Explaining Entity Resolution Predictions : Where are we and What needs to be done?

Saravanan Thirumuruganathan, Mourad Ouzzani, Nan Tang
Qatar Computing Research Institute, HBKU, Qatar
{sthirumuruganathan, mouzzani, ntang}@hbku.edu.qa

ABSTRACT

Entity resolution (ER) seeks to identify the set of tuples in a dataset that refer to the same real-world entity. It is one of the fundamental and well studied problems in data integration with applications in diverse domains such as banking, insurance, e-commerce, and so on. Machine Learning and Deep Learning based methods provide the state-of-the-art results. For practitioners, it is often challenging to understand why the classifier made a particular prediction. While there has been extensive work in the ML community on explaining classifier predictions, we found that a direct application of those techniques is not appropriate for ER. There is a huge gap between the needs of lay ER practitioners and the explanation community. In this paper, we provide a comprehensive taxonomy of these challenges, discuss research opportunities and propose preliminary solutions.

ACM Reference Format:
Saravanan Thirumuruganathan, Mourad Ouzzani, Nan Tang . 2019. Explaining Entity Resolution Predictions : Where are we and What needs to be done?. In *HILDA '19: 2019 Workshop on Human-In-the-Loop Data Analytics, July 05, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/xxxx/xxxxx.xxxx

1 INTRODUCTION

Entity Resolution (ER) (*a.k.a.* record linkage or deduplication) is a fundamental problem in data integration. Given a dataset, ER seeks to identify the set of tuple pairs that refer to the same real-world entity. This has a number of applications in domains as diverse as banking, insurance, e-commerce, and many more. An e-commerce website would like to know if two products from different suppliers are the same so that they can be listed in the same product page. Two banks merging would want to identify and reconcile common customers.

Traditional approaches for ER operate in two phases: blocking and matching. Given a dataset D, blocking methods generate a candidate set $C \subseteq D \times D$ that excludes tuple pairs which are unlikely to match. The matching phase typically uses a classifier for predicting if a given tuple pair is a duplicate. Due to its importance, ER has been extensively studied with state-of-the-art results provided by Machine Learning (ML) and Deep Learning (DL) based solutions such as DeepER [5] and DeepMatcher [17]. In this paper, we focus primarily on explaining the predictions of an ER matcher.

Explanations for ML. Current methods for explaining predictions of an ML classifier can be broadly categorized [3, 9, 14] as local or global. Local explanations seek to provide why the model predicted a label for a specific instance. In the context of ER, a local explanation would explain why a tuple pair was predicted to be a duplicate. Common approaches include identifying a ranked list of features that contribute towards the prediction as a duplicate (such as LIME [19]) or as a rule that holds true in the vicinity of the tuple-pair with high precision (such as Anchor [20]). Global explanations seek to concisely summarize the entire model for the user. For example, the global explanation could summarize an ML model for ER as a set of if-then rules.

Explaining ML Classifiers for ER. While there has been extensive work on explanations for ML models, they are often insufficient for explaining matching predictions to an ER practitioner. We now briefly enumerate the potential issues and discuss the key dimensions along with preliminary solutions in Section 2.

1. Application (ER) Specific Explanations. Most of the prior explanation methods are often agnostic to the task and classifier. While agnosticism to classifier is indeed desirable, one can generate much better explanations by tailoring it to the task at hand. This is especially relevant for ER that has a number of distinctive properties. ER often exhibits a skewed distribution of labels where non-duplicates often outnumber duplicates by a factor of 100 or more. Often, similarity/distance values between two aligned attributes (such as customer name in two relations) are used as features.

HILDA '19, July 05, 2019, Amsterdam, Netherlands

https://doi.org/xxxx/xxxxx.xxxx

Generating explanations using popularly used features/similarity functions for ER such as Jaro-Winkler, Levenshtein, Monge-Elkan would not be easily understandable even for data scientists, let alone a domain expert.

2. Explanation Templates. Explanations for structured predictions almost exclusively rely on feature importance and if-then rules. However, it is desirable to explore other forms of explanations. Common examples include:

(i) Explanations using tuple pairs whereby we identify a small number of very relevant tuple pairs from the training dataset that could be used to explain the prediction of an instance.

(ii) Of course, relevant tuple pairs might not always exist which leads us to explanations through "sufficient input subsets" of features [1] whose observed values are sufficient for the same decision to be reached. By showing a small collection of tuple pairs that share these feature subsets, the user could understand the prediction.

(iii) Counterfactual explanations that describe the smallest change that could switch a tuple pair from duplicate to non-duplicate, and vice versa.

