Frost: A Platform for Benchmarking and Exploring Data Matching Results

Martin Graf
martin.graf@hpi-alumni.de
Hasso Plattner Institute
University of Potsdam, Germany

Lukas Laskowski
lukas.laskowski@student.hpi.de
Hasso Plattner Institute
University of Potsdam, Germany

Florian Papsdorf
florian.papsdorf@student.hpi.de
Hasso Plattner Institute
University of Potsdam, Germany

Florian Sold
florian.sold@student.hpi.de
Hasso Plattner Institute
University of Potsdam, Germany

Roland Gremmelspacher
roland.gremmelspacher@sap.com
SAP SE
Walldorf, Germany

Felix Naumann
felix.naumann@hpi.de
Hasso Plattner Institute
University of Potsdam, Germany

Fabian Panse
fabian.panse@uni-hamburg.de
University of Hamburg, Germany

ABSTRACT

"Bad" data has a direct impact on 88% of companies, with the average company losing 12% of its revenue due to it. Duplicates – multiple but different representations of the same real-world entities – are among the main reasons for poor data quality, so finding and configuring the right deduplication solution is essential. Existing data matching benchmarks focus on the quality of matching results and neglect other important factors, such as business requirements. Additionally, they often do not support the exploration of data matching results.

To address this gap between the mere counting of record pairs vs. a comprehensive means to evaluate data matching solutions, we present the Frost platform. It combines existing benchmarks, established quality metrics, cost and effort metrics, and exploration techniques, making it the first platform to allow systematic exploration to understand matching results. Frost is implemented and published in the open-source application Snowman, which includes the visual exploration of matching results, as shown in Figure 1.

PVLDB Reference Format:
Martin Graf, Lukas Laskowski, Florian Papsdorf, Florian Sold, Roland Gremmelspacher, Felix Naumann, and Fabian Panse. Frost: A Platform for Benchmarking and Exploring Data Matching Results. PVLDB, 15(12): 3292 - 3305, 2022.
doi:10.14778/3554821.3554823

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/HPI-Information-Systems/snowman.

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554823

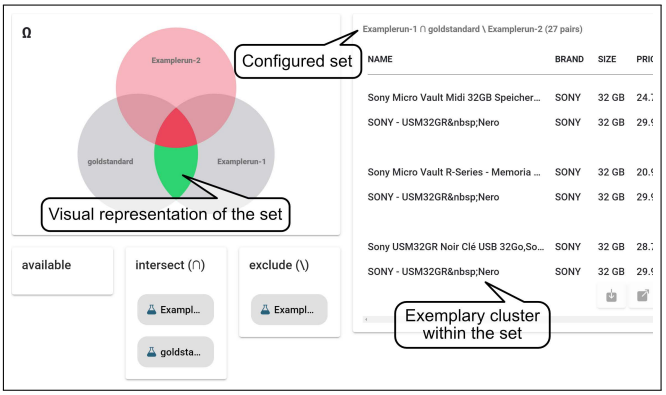

Figure 1: Exploring data matching results in Snowman. This figure shows the ground truth matches that Examplerun-1 found and Examplerun-2 did not find.

1 DATA MATCHING

Businesses and organizations rely heavily on structured data in databases. These databases often contain errors, such as outdated values, typos, or missing information, leading to large costs and non-monetary damage [16]. One prominent aspect of inaccurate data is (fuzzy) duplicates – the presence of multiple but different records representing the same real-world entity. Beyond sloppy data entry, duplicates emerge in further situations, in particular when integrating data from multiple sources. To address the issue of duplicates, various commercial and research systems to detect such duplicates have been developed [9, 12].

Systems detecting duplicates are generally referred to as (data) matching solutions, deduplication solutions, or entity resolution systems. They can be broadly categorized into two groups. Rule-based solutions are configured by hand-crafted matching rules to detect when a pair of records is a duplicate. An example rule in the context of a customer dataset could state that a high similarity of the surname is an indicator for duplicates, but a high similarity of customer IDs is not. Supervised machine learning models, on the

other hand, are trained by domain experts who label example pairs from the dataset as *duplicate* or *non-duplicate*.

1.1 A Data Matching Benchmark Platform

To find the best matching solution for a specific use case and to help configure it optimally, different benchmarks for comparing matching results have been developed. Typical data matching benchmarks consist of a dataset, a ground truth annotation, and sometimes information on how to evaluate the performance. To answer the question of which matching solution is best, all competing matching solutions are run against the dataset. Then, the results are compared to the ground truth annotation. Finally, performance metrics scores, such as precision, recall, and f1 score are determined for each matching solution and contrasted.

While many such benchmarks exist (see Section 2), the community is lacking a platform to easily (and interactively) compare the results of running against multiple such benchmarks, or to easily compare multiple systems or multiple system configurations against the same benchmark.

Almost all existing matching solution evaluation techniques focus on the quality of matching solution results. However, other aspects, such as business factors, also affect their usefulness in real-world deployment scenarios. Traditional metrics often provide only a quantitative overview over the performance of matching solutions. Qualitative analyses, such as behavioral analytics for matching solutions, are scarce in the matching context – despite having a high relevance for common use cases, such as fine-tuning a matching solution. To address these issues, we make the following contributions to the field of data matching benchmarks, noting that we do not propose yet another benchmark or further benchmark datasets, but rather a platform to systematically explore and analyze data matching results, thus enabling (business) users to more easily assess and compare their results for any given dataset.

Our extensible data matching benchmarking platform **Frost** offers both traditional quantitative quality metrics, but adds effort measurements by aggregating relevant business factors, for example purchase costs and deployment type. It includes techniques to systematically explore and compare matching results, allowing qualitative inter-system and intra-system inspection. The combination of traditional quality metrics, soft key-performance-indicators (KPIs) and exploration techniques allows deep evaluations for the industrial context. We implemented the majority of Frost in the open-source application **Snowman** and prove its practical relevance within the industrial context by demonstrating the usability of the platform and efficacy of the resulting evaluation insights.

- To suit the enterprise context, Snowman is meant to operate in, we chose and optimized evaluation algorithms for a *completely portable, self-bundled application stack*, which does not require privileged permissions for installation or execution that might be difficult to acquire.
- The extensibility of Snowman with *additional data import formats as well as new evaluation techniques* is an intrinsic part of its modular architecture.
- Apart from traditional evaluation metrics, Snowman supports *exploration techniques that do not require a ground truth* for industry use cases with datasets lacking a ground truth.
- In a variety of projects at SAP, we were able to observe that Snowman's user interface fosters *engagement with data matching stakeholders beyond IT*. Therefore, we consider it to be the reference implementation of the ideas presented in Frost and a starting point for open-source collaboration.
- Snowman does not execute the matching solutions itself, but takes their results as input, which it then matches against the ground truth. This allows Snowman to be used quickly in many different environments with low resource requirements.

1.2 Formal Matching Process

In this section, we define the formal matching process, and introduce abbreviations and variables that we use throughout the paper.

A dataset D is a collection of records that may contain duplicates. A record pair is a set of two records $\{r_1, r_2\} \subseteq D$. We denote the set of all record pairs in D as $[D]^2 = \{ A \subseteq D \mid |A| = 2 \}$. A matching solution M is a function which takes a dataset D as input and outputs a disjoint clustering $\{C_1, C_2, ...\}$ of D. We call the output of a matching solution an *experiment*.

All pairs *within* a cluster C_i are predicted to be duplicates by the matching solution and called *matches*. All other pairs in $[D]^2$ are predicted to be no duplicates and called *non-matches*. Accordingly, a different representation for the clustering is a set of all matches $E \subseteq [D]^2$. E can be seen as a graph with a node for every record $r \in D$ and edges between all record pairs $\{r_1, r_2\} \in E$ (also called *identity link network* [34]). Because E represents a clustering of D, the graph is transitively closed, ensuring that if r_1 and r_2 are matches and r_2 and r_3 are matches, r_1 and r_3 are considered to be matches, too. Nevertheless, some real-world matching solutions output subsets of $[D]^2$ that are not transitively closed. Although the closure of such a result set could easily be created, this step often introduces many false positives [20, 31]. Instead, a clustering algorithm specific to the use case can be applied [20, 31].

While it is most common to evaluate the performance of matching solutions only via their final results, typical matching solutions consist of multiple steps [44]. Measuring the performance between these steps, as supported by Frost, can provide useful insights for tweaking specific parts of the matching solution and helps to find bottlenecks of matching performance. A data matching pipeline typically consists of the following steps:

(1) **Data preparation**: Segment, standardize, clean, and enrich the original dataset [40].
(2) **Candidate generation**: Create subset of candidate pairs that contains as many true duplicates as possible, for instance using blocking or windowing [10, 47].
(3) **Similarity-based attribute value matching**: Compute similarities between the records' attribute values for each candidate pair [9, 18].
(4) **Decision model / classification**: Given the similarities for each candidate pair, decide which candidate pairs are probably duplicates [9, 18]. Typically, this step produces a final similarity or confidence score for each candidate pair. A pair is matched if its score is higher than a specific threshold. We use the term *similarity* to refer to both similarity and confidence.

