to be arranged into groups. Next, in each group records are compared in pairs. The literature proposes two popular techniques for grouping, namely: hashing, e.g., [8, 9] or sorting - know as the *sorted neighborhood* method, e.g., [10, 11].

The *sorted neighborhood* method accepts one parameter that is the size of a sliding window in which records are compared. The larger the window size is the more potential duplicates can be found, but the longer record comparison time is, since more records have to be compared each time. Some experimental evaluations of the window size from the literature discuss a typical size that ranges from 2 to 60 records [10].

The *sorted neighborhood* method is intuitive, has an acceptable computational complexity, and is available in one of the Python libraries. For this reason, it was applied in the project. We run a series of experiments in order to determine the best window size. Our experiments showed that the size has to be adjusted experimentally for a particular data set being deduplicated. For comparing individual customers' records we used the window of 20 records, whereas for comparing institutional customers we used a variable window size with the maximum of 200 records.

2.2.4. T4: Similarity measures for text attributes

The literature on data deduplication and similarity measures lists well over 30 different similarity measures for text data, e.g., [12, 13]. One may find in the literature suggestions, supported by experimental evaluations, on the applicability of different measures to different text data, e.g., [14, 15, 16, 12].

In our project, we evaluated 44 measures available in Python packages. Some measures, e.g., Levenshtein, Jaro, Jaro-Winkler exist in a few different implementations (packages), thus we evaluated these implementations as well. The evaluation was run on three different real data sets, i.e., (1) customers' last names of average length of 10.9 characters, (2) street names of avg length of 16 chars, and (3) institution names of avg length of 45.5 chars. Customers' names included 98% of 1-word names and 2% of 2-word names, which reflected a real distribution of such types of names in our customers' population. All test data represented true positives, but with typical real errors found by data profiling.

From the evaluation we draw the following conclusions:

- for short strings, like last names and street names (composed of 7 to 28 characters), the Overlap, Jaro-Winkler, and StrCmp95 similarity measures gave the highest similarity values;
- for long strings, like institution names (composed of 46 to 116 characters and up to 12 separate words) the Overlap, Sorensen, and StrCmp95 measures gave the highest similarity values, therefore they were recommended for comparing such kinds of data.

2.2.5. T5: Attribute weights and similarity thresholds

In task [T5], we applied an iterative process of tuning weights of attributes used to compute records' similarity, with the support of domain experts. Additionally, rules had to be defined to decide whether to compare values of a given attribute. Let us assume that a pair of records r_m and r_n is compared to compute their similarity value. Some cases handled by the rules include:

- when r_m has a defined value of attribute A and the value of A of r_n is null, then the values of A cannot be compared; in this case, A is not considered for computing record similarity for pair (r_m, r_n);
- when records IDs are compared but the value of the ID for r_m is artificially generated and the value of the ID for r_n is real, then such values cannot be compared; in this case, such an ID is not considered for computing record similarity for pair (r_m, r_n);
- if IDs can be compared (i.e., they include real values), then a binary similarity value is assigned of either 1 (IDs are equal) or 0 (IDs are not equal) for a compared pair of IDs.

On top of the rules, for institutional customers we used equal weights for each attribute being the subject of comparison. Whereas for individual customers, higher weights were set for ID and last name attributes. These weights were set based on an iterative experimentation process and evaluation of the results by domain experts.

Based on the weighted values of similarities between pairs of individual attributes of records r_m and r_n, a total similarity of (r_m, r_n) was computed. Let us denote it as r_{sim}. Based on its value, a given pair of records was classified either as *similar* (matches) or *non-similar* (non-matches), or *undecided.* For this kind of classification, the so-called *similarity thresholds* had to be defined. Again, in practice, these thresholds are defined based on the analysis of the obtained record pairs and based on knowledge of domain experts [12, 17].

In our project we applied the same approach. Based on the knowledge of the FI experts, the lowest value of r_{sim} (i.e., for similar records) was set to 0.8 and the highest value for *non-similar* was set to 0.6.

2.2.6. T6: Building pairs of similar records

The *sorted neighborhood* method produces pairs of similar records with: (1) their overall value of r_{sim} and (2) similarity values for each attribute being compared. These data are stored in a repository, cf. Section 2.1 and visualized in a spreadsheet for expert verification.