3. Explanations for Debugging. Current explanation approaches focus on two extremes: local or global. Often, the domain expert is interested in understanding other phenomena that do not cleanly fall in this spectrum. For example, a domain expert would be interested in *failure analysis* seeking to understand how/why the classifier fails in certain circumstances. This could help in *debugging* the classifier resulting in better features, training dataset and so on. Similarly, a domain expert would be interested in *differential analysis* where the scientist seeks to compare and contrast two models for the same ER problem through the lens of explanations.

4. Explanation Summarization. Local explanation methods often provide justification for the prediction of an individual tuple pair. However, the dataset could contain millions of such instances and one cannot expect a domain expert to go through each of them. Global methods are an alternative but do not work in many scenarios. It is important to identify mechanisms to generate "summaries" (such as a report for a ER practitioner) that concisely details the model predictions. This could either aggregate local explanations of all relevant tuple pairs or use other novel mechanisms such as identifying representative tuple pairs.

5. Explanation Scalability. Current explanation methods are often slow and could take seconds for individual predictions. Running the model on millions of predictions could take a lot of time even if one uses distributed methods. A large number of explanation methods share many commonalities under the hood such as sample generation and gradient computation. By using database techniques such as materialization and reuse, it is possible to speed up these approaches by orders of magnitude.

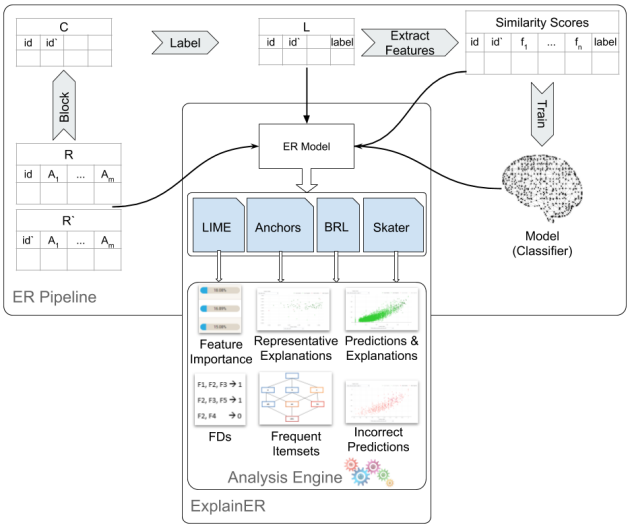

Figure 1: ER Pipeline and Architecture of ExplainER – Our preliminary version for Explanations for ER.

2 BETTER EXPLANATIONS FOR ER

In this section, we discuss key dimensions for improving explanations for ER, identify interesting research challenges and propose preliminary solutions. Our ultimate goal is to design a Python package for explaining ER predictions. We demonstrate in [4] ExplainER, a preliminary version that solves a handful of these challenges by leveraging existing local and global explanation methods. The architecture of ExplainER is shown in Figure 1.

2.1 Explanations for Skewed Datasets

Typically in ER, non-duplicates outnumber duplicates by a factor of 100 even after blocking is performed. This label skew causes one of the fundamental challenges in adapting existing explanation methods for ML classifiers to ER. A naive invocation often produces meaningless results. Concretely, we discuss how this issue affects two popular approaches: LIME [19] and Anchors [20].

LIME [19] is a model agnostic explanation system that generates explanations using features ranked based on their importance. Suppose we are given a tuple pair t that a classifier labeled as a duplicate. LIME might identify that features f_i, f_j, f_k contributed with weights w_i, w_j, w_k towards a

prediction of the pair being a duplicate while features f_l, f_m contributes towards the prediction of non-duplicate. It makes numerous random perturbations of the feature vector for t, assigns a notion of similarity for each perturbation based on its distance to t, runs the classifier on each perturbation and selecting the K best features using Lasso.

This proposed approach works well for balanced datasets where invoking the classifier on perturbations provides a good mix of positive and negative labels. For ER, this approach would only work for tuple pairs in the vicinity of the classifier's global decision boundary such as duplicates where minor perturbations could result in a prediction of non-duplicate. However, for non-duplicates which account for the vast majority of tuple pairs this does not work. Minor perturbations only results in predictions of more non-duplicates - so LIME would misidentify the top-K features and produce an incorrect explanation.