(5) **Duplicate clustering**: Given the set of high probability duplicate pairs, cluster the original dataset into disjoint sets of duplicates [20, 31].
(6) **Duplicate merging / record fusion**: Merge the clusters of duplicates into single records [5, 17, 32]

In the following sections, we first discuss related work (Section 2). Then, we discuss different means for benchmarking, including traditional quality measures and soft KPIs (Section 3). We introduce different evaluation techniques that allow for qualitative insights about matching solutions in Section 4. Finally, we showcase Snowman, our implementation of Frost, and present a short study on the impact of effort measurements in Section 5. A long version of this paper with further architectural details is available at [29].

2 RELATED WORK

We outline existing work on benchmarking and exploring matching solutions. First, we discuss existing benchmarks, then approaches that yield insights similar to our exploration techniques, and lastly other benchmark platforms. Please note that reviewing related work on actual matching solutions, such as JedAI [46] or Magellan [19], is beyond the scope of this paper, and we refer to corresponding surveys [9, 45].

2.1 Data Matching Benchmarks

Benchmarks evaluate the performance of matching solutions, laying the foundation for major parts of our benchmark platform Frost. We discuss benchmark datasets in more detail in Section 3.1. A list of prominent benchmarks, generators, and polluters is collected in [44]. Below, we discuss two recent benchmarks that are especially relevant to Frost.

The *Semantic Publishing Instance Matching Benchmark* (SPIMBench) consists of a data generator capable of value, structural, and logical transformations producing a *weighted gold standard* and a set of metrics for evaluating matching performance [55]. The weighted gold standard includes a history of which transformations were applied to generated duplicates to allow a detailed error analysis. While SPIMBench is helpful to optimize duplicate detection within RDF datasets, it cannot be used with relational data that is common in our industrial context.

Crescenzi et al. propose the flexible schema matching and deduplication benchmark Alaska [14]. The authors profiled their datasets with traditional profiling metrics and three new metrics for measuring heterogeneity, namely *attribute sparsity*, *source similarity*, and *vocabulary size*. In work from 2020 Primpeli and Bizer focused solely on profiling benchmark datasets and grouped 21 benchmarks according to five profiling metrics, namely *schema complexity*, *textuality*, *sparsity*, *development set size*, and *corner cases* [49]. Such profiling metrics allow for better comparability of quality metrics from different benchmarks because the profiled factors can be considered. Moreover, profiling metrics that measure the difficulty or heterogeneity of datasets are a crucial step towards finding representative datasets for a given matching task when no ground truth annotations exist.

Frost supports a wide range of profiling metrics for measuring how similar a benchmark dataset and a real-world dataset are (see Section 3.1.3). Additionally, we use the notion of attribute sparsity proposed by Crescenzi et al. for classifying errors of matching solutions (see Section 4.4).

2.2 Exploration Opportunities

There has been surprisingly little work on techniques to systematically explore and understand matching results. SIMG-VIZ interactively visualizes large similarity graphs and entity resolution clusters [52], helping users to detect errors in the duplicate clustering stage. This is useful for improving the clustering algorithm and can give an overview on the matching result. Yet, only a limited number of possible errors are highlighted and large graphs easily overwhelm users. To counteract these problems, we propose techniques that help users to detect errors within the decision model. Specifically, we reduce the amount of information presented to the user by filtering out irrelevant data, sorting it by interestingness (Section 4), and enriching it with useful information about the error.

NADEEF/ER introduces additional investigation techniques for the rule-based approach NADEEF [24]. NADEEF/ER offers users a complete suite for rule-based entity matching, including an exploration dashboard to analyze matching results, for example the influence of each individual rule on the result. Compared to NADEEF/ER, Frost uses a more generic approach to evaluate matching results, as it supports a broad variety of matching solutions.

Matching solutions utilizing active learning, such as proposed by Sarawagi and Bhamidipaty [54], try to minimize review cost by asking human annotators only about uncertain matching decision; they are shown only uncertain matching decisions, and thus, they can understand weaknesses of the matching solution. Qian et al. introduced SystemER [50] as an active-learning-based entity resolution pipeline that uses solely rules comprehensible to humans, thus explaining individual matching decisions.

2.3 Benchmark Platforms

In [59], the authors described an early benchmark platform for XML data, measuring effectiveness (matching quality) and efficiency (runtime). Frost integrates both effectiveness and efficiency measurements, but is not limited to them.

A new measurement dimension, *effort*, was proposed in the benchmark platform FEVER [38]. Next to quality metrics, such as precision and recall, FEVER allows measuring the effort to configure a matching solution run by specifying labeling and parametrization effort. These KPIs can be compared in effort-metric diagrams, answering questions such as "How much effort is needed to reach 80% precision?" Frost builds on this idea by integrating business requirements to support the decision-making process of selecting a matching solution. For instance, it allows the comparison of matching solutions based on context-sensitive effort measurements, but also on further KPIs, such as deployment type and costs.

In later work, the authors used FEVER to evaluate different matching solutions. They found that "some challenging resolution tasks such as matching product entities from online shops are not sufficiently solved with conventional approaches based on the similarity of attribute values" [39]. This insight emphasizes the need for a comprehensive benchmark platform and the ability to systematically explore matching results.

Another benchmark platform, GERBIL, is based on the BAT framework [13] and provides 46 datasets, 20 matching solutions and eight tasks [51]. New matching solutions can be integrated by conforming to a REST API. Afterwards, they can be evaluated with the available datasets and tasks. The importance of such a platform is outlined by the fact that the GERBIL community already carried out more than 24,000 evaluations with GERBIL. While GERBIL is useful for evaluating quality metrics in different scenarios, it does not provide a way to compare soft KPIs or to explore matching results. Thus, we see it as very useful for finding the best among a selection of matching solutions. Frost also provides a selection of these quality metrics. However, it integrates evaluations allowing developers to gradually improve their matching solutions, such as metric-metric diagrams (see section 4.5.1), as well.

Some data matching execution frameworks also work as (partial) benchmark platforms. *DuDe* is a modular duplicate detection toolkit [21], consisting of six components to facilitate the entire matching process. One of those components, the postprocessor, can evaluate the performance of the matching solution in each run. For this purpose, metrics, such as precision and recall, are calculated. This reduces the feedback loop between running a matching solution and interpreting performance results, and allows comparing different experiments performed with DuDe. On the other hand, DuDe does not support general comparability between matching solutions, because many matching solutions use other matching frameworks or do not use a framework at all. A newer approach to this concept is the weakly supervised Panda platform [60], which uses user-created labelling functions to solve a given matching task. It allows for constant feedback on the performance of a certain labeling function and also offers debugging tools. Nevertheless, its use-cases are rather limited, as it also does not allow general comparability between arbitrary matching solutions.

3 BENCHMARKING MATCHING SOLUTIONS

Frost is a platform that supports users in evaluating their matching solutions using arbitrary data matching benchmarks. A data matching benchmark typically consists of

- One or more dirty *datasets* containing duplicates. These duplicates can be within, but also between, the individual datasets (intra-source vs. inter-source duplicates).
- A *gold standard* modeling the ground truth, i.e., the correct duplicate relationships between the given data records.
- A set of *quality metrics* to evaluate the given matching solutions. These can be metrics that compare these solutions' results with the gold standard, such as recall or precision [42], but also metrics that measure some inherent properties of these results, e.g., the number of pairs that are missing to transitively close the set of discovered matches or some soft KPIs, such as the effort that is needed to compute them.

3.1 Benchmark Datasets

A good benchmark should meet several conditions. (i) Most importantly, its ground truth annotation should be as correct and complete as possible (see Section 3.1.1). (ii) Second, to generalize well, its data and error patterns should be real or at least realistic (see Section 3.1.2). (iii) Third, the dataset should contain some so-called *corner cases* [49] to push matching solutions to their limits and reveal their quality differences. (iv) Finally, the dataset must be compatible with the objectives of the evaluation. For example, evaluating a matching solution focused on scalability requires a large dataset with millions of records, while the evaluation of a clustering algorithm requires a dataset with duplicate-clusters of various sizes.

3.1.1 Gold Standards. To measure the correctness of an experiment, we need a reference solution against which we can compare the result of the experiment. This solution is also called *gold standard* or *ground truth* and should accurately reflect the true state of the real-world scenario as defined by the use case (e.g., matching by household vs. by person).

The truth about the correct duplicate relationships between the records of a dataset D can be captured in different ways. The most common approach is to store a list of all pairs of duplicate records (or their IDs respectively) in a separate file. The gold standard, however, typically represents complete knowledge about the correct duplicate relationships and thus corresponds to a final matching solution [42], i.e., it is a clustering of D where every record belongs to exactly one cluster. Thus, the gold standard can also be modeled within the actual dataset by adding an extra attribute that associates each record with its corresponding cluster. Frost supports both formats, making importing new gold standards as easy as possible.

3.1.2 Reference Datasets. Many users who need a benchmark platform have their own use cases with their own datasets. Since the true duplicate relationships within these datasets are usually unknown (this is, after all, the reason matching solutions are applied), the performance of a matching solution cannot be evaluated on the whole dataset of the use case itself. Instead, the evaluation is frequently performed on a small subset of the dataset or on a similar reference dataset (see Section 3.1.3). Sometimes, as is also the case within SAP, manually annotated datasets from previous cleaning processes are available and can be reused.