2.2.7. T7: Building cliques of similar records

Since similar records' pairs may form larger sets, to find such sets, all similar pairs have to be combined in a graph, with records representing nodes and labeled edges representing similarities between records. In such a graph, a group of similar records forms a maximal clique. Thus, the problem of finding sets of similar records transforms to finding maximal cliques in a graph. In general, it is a NP-hard problem [18]. This problem becomes computationally less expensive for sparse graphs, e.g., [19, 20], which is the case of a graph created from similar records.

For finding maximal cliques a few fast algorithms were developed.

One of them is the *Bron-Kerbosh* algorithm [21], which we decided to use for the following reasons. First, it is frequently used in the community working on graph processing. Second, it is implemented in multiple programming languages, including Python. Third, its worst case computational complexity is $O(3^{N/3})$, where N denotes the number of graph nodes. For sparse graphs the complexity is lower.

The algorithm was used for finding cliques in a graph composed of 2228580 customers' nodes. This evaluation confirmed its efficiency in terms of processing time and its applicability to the deduplication problem (confirmed by the experts from the FI).

2.2.8. T8: Merging cliques of similar records

If the number of similar records is larger than the size of a sliding window in *sorted neighborhood*, then a few cliques are created and all of them contain records that are similar to each other. Therefore, the final step is to merge cliques that include a certain number of common records (currently the Jaccard coefficient is used to decide which cliques to merge). We are also experimenting with a variable, automatically adjustable window size.

3. Final observations

In this paper, we reported our experience from a R&D project for a FI on deduplicating customers' data. The project is ongoing (two out of four stages have been already realized). In the project, we adapted the standard deduplication pipeline from the literature to the particular characteristics of data being deduplicated and to the project requirements. The whole pipeline was implemented and verified by domain experts. The results obtained so far were accepted by the FI.

It must be stressed that the reality of the discussed project differs from the one assumed in the research literature, i.e., (1) the assumption on the cleanness of data being deduplicated, (2) the sizes of deduplicated data sets, (3) the availability of tagged data for ML algorithms, and (4) neglecting data aging process. Neither of these assumptions is true in our project, as outlined in the following sections.

3.1. Data cleanness

The base-line data deduplication pipeline [22, 2, 3, 23, 4] assumes that data delivered to the pipeline are clean (e.g., no null values, no spelling errors, homogenized full names and abbreviations). Unfortunately, this assumption in real projects cannot be guaranteed, especially in the financial sector. There exist some typos, missing values, inconsistent values in attributes storing personal data, institution names, and addresses. Moreover, not all natural IDs are reliable. By regulations, even known dirty customers data cannot be cleaned without an explicitly permission of a customer. Getting such permissions from millions of customers in a finite time frame is impossible. For this reason, only simple cleaning is possible, like removing leading or trailing erroneous signs from customers addresses. For this reason, in practice the deduplication pipeline has to be applied to data that has undergone only basic cleaning.

3.2. Data size

Most of the methods used in the base-line data deduplication pipeline were verified on either small real data sets, e.g., bibliographical with 32000 records [6, 24, 25, 17, 26], restaurants - 500 records [24, 25], movies - 5000 records [26], or patients - 128000 records [27], or on data sets generated artificially [6, 7].

Whereas, in this paper we reported our experience on deduplicating customers' data of much larger volumes, i.e.,: (1) 2228580 records describing individual customers and (2) 1185290 records describing institutional customers. The final goal of the reported project is to apply the developed pipeline and techniques to a database storing more than 11 million of customers' records (since the project is ongoing, this stage will be run at the end of the project).

3.3. Tagged data for ML

Some tasks in the deduplication pipeline can be run with the support of machine learning (ML) techniques, e.g., blocking [28, 29], selecting similarity measures and thresholds [30], matching similar records [31, 32]. If the pipeline applies ML for the entity matching task, it is assumed that there exists a set of training records tagged as true positives and true negatives. Unfortunately, in a large FI it is impossible to create such a set of training data because of the volume of data to be processed by the pipeline. For an original data set composed of several million of customers, a training data set of a reasonable size should include at least several thousands of tagged training records. In reality, such a large number of training records is impossible to be created by experts. For this reason, in practice, training data are frequently unavailable for ML algorithms.