Reference datasets can originate from the real world or can be artificially created [44]. In real-world datasets, the true duplicate relationships need to be labeled by domain experts. The data matching community has compiled several such datasets over the past few decades, which are publicly available via various sources, such as the Magellan Data Repository [15]. The artificial creation of test data can be automated. Examples of such test data generators are TDGen [2], GeCo [11], LANCE [56], BART [1], or EMBench++ [35].

Our reference implementation Snowman already includes a number of popular benchmark datasets, such as Cora and CDDB [21]. However, to allow easy use of any dataset (whether real-world or artificially created), it also supports an easy creation of custom importers (see Section 5.1).

3.1.3 Finding a Representative Benchmark Dataset. Researchers usually want to test their newly developed matching solutions under different conditions, and therefore like to use benchmark datasets that differ in their characteristics. To achieve this, they either create these artificially with the help of generators or make use of the numerous datasets provided by the community. In contrast, practitioners usually do not develop new matching solutions,

but must use an existing one to detect all duplicates within their use case-specific datasets. Thus, while researchers want to evaluate a particular solution on different datasets, practitioners aim to evaluate different solutions on a particular dataset. Therefore, practitioners cannot just take any benchmark dataset for their evaluation, but strive to find one that is similar to their use case dataset so that they can estimate the performance of different matching solutions on the latter by evaluating them on the first. Logically, the performance estimated in this way is only meaningful if both datasets pose similar challenges to the matching solutions. This makes finding a suitable benchmark dataset difficult.

To assist users in this search process, Frost includes a list of features impacting matching difficulty and provides decision matrices to compare a given use case dataset with several benchmark datasets based on these features. It remains to the experts to determine how important the individual features are for their use case, and to select the benchmark dataset that they think is best suited for their evaluation goals. In addition to the features proposed by Crescenzi et al. [14] and Primpeli et al. [49], such as sparsity, textuality, and schema complexity, additional useful features are:

- **Domain**: The domain of both datasets should match or be closely related.
- **Record count**: Draisbach and Naumann showed that dataset size has influence on the optimal similarity threshold [22]. Thus, using a benchmark dataset with similar size compared to the use case dataset may yield more representative results.
- **Number and size of duplicate clusters**: The amount and size of duplicate clusters in the ground truth annotation of the benchmark dataset should closely resemble that of the use case dataset. Because the ground truth annotation for the use case dataset is unknown, these numbers have to be estimated. Heise et al. developed a method for this estimation [33].
- **Matching solution**: The matching solution itself may provide valuable insights into how similar both datasets are. Relevant features include (i) metrics for approximating quality without requiring a ground truth annotation (see Section 3.2.3), (ii) the similarity of the clusterings of the matching solution on use case and benchmark dataset, and (iii) the number of pairs from the transitive closure that are missing in the solution's classification results on both datasets. Note that some of these metrics require normalization if certain properties of the datasets, such as record count, do not match.
- **Vocabulary similarity**: Vocabulary similarity *VS* quantifies the similarity of the vocabularies of two datasets. Similar vocabularies might cause similar behavior of the matching solution. We calculate this similarity using the Jaccard coefficient:

$$VS(D_1, D_2) := \frac{|vocab(D_1) \cap vocab(D_2)|}{|vocab(D_1) \cup vocab(D_2)|}$$

  where D_1, D_2 are datasets and $vocab(D_i)$ is the vocabulary-set of D_i, tokenized by whitespace.

3.2 Measuring Data Matching Quality

When ground truth annotations are available, a multitude of different metrics can be calculated. While some are generally used and considered essential, others suit specific needs. To be universally useful but highly adaptable, Frost focuses on many well-known metrics, but can be extended easily by any other metrics. We distinguish between pair-based metrics and cluster-based metrics.

	Positive		Negative	
Predicted Positive	$E \cap G$	(TP)	$E \setminus G$	(FP)
Predicted Negative	$G \setminus E$	(FN)	$([D]^2 \setminus E) \setminus G$	(TN)

Figure 2: Confusion Matrix. Comparison of experiment E against ground truth annotation G on dataset D as sets of pairs.

3.2.1 Pair-based Metrics. To compare an experiment E against a ground truth annotation G of a dataset D as sets of pairs, the confusion matrix can be defined as shown in Figure 2.

This matrix allows the calculation of all metrics known from the context of binary classification. Pair-based metrics do not require the identity link network of experiment E to be transitively closed. Therefore, they can be used to calculate matching quality even at intermediate stages of the matching pipeline. For example, pair-based metrics allow measuring the performance of the candidate generation phase. Additionally, they directly contrast the quality of matching solutions that return clusters with matching solutions that return pairs (and do not necessarily output transitively closed identity link networks) [59]. Note that pair-based metrics implicitly give more weight to larger clusters, as each pair of records within a cluster is counted towards the result.

Another weakness of pair-based metrics is the fact that in the real-world there is almost always a large imbalance between true positives and true negatives (called class imbalance) [9]. While a dataset of n tuples usually contains only $O(n)$ duplicate pairs, it may consist of up to $O(n^2)$ non-duplicate pairs. Metrics that judge upon correctly classified non-duplicates (true negatives) are therefore considered unreliable. For example, the *accuracy* of matching results compared to a ground truth might be close to 1, even when all record pairs were classified as non-duplicates.

Frost supports a wide selection of pair-based metrics considering the above observations including the common precision, recall and f1 score [42], but also more special ones, such as the Reduction Ratio [37], the f* score [30], the Fowlkes-Mallows index [26], and the Matthews correlation coefficient [8].

3.2.2 Cluster-based Metrics. Cluster-based metrics are most often computed using similarities between clusters of the ground truth and the experiment [3, 42, 43]. An advantage of cluster-based metrics is that they are immune to the class imbalance described above. On the other hand, they cannot be used to directly evaluate matching solutions that produce non-transitively closed sets of matches [59]. For example, the output of intermediate stages of a matching pipeline is usually not clustered.

Frost utilizes several prominent cluster-based metrics including the closest-cluster-f1 score [4], the Variation of information [41] and the Generalized merge distance [42].

3.2.3 Evaluating Quality Without Ground Truth. In many real-world use cases, labeled data is not available. Frost also supports

metrics and evaluation strategies that try to estimate matching quality on real-world datasets without ground truth annotations:

- Idrissou et al. show that redundancy in identity link networks correlates with high matching quality [34]. Interestingly, their experiments show a "very strong predictive power of [...their] e_Q metric for the quality of [...identity link networks] when compared to human judgement" [34].
- The minimum number of pairs that must be added to or removed from the set of detected matches for it to be transitively closed is another relevant metric. The larger this number, the more inconsistent the proposed matches.
- Duplicate records are typically closer to each other than to other records. Thus, the compactness of the individual clusters and the sparsity of their local neighborhoods as proposed by Chaudhuri et al. [7] can estimate the quality of the whole matching result. To calculate compactness and sparsity, however, we need similarity scores between the individual records provided by the matching solution for both matches (compactness) and close non-matches (sparsity).
- If the set of detected matches is not transitively closed, we can achieve this closeness by applying several clustering algorithms [20, 31], such as maximum clique clustering or Markov clustering. Here we can assume that the more similar the resulting clusterings are, the more consistent are the initially discovered matches. Again, many clustering algorithms (e.g., Markov clustering) require similarity scores for the matches.
- We can compare the matching result with those of other matching solutions applied to the same dataset. The consensus on an individual matching decision (match or non-match) is a good indicator on its correctness [58]. Thus, the total number of deviations from the majority votes can be used to estimate the quality of the whole matching result.

Many of these aspects can be used not only to calculate a metric, but are predestined to guide users in the exploration of their matching results, e.g., by presenting record pairs that are likely misclassified by their solution (false positive or false negative). We describe these exploration techniques in Section 4.

3.3 Soft KPIs: Effort and Cost

Every matching solution has different advantages and disadvantages and requires a different type of configuration. As an example, supervised machine learning approaches need training data, whereas rule-based approaches need a set of rules. When deciding which matching solution to use for a specific use case, these properties are of importance, as they influence how expensive and time-consuming it is to employ the solution. To assist the decision process, Frost includes a benchmark dimension for soft key performance indicators (KPIs), which models such business aspects.

The main goal of these soft KPIs is to provide users a comparable overview of relevant, non-performance properties of matching solutions and experiments. Most of these KPIs model the human effort (i.e., the amount and complexity of work) necessary to perform a specific task. While many non-effort KPIs are objective and therefore easily comparable, effort is subjective and has to be estimated. People with varying skills often have different opinions on how long it takes to configure a matching solution. Therefore, we measure such effort using two variables: (i) The amount of time an expert needs to finish the task (HR-amount), and (ii) the expert's skill level from 0 (untrained) to 100 (highly skilled). HR-amount and expertise are interdependent. When comparing two persons with different expertise, usually, the person with more expertise is faster. Chatzoglou and Macaulay state that low experience is an indicator for increased time and cost, and that experience is considered an important factor for productivity [6]. Expertise is typically related to pay level. Therefore, combining HR-amount and expertise yields a rough estimation of the monetary cost of performing the task.

The soft KPIs supported by Frost can be categorized into three classes:

- **Lifecycle Expenditures**: One important business aspect is the expenditure for integrating and operating a matching solution over its entire life-cycle. Based on life-cycle cost analysis (LCCA) [23], Frost supports several soft KPIs to represent the different product phases, such as the general costs of the life-cycle or the effort required to get the matching solution ready for production within a company's ecosystem and configure the matching solution for its particular use case, where we distinguish between domain-specific configurations (e.g., the manual labeling of training data) and technique-specific configurations (e.g., the selection of algorithms).
- **Categorical Soft KPIs**: Apart from lifecycle expenditures, there are a few more aspects relevant for businesses: These include the (i) development types (e.g., on-premise or cloud-based), (ii) interfaces (e.g., GUI, API, CLI), and (iii) techniques (e.g., rule-based, clustering, or probabilistic decision models) supported by the given matching solution.
- **Soft KPIs of Experiments**: Frost supports measuring and evaluating soft KPIs on a per experiment basis. This includes the effort needed to set up the experiment (e.g., acquisition of suitable test data) and the runtime that the matching solution required to complete the experiment.

The underlying effort and cost values need to be provided by the users. However, Frost helps manage these numbers beyond single experiments and supports calculating, comparing and evaluating all the aforementioned soft KPIs that are based on these numbers. Frost supports two different evaluation techniques for soft KPIs. On the one hand, it provides a decision matrix including all above metrics side by side. Importantly, this decision matrix also includes quality metrics to provide a holistic view of the attractiveness of the compared solutions. On the other hand, Frost provides users the ability to aggregate metrics. For example, to estimate costs, the effort-based metrics can be converted into costs as described above and added to general costs. Because this aggregation depends on the use case, Frost does not pre-define aggregation strategies, but provides a framework for aggregating soft KPIs and quality metrics into use case specific KPIs.

As proposed and used by Köpcke et al. [38, 39], Frost aids users in analyzing soft KPIs for experiments with a diagram-based approach. This helps answer questions, such as how much effort is needed to achieve a specific metric threshold (e.g., 80% precision), whether increased runtime yields better results, or how good a matching solution is out-of-the-box versus how much effort it takes to improve the results. The diagram is especially interesting when

experiments from multiple matching solutions are compared. Evaluations thereby become competitive and allow discovering different characteristics of the matching solutions.

4 EXPLORING DATA MATCHING RESULTS

The general workflow for improving matching solutions and arriving at a sufficient configuration is usually iterative. Thus, after one run has finished, its results need to be analyzed to gain insights about the solution's behavior. Afterwards, the matching solution can be refined accordingly and re-run. As motivated in Section 1.2, we present structured approaches to explore data matching results. Specifically, we reduce the amount of information presented to the user by filtering out irrelevant data, sorting it by interestingness, and enriching it with useful information about the type of error. Finally, we introduce diagram-based evaluations.

4.1 Set-based Comparisons

Manual inspection of experimental results can be a poor experience. As an example, some output formats consist solely of identifiers and thus require to be joined with the dataset to be helpful. Additionally, only limited information can be extracted by looking at results side-by-side; in practice usually more than two result sets are compared. A common use-case is to contrast multiple runs of the same matching solution with each other, or to evaluate differences between two distinct solutions and a ground truth.

Frost supports a generic set-based approach to result evaluation that enriches identifiers with the actual dataset record. The set operations *intersection* and *difference* can describe all partitions of the confusion matrix, as introduced in Section 3.2.1. As an exemplary evaluation, consider two result sets $E_1, E_2 \subseteq [D]^2$, where E_2 serves as ground truth. The subset of false positives is defined as the set of elements in E_1 that are not part of the ground truth E_2, or simply $E_1 \setminus E_2$. While the confusion matrix is limited to evaluating binary classification tasks with two result sets, the generic approach can compare multiple result sets.

As an intuitive visualization technique, Frost makes use of Venn diagrams. When n experiments are compared, these diagrams describe all $\binom{n}{2}$ possible subsets visually. A disadvantage with Venn diagrams is that they get very complex for larger numbers of sets. Venn diagrams of more than three sets need to use geometric shapes more advanced than circles [53]. Set-based comparisons and Venn diagrams in particular can help to answer a variety of evaluation goals, such as:

- Compare two matching solutions' result sets against a ground truth to discover common pairs. This evaluation can easily be visualized with circle-based Venn diagrams.
- Find shortcomings or improvements of a new matching solution compared to a list of proven solutions by selecting all duplicate pairs only the new solution detected.
- Create an experimental ground truth [58] from the intersection of multiple experiments.

Because exploration is supposed to be interactive, an implementation should provide vivid Venn diagrams. Clicking on regions should allow selecting the corresponding set intersection. Thereby, the desired configuration can be composed easily according to its visual representation.

4.2 Pair Selection Strategies

While set-based comparisons are useful on their own, real-world datasets can contain millions of records, making it unfeasible to examine all pairs in a set. Therefore, strategies to reduce the number of pairs shown are crucial. Frost supports a wide range of selection techniques to highlight relevant pairs which can be used separately or as a composition according to the current use case.

4.2.1 Pairs around the Threshold. For matching solutions that provide a meaningful similarity threshold, an easy section of the result to further investigate is located close to the similarity threshold, as it includes information on border cases. Pairs in this section are usually considered uncertain, as a slight shift of the threshold may change their state. Nevertheless, they still yield helpful insights about what is especially difficult for the matching solution. To select k pairs, one can either choose $\frac{k}{2}$ pairs above and below the threshold or based on a certain proportion. For instance, one interesting proportion is the ratio of incorrectly classified pairs above the threshold to below the threshold.

4.2.2 Incorrectly Labeled Outliers. Another group of interesting pairs lies further away from the threshold. For example, one could evaluate why the matching solution failed by searching for a common "misleading" feature among the selected pairs. Therefore, we allow selecting incorrectly labeled pairs that are the furthest away from the threshold.

4.2.3 Percentiles with Representatives. Sometimes, the goal is to get an overview over the matching quality before diving into details. For this, we support finding representative pairs from all parts of the result set. Conceptually, this strategy sorts result sets by a similarity score and then splits them into smaller partitions. Each of these partitions is then reduced to a few representative pairs that represent the matching solution's behavior within this partition.

Let E be a result (sub)set with m pairs that is split into k equally-sized partitions. To sample b representative pairs for each partition, different choices exist:

- **Random sampling**: b pairs are sampled randomly from each partition. While this technique is unbiased, it may also only yield uninteresting pairs and thereby no helpful insights.
- **Class-based sampling**: For a partition with k_T correctly and k_F incorrectly classified pairs, we randomly sample $b \cdot k_T/(k_T + k_F)$ correctly and $b \cdot k_F/(k_T + k_F)$ incorrectly labeled pairs. Thereby, we make sure to weigh the numbers of pairs according to the algorithms performance.
- **Quantile sampling**: Alternatively, b pairs can be sampled by selecting b quantiles, again based on the similarity score. For $b = 5$, this would mean to select quantiles 0, 0.25, 0.5, 0.75, and 1. This technique has the advantage of unbiasedly representing the different parts of the partition.

Additionally, we can label each partition with its confusion matrix and metrics. Thus, users can focus on those partitions with high error levels. A partition with few to no incorrectly labeled pairs is considered to be a confident section. In contrast, a section with many false positives and/or false negatives is very unconfident, and therefore deserves more attention.

4.2.4 Plain Result Pairs. As outlined in Section 1.2, Frost requires result sets to be transitively closed. On the one hand, this can lead to more realistic metrics. But on the other hand, it can also enlarge small result sets to a very large number of pairs and thereby possibly introduces a substantial number of false positives. Thus, Frost includes a selection strategy that will hide all pairs that were added by a clustering algorithm from a given result subset. What remains are all pairs that were originally labeled by a matching solution. To enable this, Frost requires information on which pairs were added during the clustering process and which were labelled by the matching solution itself.

4.3 Sorting Strategies

Besides reducing the result sets to smaller subsets, Frost also supports to sort pairs by their *interestingness* within a given subset. When relevant pairs are shown first, developers can gain insights more quickly to improve the matching solution's performance on a given dataset. The usefulness of the sorting procedure varies between strategies and use case. Below, we discuss several measures of interestingness of record pairs.

4.3.1 Similarity Score. A common score to rank any set of pairs is the similarity of a pair's records. Whenever similarity values are available for all pairs, this technique offers a view on the data from the matching solution's perspective.

4.3.2 Column Entropy. We also define independent scores that were not part of a matching solution's output. For each token t within a given cell, let $prob_t$ be its occurrence probability within the cell and $columnProb_t$ the probability within the column. The cell entropy is calculated by:

$$\sum_{token\ t} prob_t \cdot -log(columnProb_t)$$

where the second factor describes a token's information content within its column. This formula is close to the original definition of entropy by Shannon [57], but is applied column-wise. For a given pair $p = \{r_1, r_2\}$, we can calculate its entropy as the sum of all cell entropies of both records. Pairs with a particularly high entropy score contain many rare tokens and are therefore expected to be easier to correctly classify. Depending on dataset and matching solution, we may observe a divergence in the distribution of entropy among the confusion matrix. If not, we can still use entropy as a score to sort pairs within a subset of the result set(s).