In order to overcome this difficulty, unsupervised learning techniques are used, e.g., [33, 34]. Some publications report on applying active learning techniques to a deduplication process, e.g., [35, 36, 37, 38, 39] and this direction will also be investigated in the project. Currently we are experimenting with weakly supervised learning [40] with the support of the *snorkel* library.

3.4. Data aging

An inherent feature of some types of data, is their aging. For example, customers' last names, identification documents, different types of postal addresses, and contact data (phone numbers, emails) have this feature. Outdated data impact the possibility to discover duplicate records. For this reason, for a deduplication process it would be profitable to know which pieces of compared data are likely to be outdated.

Building data aging models has not been researched so far (the only approach addressing a related problem is [41], but in the context of temporal data). Including aging models into a deduplication pipeline seems to be totally unexplored field of research either. In the last stage of our project we aim at developing data aging models based on ML algorithms.

3.5. Working with experts

While designing the deduplication pipeline and evaluating its results, we have benefited from the help of experts. Their knowledge was used to determine an initial set of attributes used for comparing records and choosing similarity thresholds. The pipeline was tuned in an iterative way, each time being based on the input from the experts evaluating the obtained results.

In particular, grouping clients into clicks turned out to be very useful - it provided a holistic view of customers' representations and allowed to clearly identify duplicates. In general, the proposed deduplication approach allowed for the proper implementation of FI business goals.

Acknowledgements. The work of Mariusz Sienkiewicz is supported by the Applied Doctorate grant no. DWD/4/24/2020 from the Polish Ministry of Education and Science.

References

[1] M. Sienkiewicz, R. Wrembel, Managing data in a big financial institution: Conclusions from a r&d project, in: Proc. of the Workshops of the EDBT/ICDT 2021 Joint Conference, volume 2841 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021.

[2] A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios, Duplicate record detection: A survey, IEEE Trans. Knowl. Data Eng.g 19 (2007) 1–16.

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

[4] 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.

[5] A. Colyer, The morning paper on An overview of end-to-end entity resolution for big data, https://blog.acolyer.org/2020/12/14/entity-resolution/, 2020.

[6] M. Bilenko, B. Kamath, R. J. Mooney, Adaptive blocking: Learning to scale up record linkage, in: IEEE Int. Conf. on Data Mining (ICDM), IEEE Computer Society, 2006, pp. 87–96.

[7] L. de Souza Silva, F. Murai, A. P. C. da Silva, M. M. Moro, Automatic identification of best attributes for indexing in data deduplication, in: A. Mendelzon Int. Workshop on Foundations of Data Management, volume 2100 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018.

[8] N. N. Dalvi, V. Rastogi, A. Dasgupta, A. D. Sarma, T. Sarlós, Optimal hashing schemes for entity matching, in: Int. World Wide Web Conf. WWW, 2013, pp. 295–306.

[9] H. Kim, D. Lee, HARRA: fast iterative hashed record linkage for large-scale data collections, in: Int. Conf. on Extending Database Technology EDBT, volume 426, ACM, 2010, pp. 525–536.

[10] M. A. Hernández, S. J. Stolfo, The merge/purge problem for large databases, in: ACM SIGMOD Int. Conf. on Management of Data, ACM Press, 1995, pp. 127–138.

[11] B. Ramadan, P. Christen, H. Liang, R. W. Gayler, Dynamic sorted neighborhood indexing for real-time entity resolution, ACM Journal of Data and Information Quality 6 (2015) 15:1–15:29.

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

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

[14] M. Alamuri, B. R. Surampudi, A. Negi, A survey of distance/similarity measures for categorical data, in: Int. Joint Conf. on Neural Networks (IJCNN), IEEE, 2014, pp. 1907–1914.

[15] S. Boriah, V. Chandola, V. Kumar, Similarity measures for categorical data: A comparative evaluation, in: SIAM Int. Conf. on Data Mining (SDM), SIAM, 2008, pp. 243–254.

[16] P. Christen, A comparison of personal name matching: Techniques and practical issues, in: Int. Conf. on Data Mining (ICDM), IEEE Computer Society, 2006, pp. 290–294.

[17] 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.