4.4 Error Analysis

To better understand why a pair was misclassified by a certain matching solution, one could analyze why a similar pair was labelled correctly. Thereby, one can gain insights on why the matching solution came to a false conclusion and find errors within the decision model. Frost allows enriching a misclassified pair $p_f = \{e_{f,1}, e_{f,2}\}$ with a correctly classified pair $p_t = \{e_{t,1}, e_{t,2}\}$. We search for p_t by considering only correctly classified pairs and selecting the one which is most similar to p_f. We describe the similarity between the pairs p_f and p_t with vectors

$$\mathbf{v}_{direct} = \begin{pmatrix} sim(e_{f,1}, e_{t,1}) \\ sim(e_{f,2}, e_{t,2}) \end{pmatrix} \text{ and } \mathbf{v}_{cross} = \begin{pmatrix} sim(e_{f,1}, e_{t,2}) \\ sim(e_{f,2}, e_{t,1}) \end{pmatrix}$$

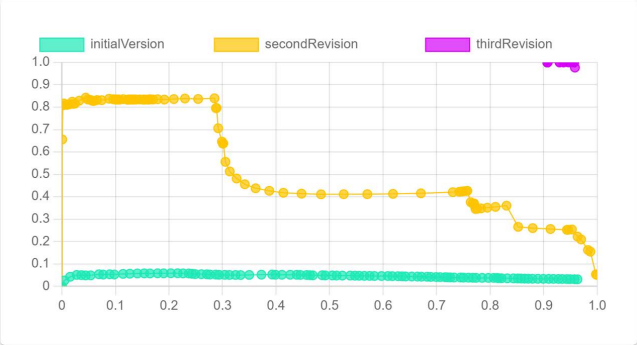

Figure 3: Precision-Recall Curve. This diagram, taken from our implementation of Frost (Snowman, Section 5), plots recall against precision for a given set of similarity thresholds.

To compare these vectors with each other, we convert each one into a scalar distance measure. For this, the Minkowski metric with $q \in [1, 2]$ is used against $\vec{0}$ as the reference point:

$$distance(\mathbf{v}) = D(\mathbf{v}, \vec{0}) = (|\mathbf{v}_1 - 0|^q + |\mathbf{v}_2 - 0|^q)^{\frac{1}{q}}$$

For $q = 1$, this equals the Manhattan distance and for $q = 2$ the Euclidean distance. It depends on the user to choose $q \in [1, 2]$ depending on the use-case. Finally, we define the distance score of p_t against p_f as

$$score = \max\{distance(\mathbf{v}_{direct}),\ distance(\mathbf{v}_{cross})\}$$

Whichever candidate pair p_t scores highest is then selected.

To receive best results, all possible pairs should include a similarity score. Since this would require the matching solution to compare $O(n^4)$ values for a dataset of size n, a possible extension to Frost could be to calculate a simple distance measure for a set of promising pairs internally.

4.5 Diagram-based Exploration

All strategies so far are for set-based comparisons and either limit the number of pairs shown (Section 4.2), sort them (Section 4.3) or add additional information (Section 4.4). Here, we introduce a set of diagrams that aid in understanding a matching solution's behavior.

4.5.1 Metric/Metric Diagrams. For matching solutions that return similarity scores, one objective is to find a good similarity threshold. Frost utilizes metric/metric diagrams for this objective. Those diagrams compare two quality metrics against each other for a given set of similarity thresholds, and each data point is based on a different similarity threshold. A commonly known diagram is the precision/recall curve (see Figure 3). With it, one can visually observe which point (and thereby similarity score) yields the best ratio between both metrics. Another well-known diagram is the ROC curve [28] plotting sensitivity (*also:* recall) and specificity against each other. This may not be suitable for every use case, though, as specificity depends on the count of true negatives. However, depending on the shape of the curve, these diagrams may yield insights upon both a good similarity threshold and the reliability of the matching solution. Next to using metric/metric diagrams in

isolation, multiple diagrams between multiple matching solutions can be compared for competitive insights.

A limitation of this technique is that it heavily depends on how many pairs have a similarity score assigned. In practice, metrics sampled at similarity scores significantly lower than the similarity threshold of the matching solution may not be representative because pairs with such low similarity scores are often excluded from the result set.

4.5.2 Attribute Sparsity. As missing attribute values are known to influence and complicate matching tasks [14, 48, 49], we want to further investigate which attributes precisely affected a matching solution's performance most. Attribute Sparsity as introduced by Crescenzi et al. measures how often attributes are, in fact, populated within a given dataset [14]. Thereby, the authors profile a given dataset's difficulty together with additional profiling dimensions. Since we aim to profile a matching solution's result set instead, we define a metric that measures the influence of null-valued attributes by the count of incorrectly assigned labels as follows: Let D be a dataset and a be an attribute of D. We define $nullCount(a)$ as the count of pairs in $[D]^2$ where at least one record of the pair is null in attribute a. Additionally, we define $falseNullCount(a)$ as the count of incorrectly classified pairs in $nullCount(a)$ and $nullRatio(a)$ as:

$$nullRatio(a) = \frac{falseNullCount(a)}{nullCount(a)}$$

In contrast to the raw $nullCount$, immoderate amounts of null occurrences within an attribute do not bias the $nullRatio$. Calculating the metric for all attributes a in D yields a statistical distribution. Graphical representations that show scores for discrete buckets, such as bar charts, support comparing measured scores. Thereby we can observe the following: attributes with high *nullRatio* scores are statistically highly relevant for the matching decision as their absence could be related to many incorrectly assigned labels [49]. For instance, we observed that the ratio reveals a high significance for the attributes *author* and *title* in the Cora dataset [21] for the Magellan matching solution [36].

If the revealed significant attributes do not match the expectation, this likely comes down to one of two reasons:

- **Semantic mismatch**: A semantic mismatch exists if the matching solution weighs attributes heavily that are semantically irrelevant for a matching decision. For instance, a matching solution learned to weigh attributes b and c more significantly while a and b are more important in reality. A semantic mismatch is an indication that the provided rule set or the learned network's weights are not consistent with the domain of the given dataset.
- **Material mismatch**: A material mismatch exists if the statistically assumed significance of attributes is not adequate for the underlying dataset. For instance, a matching solution weighs attributes a and b while the underlying dataset is often null in these attributes. This mismatch might occur when a matching solution is used on another dataset than it was initially optimized for (for example due to transfer learning).

A downside is that *nullRatio* relies on interspersed null values within the dataset D and a meaningful and sophisticated schema. Such a schema contains several attributes that provide meaningful information, for example street and city split instead of combined in a single address field. For instance, the Cora dataset fulfills the requirements with an average attribute sparsity of 0.58 and a schema with 17 attributes [21].

In conclusion, the exploration of *nullRatio* allows insights into the matching solution's handling of null values.

4.5.3 Attribute Equality. Similar to attribute sparsity, Frost allows investigating the influence of equal attribute values on the matching process, too. Equal attribute values can indicate a duplicate pair, although equality in one attribute is usually not a sufficient criterion. For instance, while attributes, such as the person's name, may be sufficient for a match, others, such as post code, may not. Therefore, Frost includes attribute equality as a dimension to statistically analyze which equal attributes are related to incorrectly assigned labels significantly often.

Let D be a dataset and a be an attribute of D. First, we define $equalCount(a)$ as the count of pairs in $[D]^2$ where both records of the pair are equal in a. Second, we define $falseEqualCount(a)$ as the count of incorrectly classified pairs in $equalCount(a)$. We set:

$$equalRatio(a) = \frac{falseEqualCount(a)}{equalCount(a)}$$

A high $equalRatio(a)$ for a given attribute a indicates that the matching solution did not weigh the matching sufficiency of a correctly (either too high or too low). Again, calculating the metric for all attributes a in D yields a statistical distribution which, if compared across all attributes, can yield helpful insights. Similarly, bar charts can be used as an evaluation tool.

5 REFERENCE IMPLEMENTATION

In this section, we present our reference implementation of Frost called Snowman and perform two example evaluations with it. Figure 4 shows a screenshot that highlights the fact that Snowman addresses both data stewards (domain experts) and developers.

5.1 Snowman's Features

Snowman provides many features enabling developers to explore and evaluate data matching results. Next to evaluation techniques that require a ground truth, Snowman also supports those that operate solely on the matching results.

Besides traditional metric evaluation pages, Snowman has full support for our soft KPI dimensions from Section 3.3 and supports the main exploration concepts from Section 4. Below, we present a selection of evaluations that are already part of Snowman. A full list can be found in Snowman's online documentation[1].

- **Data matching expenditures**: Snowman implements both the decision matrix and the diagram for evaluating experiment level expenditures as described in Section 3.3.
- **Set-based comparisons**: Snowman supports intersecting and subtracting experiments and ground truths with the help of an interactive Venn-diagram as described in Section 4.1 (see Figure 1). To enhance the evaluation process, Snowman shows complete records instead of only entity IDs; if only intersection operators are used, clusters are grouped.

[1] https://hpi-information-systems.github.io/snowman

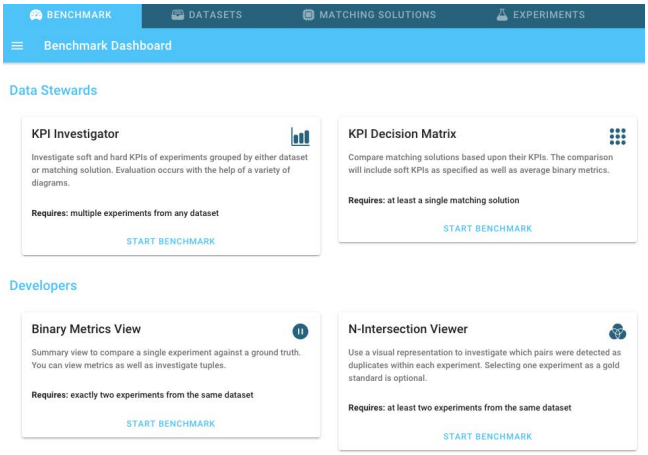

Figure 4: Snowman. The start screen of the reference implementation of Frost: Snowman. From here, the individual benchmark actions are available.

- **Evaluating similarity scores**: Snowman helps users find the best similarity threshold by plotting the metric/metric diagrams discussed in Section 4.5.1 (see Figure 3). It also allows to compare similarity functions of multiple matching solutions and multiple similarity functions of one matching solution.

Snowman provides a range of preinstalled benchmark datasets (including ground truth annotations) giving users the ability to easily evaluate and compare matching solutions in multiple domains without further imports. Besides, it supports a range of different dataset and experiment formats and provides a convenient interface for additional custom CSV-based formats as well as other file-based experiment or dataset formats through customized importers. Existing importers for experiments are 30-60 lines long and in the case of a CSV-based format as simple as defining the separator, quote, escape symbols and a mapping for rows to duplicate pairs or clusters. Snowman only requires the output that a matching solution provides. Further integration is not necessary.

5.2 Snowman's Architecture

Our application stack (see Figure 5) makes use of ElectronJS and splits Snowman into a NodeJS back-end and a ReactJS front-end. Both are built using TypeScript, increasing maintainability and simplifying the onboarding process of new contributors. More importantly, experiments show that many workloads are not significantly slower than for similar implementations in Java [27]. A more detailed architectural overview can be found in [29].

All communication between front-end and back-end occurs through a REST API specified according to the OpenAPI 3 standard. This allows third-party applications to integrate easily with Snowman, for example to ingest matching results directly from within a matching solution or to automatically retrieve evaluation results, for example directly within Python3 code. Furthermore, it means that Snowman can be deployed both locally and in a shared environment among multiple users. To be easily usable in corporate environments where administrative privileges are rare and device

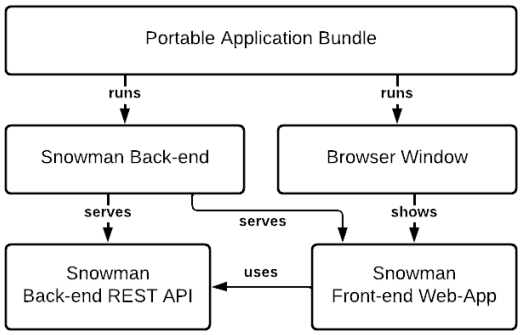

Figure 5: Architectural overview. Snowman's architecture is bundled into a single portable executable, but still offers a variety of extension points.

settings might be restrictive, Snowman is portable and requires no installation or external dependencies. All major operating systems are supported, including recent versions of Windows, macOS, Ubuntu, and Debian.

5.3 Snowman's User Experience

To be useful to enterprise data stewards, it is crucial for Snowman to provide results quickly for even the most demanding evaluations. Most developers only have limited access to huge data centers, and gold standards containing multiple billion records are rare. Therefore, we instead optimized Snowman so that it can run well on a typical enterprise laptop, but still provide results for medium-sized datasets in hundreds of milliseconds to a few seconds.[2]

To achieve this, Snowman optimizes every step of the evaluation process, beginning with optimizing matching results while they are imported into the tool: During import, a unique numerical ID is assigned to each record, allowing constant time access to records. Additionally, a clustering of the experiment is constructed. We do this because currently, nearly all calculations in Snowman are performed using transitively closed clusters instead of pairs, which leads to much faster runtimes (up to linear to the dataset length) in practice, compared to the quadratic number of pairs to be evaluated otherwise. Let D be the dataset that the matching solution was executed on, and let *Matches* be a set of matches predicted by the matching solution. The pre-calculations during the import of an experiment take $O(|Matches| \cdot log(|D|)$ *(to map the dataset's native IDs to numeric persistent IDs).*

Even when using clusterings instead of pairs to calculate results, in some cases, the run-time of evaluations on datasets with tens of thousands of records takes too long. As an example, one of the most demanding evaluations of Snowman is calculating pair-based metric/metric diagrams (see Section 4.5.1). A naïve approach to calculate these diagrams using clusterings is to sample metrics at different similarity thresholds without re-using insights from other thresholds. Running this algorithm on a dataset with 100 000 records and roughly 45 000 matches produced by an industry-grade matching solution on an enterprise laptop led to a runtime of roughly

[2] In case more compute power is required than available to the user on his local machine, Snowman can also be set up as a shared environment with its back-end hosted in the cloud.

Table 1: Runtime of Metric/Metric Diagrams. The table shows a comparison of the runtime of Snowman's optimized algorithm for pair-based metric/metric diagrams against a naïve approach. For each diagram, 100 different similarity thresholds were calculated.

Dataset	Record count	Matched pairs	Metric diagram		Approximate speedup factor
			Custom	Naïve	
Altosight X4	835	4 005	184ms	1.7s	9
HPI Cora	1 879	5 067	245ms	7.4s	30
FreeDB CDs	9 763	147	293ms	16.4s	56
Songs 100k	100 000	45 801	1.6s	43.9s	28
Magellan Songs	1 000 000	144 349	6.1s	6min 43s	66

44 seconds (see Table 1), which is longer than most people comfortably wait for results. To address this, we developed an efficient algorithm to compute metric/metric diagrams, which reuses intermediate results and dynamically builds a clustering while tracking relevant metrics. An analysis of the algorithm shows that the worst-case runtime (excluding the time of the optimization during import) is in $O(|D| + |Matches| \cdot s)$ where s is the amount of data points on the diagram. Additionally, the algorithm runs the faster, the more similar ground truth and experiment clusterings are. Table 1 confirms that, for real-world datasets, the algorithm is considerably faster compared to the naïve approach. As an example, for the above-mentioned dataset with 100 000 records and experiment with 45 000 matches, it took only a little under two seconds.

In summary, Snowman enables users to run most evaluations directly on their laptops without the need for special hardware or a compute cluster while still enabling fast and easy iterations.

5.4 SIGMOD Programming Contest

The ACM SIGMOD programming contest 2021 presented the participants with an entity resolution task [25]. The goal was to deduplicate three datasets and achieve the highest average f1 score. All participants were given the opportunity to use Snowman as a preconfigured evaluation tool to investigate matching results. After the contest finished, we analyzed five high performing matching solutions with our benchmark platform on the evaluation dataset Z_4. Three of the matching solutions used a machine learning approach, one used a rule-based approach, and one used a combination of rules and machine learning. In the following, we present key insights uncovered by application of Snowman:

For an initial overview, we used Snowman's N-Metrics Viewer to compare quality metrics, such as precision, recall, and f1 score. On average, the top-5 contest teams achieved an f1 score of 90.34% with 87.4% as the minimum and 92.7% as the maximum. These results are impressive, as the dataset constitutes a quite difficult matching task: most of the matching has to be based on unstructured, cluttered information in the attribute *name*.

As the performance of a matching solution is often strongly related to the selected similarity threshold, metric/metric diagrams as introduced in Section 4.5.1 can be used to find the optimal threshold. Using Snowman, we ascertained that two matching solutions had, in fact, not selected the optimal similarity threshold for their results. Selecting a higher similarity threshold would have increased their f1 score by 8% and 6%, respectively. Surprisingly, these observations are also true for the training dataset.

With Snowman, we identified three true duplicate pairs that were not detected by at least four solutions. This evaluation can be accomplished with the N-Intersection Viewer (see Figure 1) by subtracting all result sets from the ground truth. Interestingly, all three pairs include the record with ID *altosight.com//1420*. This is an indicator that this record is especially difficult to match, or all considered matching solutions make equal assumptions about related information representations.

These findings confirm that useful insights can be gained by coherently applying structured evaluation techniques and result exploration, emphasizing the need for a benchmark platform. In summary, Snowman can help to better understand a matching task as well as result sets and thereby accelerates the development of successful matching solutions.

5.5 Experimental Evaluation of Soft KPIs

We conducted a study to show the relevance of effort measurements when benchmarking data matching solutions, and to examine what impact spent *effort*, as discussed in Section 3.3, has on matching performance. We expected that matching solutions improve with additional effort invested into their configuration, and that the curve of a target metric (e.g., f1 score) asymptotically approaches an optimum – specific to each matching solution and dataset. To validate this expectation, we manually optimized three different matching solutions, ranging from rule-based to machine learning approaches, for a given dataset. Specifically, we deduplicated the SIGMOD contest's D4 dataset with the goal to optimize the f1 score achieved on the test dataset Z4 by using the training dataset X4 as well as its ground truth annotation. Throughout the process, we tracked the effort spent. Figure 6 illustrates how the f1 score evolved against the effort.