[18] Y. Ma, T. Tran, Typimatch: type-specific unsupervised learning of keys and key values for heterogeneous web data integration, in: ACM Int. Conf. on Web Search and Data Mining (WSDM), ACM, 2013, pp. 325–334.

[19] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, Operations Research Letters 9 (1990) 375–382.

[20] D. R. Wood, An algorithm for finding a maximum clique in a graph, Operations Research Letters 21 (1997) 211–217.

[21] C. Bron, J. Kerbosch, Finding all cliques of an undirected graph (algorithm 457), Communications of the ACM 16 (1973) 575–576.

[22] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, K. Stefanidis, An overview of end-to-end entity resolution for big data, ACM Computing Surveys 53 (2021) 127:1–127:42.

[23] G. Papadakis, D. Skoutas, E. Thanos, T. Palpanas, Blocking and filtering techniques for entity resolution: A survey, ACM Computing Surveys 53 (2020) 31:1–31:42.

[24] W. W. Cohen, J. Richman, Learning to match and cluster large high-dimensional data sets for data integration, in: ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, ACM, 2002, pp. 475–480.

[25] M. Kejriwal, D. P. Miranker, An unsupervised algorithm for learning blocking schemes, in: IEEE Int. Conf. on Data Mining, IEEE Computer Society, 2013, pp. 340–349.

[26] W. Shen, X. Li, A. Doan, Constraint-based entity matching, in: Nat. Conf. on Artificial Intelligence and Innovative Applications of Artificial Intelligence Conf., AAAI Press / The MIT Press, 2005, pp. 862–867.

[27] M. A. Hernández, S. J. Stolfo, Real-world data is dirty: Data cleansing and the merge/purge problem, Data Mining and Knowledge Discovery 2 (1998) 9–37.

[28] L. O. Evangelista, E. Cortez, A. S. da Silva, W. M. Jr., Adaptive and flexible blocking for record linkage tasks, Journal of Information and Data Management 1 (2010) 167–182.

[29] M. Michelson, C. A. Knoblock, Learning blocking schemes for record linkage, in: Nat. Conf. on Artificial Intelligence and Innovative Applications of Artificial Intelligence Conf., AAAI Press, 2006, pp. 440–445.

[30] M. Bilenko, R. J. Mooney, Adaptive duplicate detection using learnable string similarity measures, in: ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, ACM, 2003, pp. 39–48.

[31] 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.

[32] M. Paganelli, F. D. Buono, M. Pevarello, F. Guerra, M. Vincini, Automated machine learning for entity matching tasks, in: Int. Conf. on Extending Database Technology EDBT, OpenProceedings.org, 2021, pp. 325–330.

[33] I. Bhattacharya, L. Getoor, A latent dirichlet model for unsupervised entity resolution, in: SIAM Int. Conf. on Data Mining, SIAM, 2006, pp. 47–58.

[34] M. Gheini, M. Kejriwal, Unsupervised product entity resolution using graph representation learning, in: SIGIR Workshop on eCommerce @ ACM SIGIR Int. Conf. on Research and Development in Information Retrieval, volume 2410, CEUR-WS.org, 2019.

[35] U. Brunner, K. Stockinger, Entity matching on unstructured data: An active learning approach, in: Swiss Conf. on Data Science SDS, IEEE, 2019, pp. 97–102.

[36] 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.

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

[38] V. V. Meduri, L. Popa, P. Sen, M. Sarwat, A comprehensive benchmark framework for active learning methods in entity matching, in: SIGMOD Int. Conf. on Management of Data, ACM, 2020, pp. 1133–1147.

[39] M. Sariyar, A. Borg, K. Pommerening, Active learning strategies for the deduplication of electronic patient data using classification trees, Journal of Biomedical Informatics 45 (2012) 893–900.

[40] P. Nodet, V. Lemaire, A. Bondu, A. Cornuéjols, A. Ouorou, From weakly supervised learning to biquality learning: an introduction, in: Int. Joint Conf. on Neural Networks (IJCNN), IEEE, 2021, pp. 1–10.

[41] A. Zakrzewska, D. A. Bader, Aging data in dynamic graphs: A comparative study, in: Int. Conf. on Advances in Social Networks Analysis and Mining, ASONAM, IEEE Computer Society, 2016, pp. 1055–1062.