Each solution had a breakthrough point-in-time at which the performance increased significantly. Afterwards, all solutions reached a barrier at around 14 hours, above which only minor improvements were achieved. This could either mean that a major configuration change is required or that the maximum achievable performance for this matching solution on dataset D4 is reached.

Additionally, we analyzed the f1 score of the submissions from five top teams of the SIGMOD contest over time (see Figure 7). The matching quality of the different teams generally increased over time, but sometimes faced significant declines in matching

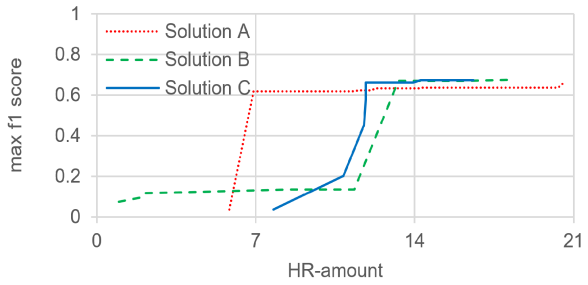

Figure 6: Maximum f1 score against effort spent (in hours). We optimized three solutions for the SIGMOD D4 dataset from scratch and tracked the effort spent throughout the process.

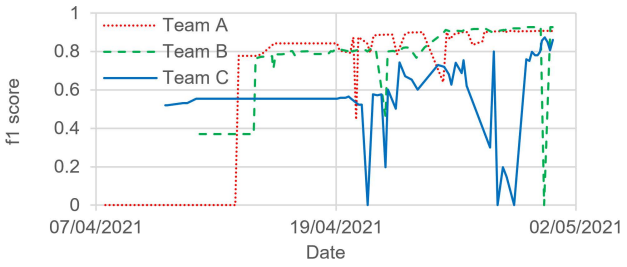

Figure 7: f1 score over time at the SIGMOD contest. The evolution of the f1 score on dataset D4 of three of the top five teams at the SIGMOD contest.

performance. Thus, the matching task had an overall trial-and-error character, which indicates that dataset D4 seems to be challenging even for matching specialists. Furthermore, Frost'sexploration features might reveal starting points for identifying the error of reasoning in taking new assumption for changing configuration. Thereby, Frost essentially foster business efficiency in configuring matching solutions. In conclusion, effort diagrams are beneficial in a variety of ways: They help users track the cost spent for optimizing matching solutions, to detect time points when larger configuration changes are necessary or additional effort might be wasted, prefigure when result exploration should be applied, and give insights about the difficulty of a dataset.

6 INDUSTRY USE CASES

Today's technologies offer a variety of approaches for data matching, targeted at a variety of use-cases. Finding a well-performing matching solution is both crucial and difficult for companies facing poor data quality, resulting in non-transparent and suboptimal decisions. Therefore, there is an industry-wide need for more transparency for data matching solutions. In response, within SAP Frost has the goal to standardize the comparison and evaluation process for its various data matching solutions across departments. Throughout this project, we received constant feedback from several teams within SAP working on future matching solutions.

In practice, matching solutions must be precisely adapted and optimized for a given use case. For SAP customers and also for internal teams, this process is crucial to achieve maximum performance for their matching needs. With the help of Frost's implementation Snowman, they were able to further optimize their workflow and receive helpful insights more quickly than with traditional, use-case specific evaluation tools. Especially Snowman's ability to contrast multiple runs of a given matching solution against a ground truth allowed the teams to more quickly identify blind spots.

Beyond SAP, we were able to work with a global company on improving their in-house data matching solutions for business partner data. With the help of Snowman, the company was able to evaluate alternative solutions to replace their current system in an interactive way. Thereby, information that would otherwise be only available to IT became directly accessible to data-matching stakeholders and domain experts:

- Snowman simplifies this process by providing a standardized way to compare matching solutions using hard- and soft KPIs.
- Additionally, revealing strengths and weaknesses became easier as Snowman conveniently displayed tuples found by both matching solutions, only one of them, or neither.
- Snowman's support for a variety of output formats makes this process easy and enables even persons with limited tech skills to gain meaningful insights.

7 CONCLUSION AND OUTLOOK

We introduced Frost, a benchmark platform for data matching solutions. Besides the traditional benchmark evaluation for result quality, it offers a dimension for expenditures as well as techniques to systematically explore and understand data matching results. We examined how Frost can benefit both organizations in their buying decision and developers in improving their matching solutions. Finally, we presented Snowman as a reference implementation for Frost and evaluated the results of this year's top teams from the SIGMOD programming contest with it. Although we consider Frost and Snowman a significant step in the direction of a standardized and comprehensive benchmark platform for entity resolution systems, our long-term goal is to advance Frost even further along several lines of research:

- **Compatibility with non-relational data**: Data matching is relevant beyond tabular data. Thus, Frost needs support for non-relational data models, such as XML or JSON.
- **Selecting benchmark datasets**: As discussed in Section 3.1.3, it is difficult to find representative benchmark datasets for a real-world matching task. A suitability score based on profiling metrics would be an important contribution towards the search for suitable benchmark datasets.
- **Categorizing errors**: The ability to categorize the errors of a matching solution helps to more easily find structural deficiencies. For example, a matching solution could be especially weak in the handling of typos.
- **Recommending matching solutions**: A long-term goal might be to gather matching solutions, benchmark datasets, and evaluation results in a central repository. To assist organizations with real-world matching tasks, Frost could use this information to automatically determine promising matching solutions.

REFERENCES

[1] Patricia C. Arocena, Boris Glavic, Giansalvatore Mecca, Renée J. Miller, Paolo Papotti, and Donatello Santoro. 2015. Messing Up with BART: Error Generation for Evaluating Data-Cleaning Algorithms. *PVLDB* 9, 2 (2015), 36–47. https://doi.org/10.14778/2850578.2850579

[2] Tobias Bachteler and Jörg Reiher. 2012. *TDGen: A Test Data Generator for Evaluating Record Linkage Methods.* Technical Report wp-grlc-2012-01. German Record Linkage Center.

[3] Matt Barnes. 2015. A Practioner's Guide to Evaluating Entity Resolution Results. *Computing Research Repository (CoRR)* (2015). http://arxiv.org/abs/1509.04238

[4] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: a generic approach to entity resolution. *VLDB Journal* 18, 1 (2009), 255–276. https://doi.org/10.1007/s00778-008-0098-x

[5] Jens Bleiholder and Felix Naumann. 2008. Data fusion. *Comput. Surveys* 41, 1 (2008), 1:1–1:41. https://doi.org/10.1145/1456650.1456651

[6] Prodromos D Chatzoglou and Linda A Macaulay. 1997. The importance of human factors in planning the requirements capture stage of a project. *International Journal of Project Management* 15, 1 (1997), 39–53. https://doi.org/10.1016/S0263-7863(96)00038-5

[7] Surajit Chaudhuri, Venkatesh Ganti, and Rajeev Motwani. 2005. Robust Identification of Fuzzy Duplicates. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 865–876. https://doi.org/10.1109/ICDE.2005.125

[8] Davide Chicco, Niklas Tötsch, and Giuseppe Jurman. 2021. The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Mining* 14, 1 (2021), 13. https://doi.org/10.1186/s13040-021-00244-z

[9] Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer Verlag. https://doi.org/10.1007/978-3-642-31164-2

[10] Peter Christen. 2012. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 24, 9 (2012), 1537–1555. https://doi.org/10.1109/TKDE.2011.127

[11] Peter Christen and Dinusha Vatsalan. 2013. Flexible and Extensible Generation and Corruption of Personal Data. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. 1165–1168. https://doi.org/10.1145/2505515.2507815

[12] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2021. An Overview of End-to-End Entity Resolution for Big Data. *Comput. Surveys* 53, 6 (2021), 127:1–127:42. https://doi.org/10.1145/3418896

[13] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A framework for benchmarking entity-annotation systems. In *Proceedings of the International World Wide Web Conference (WWW)*. 249–260.

[14] Valter Crescenzi, Andrea De Angelis, Donatella Firmani, Maurizio Mazzei, Paolo Merialdo, Federico Piai, and Divesh Srivastava. 2021. Alaska: A Flexible Benchmark for Data Integration Tasks. *Computing Research Repository (CoRR)* (2021). https://arxiv.org/abs/2101.11259

[15] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. 2016. The Magellan Data Repository. https://sites.google.com/site/anhaidgroup/useful-stuff/data.

[16] Ben Davis. 2014. The cost of bad data: stats. https://econsultancy.com/the-cost-of-bad-data-stats.

[17] Dong Deng, Wenbo Tao, Ziawasch Abedjan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Unsupervised String Transformation Learning for Entity Consolidation. *Computing Research Repository (CoRR)* (2018). http://arxiv.org/abs/1709.10436

[18] AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration.* Morgan Kaufmann. http://research.cs.wisc.edu/dibook/

[19] AnHai Doan, Pradap Konda, Paul Suganthan G. C., Yash Govind, Derek Paulsen, Kaushik Chandrasekhar, Philip Martinkus, and Matthew Christie. 2020. Magellan: toward building ecosystems of entity matching solutions. *Commun. ACM* 63, 8 (2020), 83–91.

[20] Uwe Draisbach, Peter Christen, and Felix Naumann. 2020. Transforming Pairwise Duplicates to Entity Clusters for High-quality Duplicate Detection. *Journal on Data and Information Quality (JDIQ)* 12, 1 (2020), 3:1–3:30. https://doi.org/10.1145/3352591

[21] Uwe Draisbach and Felix Naumann. 2010. DuDe: The duplicate detection toolkit. In *Proceedings of the International Workshop on Quality in Databases (QDB)*. Singapore.

[22] Uwe Draisbach and Felix Naumann. 2013. On choosing thresholds for duplicate detection. In *The International Conference on Information Quality (ICIQ)*.

[23] Byron A Ellis. 2007. Life cycle cost. In *International Conference of Maintenance Societies*. Maintenance Engineering Society of Australia, 1–8.

[24] Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. 2014. NADEEF/ER: generic and interactive entity resolution. In *International Conference on Management of Data (SIGMOD)*. ACM, 1071–1074. https://doi.org/10.1145/2588555.2594511

[25] Donatella Firmani, Giovanni Simonini, Andrea De Angelis, and Luca Zecchini. 2021. ACM SIGMOD 2021 Programming Contest. https://dbgroup.ing.unimore.it/sigmod21contest/ Last accessed on 2022-06-10.

[26] Edward B Fowlkes and Colin L Mallows. 1983. A method for comparing two hierarchical clusterings. *Journal of the American statistical association* 78, 383 (1983), 553–569. https://doi.org/10.2307/2288117

[27] Brent Fulgham and Isaac Gouy. 2022. The Computer Language 22.05 Benchmarks Game. https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/javascript.html Last accessed on 2022-06-10.

[28] Luzia Gonçalves, Ana Subtil, and M Rosário Oliveira P d Bermudez. 2014. ROC curve estimation: An overview. *REVSTAT* 12, 1 (2014), 1–20.

[29] Martin Graf, Lukas Laskowski, Florian Papsdorf, Florian Sold, Roland Gremmelspacher, Felix Naumann, and Fabian Panse. 2022. Frost: A Platform for Benchmarking and Exploring Data Matching Results. (2022). arXiv:2107.10590 [cs.DB]

[30] David J. Hand, Peter Christen, and Nishadi Kirielle. 2021. F*: an interpretable transformation of the F-measure. *Mach. Learn.* 110, 3 (2021), 451–456. https://doi.org/10.1007/s10994-021-05964-1

[31] Oktie Hassanzadeh, Fei Chiang, Renée J. Miller, and Hyun Chul Lee. 2009. Framework for Evaluating Clustering Algorithms in Duplicate Detection. *PVLDB* 2, 1 (2009), 1282–1293. https://doi.org/10.14778/1687627.1687771

[32] Alireza Heidari, George Michalopoulos, Shrinu Kushagra, Ihab F. Ilyas, and Theodoros Rekatsinas. 2020. Record fusion: A learning approach. *Computing Research Repository (CoRR)* (2020). https://arxiv.org/abs/2006.10208

[33] Arvid Heise, Gjergji Kasneci, and Felix Naumann. 2014. Estimating the Number and Sizes of Fuzzy-Duplicate Clusters. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. Association for Computing Machinery, 959–968. https://doi.org/10.1145/2661829.2661885

[34] Al Koudous Idrissou, Frank van Harmelen, and Peter van den Besselaar. 2018. Network Metrics for Assessing the Quality of Entity Resolution Between Multiple Datasets. In *Knowledge Engineering and Knowledge Management (EKAW) (Lecture Notes in Computer Science)*, Vol. 11313. Springer Verlag, 147–162. https://doi.org/10.1007/978-3-030-03667-6_10

[35] Ekaterini Ioannou and Yannis Velegrakis. 2019. EMBench^{++}: Data for a thorough benchmarking of matching-related methods. *Semantic Web* 10, 2 (2019), 435–450. https://doi.org/10.3233/SW-180331

[36] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *PVLDB* 9, 12 (2016), 1197–1208. https://doi.org/10.14778/2994509.2994535

[37] Hanna Köpcke and Erhard Rahm. 2010. Frameworks for entity matching: A comparison. *Data and Knowledge Engineering (DKE)* 69, 2 (2010), 197–210. https://doi.org/10.1016/j.datak.2009.10.003

[38] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2009. Comparative evaluation of entity resolution approaches with FEVER. *PVLDB* 2, 2 (2009), 1574–1577.

[39] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *PVLDB* 3, 1 (2010), 484–493. https://doi.org/10.14778/1920841.1920904

[40] Ioannis K. Koumarelas, Lan Jiang, and Felix Naumann. 2020. Data Preparation for Duplicate Detection. *Journal on Data and Information Quality (JDIQ)* 12, 3 (2020), 15:1–15:24. https://dl.acm.org/doi/10.1145/3377878

[41] Marina Meila. 2003. Comparing Clusterings by the Variation of Information. In *Annual Conference on Computational Learning Theory and Kernel Workshop (Lecture Notes in Computer Science)*, Vol. 2777. Springer Verlag, Washington, DC, USA, 173–187. https://doi.org/10.1007/978-3-540-45167-9_14

[42] David Menestrina, Steven Whang, and Hector Garcia-Molina. 2010. Evaluating Entity Resolution Results. *PVLDB* 3, 1 (2010), 208–219. https://doi.org/10.14778/1920841.1920871

[43] Charini Nanayakkara, Peter Christen, Thilina Ranbaduge, and Eilidh Garrett. 2019. Evaluation measure for group-based record linkage. *International Journal of Population Data Science (IJPDS)* 4, 1 (2019). https://doi.org/10.23889/ijpds.v4i1.1127

[44] Fabian Panse and Felix Naumann. 2021. Evaluation of Duplicate Detection Algorithms: From Quality Measures to Test Data Generation. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 2373–2376. https://doi.org/10.1109/ICDE51399.2021.00269

[45] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. *The Four Generations of Entity Resolution.* Morgan & Claypool Publishers.

[46] George Papadakis, Georgios M. Mandilaras, Luca Gagliardelli, Giovanni Simonini, Emmanouil Thanos, George Giannakopoulos, Sonia Bergamaschi, Themis Palpanas, and Manolis Koubarakis. 2020. Three-dimensional Entity Resolution with JedAI. *Information Systems (IS)* 93 (2020), 101565.

[47] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *Comput. Surveys* 53, 2 (2020), 31:1–31:42. https://doi.org/10.1145/3377455

[48] Petar Petrovski and Christian Bizer. 2020. Learning expressive linkage rules from sparse data. *Semantic Web* 11, 3 (2020), 549–567. https://doi.org/10.3233/SW-190356

[49] Anna Primpeli and Christian Bizer. 2020. Profiling Entity Matching Benchmark Tasks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. Association for Computing Machinery, 3101–3108. https://doi.org/10.1145/3340531.3412781

[50] Kun Qian, Lucian Popa, and Prithviraj Sen. 2019. SystemER: A Human-in-the-Loop System for Explainable Entity Resolution. *PVLDB* 12, 12 (2019), 1794–1797. https://doi.org/10.14778/3352063.3352068

[51] Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. 2018. Gerbil – Benchmarking named entity recognition and linking consistently. *Semantic Web* 9, 5 (2018), 605–625.

[52] M. Ali Rostami, Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2018. Interactive Visualization of Large Similarity Graphs and Entity Resolution Clusters. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. OpenProceedings.org, 690–693. https://doi.org/10.5441/002/edbt.2018.86

[53] Frank Ruskey, Carla D Savage, and Stan Wagon. 2006. The search for simple symmetric Venn diagrams. *Notices of the AMS* 53, 11 (2006), 1304–1311.

[54] Sunita Sarawagi and Anuradha Bhamidipaty. 2002. Interactive deduplication using active learning. In *Proceedings of the International Conference on Knowledge discovery and data mining (SIGKDD)*. Association for Computing Machinery, 269–278. https://doi.org/10.1145/775047.775087

[55] Tzanina Saveta. 2014. *SPIMBench: A Scalable, Schema-Aware Instance Matching Benchmark for the Semantic Publishing Domain*. Master's thesis. University of Crete.

[56] Tzanina Saveta, Evangelia Daskalaki, Giorgos Flouris, Irini Fundulaki, and Axel-Cyrille Ngonga Ngomo. 2015. LANCE: A Generic Benchmark Generator for Linked Data. In *Proceedings of the International Semantic Web Conference (ISWC)*. http://ceur-ws.org/Vol-1486/paper_43.pdf

[57] Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[58] Tobias Vogel, Arvid Heise, Uwe Draisbach, Dustin Lange, and Felix Naumann. 2014. Reach for gold: An annealing standard to evaluate duplicate detection results. *Journal on Data and Information Quality (JDIQ)* 5, 1-2 (2014), 5:1–5:25. https://doi.org/10.1145/2629687

[59] Melanie Weis, Felix Naumann, and Franziska Brosy. 2006. A duplicate detection benchmark for XML (and relational) data. In *Proceedings of the International Workshop on Information Quality for Information Systems (IQIS)*. Association for Computing Machinery.

[60] Renzhi Wu, Prem Sakala, Peng Li, Xu Chu, and Yeye He. 2021. Demonstration of Panda: A Weakly Supervised Entity Matching System. *PVLDB* 14, 12 (2021), 2735–2738. https://doi.org/10.14778/3476311.3476332