Improving LIME Explanations for ER. There are many ways to improve LIME explanations. One conservative approach used by ExplainER [4] is to only use LIME explanations on duplicates or tuple pairs for whom minor perturbations produce a mix of predictions of duplicates and non-duplicates. Another approach would be to progressively try larger perturbations till a good mix of predictions is obtained. This has to be done carefully and requires two changes. First, we must modify the kernel function such that the weight between the original tuple pair and the perturbations is based on both the distance between them and the distance from the nearest decision boundary manifold. Second, when one increases the perturbation radius, it is possible that one might straddle multiple decision boundary manifolds. In this case, we must exhaustively enumerate each of these manifolds, generate LIME explanations separately for each of them and aggregate it intelligently.

Anchors [20] are model agnostic explanations that explain ER predictions using high *precision* rules (dubbed anchors). These anchors represent sufficient conditions with high precision in the vicinity of the tuple pair being explained. We can immediately see the issue in applying anchors for ER. Given the preponderance of non-duplicates, one could come up with a trivial anchor that blindly declares every tuple pair to be a non-duplicate. For most ER datasets, this anchor will have a precision of 0.99 or higher!

Improving Anchors for ER. While this approach works for balanced datasets, it is not appropriate for ER. For imbalanced datasets, one must use alternative metrics such as F-score (the harmonic mean between precision and recall). The key issue is that Anchors relies on precision metric for identifying useful rules for explanations. However, modifying anchors to identify rules with high F-score is very challenging. A conservative approach is to limit the explanations to duplicates or those near the decision boundary. In this scenario, recall is either well-defined or non-zero allowing one to estimate F-score. While precision has a straightforward statistical estimator that could be computed from the samples, designing one for recall is much trickier. Furthermore, adapting recently developed estimators for recall of ER (or F-score) for the purpose of explanations is either expensive [6] or not straightforward [15].

2.2 Templates for ER Explanations

Almost all of the explanation methods for tabular data are based on feature importances or if-then rules. Feature importance based methods such as LIME are often challenging to interpret for a domain expert. Part of the reason is that the features themselves are harder to explain with an alphabet soup of names such as Jaro-Winkler, Levenshtein, and Monge-Elkan. A typical end user of ER matcher is often a domain expert from domains such as banking, insurance, e-commerce, etc. This requires innovation in other types of explanations. In this subsection, we identify three promising alternatives and describe simple algorithms for each.

Explanation by Examples. The simplest explanation for explaining the prediction for a tuple pair t could be very similar tuple pairs from the training dataset. Since the domain expert would have hand labeled these tuple pairs, it behoves that we use it as part of our explanation. Note that simply performing nearest neighbor in the feature space is often insufficient. This is due to the fact that features for ER are often similarity based. So two different tuple pairs could both have identical similarity values whereas the underlying raw tuple pairs might look very different. A simple solution is to use a sophisticated function for identifying the nearest neighbor.

The function must rely on four factors whose weights are set as appropriate. The nearest neighbor must have (i) very similar feature values and thereby (ii) similar explanation in terms of feature importances (one does not follow the other!). Furthermore, the nearest neighbor (iii) must have the same class label either duplicate or non-duplicate. Finally, (iv) the aligned component tuples must have similar feature vectors with each other. Suppose we wish to identify duplicate tuples between two relations D and D'. Let the two tuple pairs be $t_i = (a, b)$ and $t_j = (c, d)$ where $\{a, c\} \in D$ and $\{b, d\} \in D'$. In order for (c, d) to be a good neighbor, the final desiderata (d) requires that the feature vector between (a, d) and (c, b) be very similar. All these constraints will ensure that the chosen neighbors are similar in terms of the raw values, feature vectors, explanations and the predicted class label. Of course,

this constraint is very restrictive but provides a good set of candidates. One could always relax this requirement by modifying the weight assigned to some of these component terms.

Explanation by Sufficient Subsets. It is possible that explanations by examples could fail as it is restricted to the training dataset. One way to generalize is to identify the neighbor from both training and testing datasets. Another way is to look for alternate explanation templates. Sufficient input subsets are one such mechanism that is closely related to Anchors. This approach identifies a minimal subset of features such that their observed values are "sufficient" [1] for the classifier to make the same decision for any instance. One can view this approach as a way to generate a small group of tuple pairs that all share the same value for a subset of features and produce the same prediction. For relations with mostly categorical and numerical attributes, this approach could be readily implemented. The key challenge is handling attributes with textual content where a different semantics for sufficiency must be defined. A simplistic approach would be to remove all stop words and focus on the words with large tf-idf values. However, more research is needed to flesh out sufficient input subsets for ER.

Explanation by Counterfactuals. Intuitively, a counter factual explanation tries to find the smallest change to the feature values such that the prediction is changed from duplicate to non-duplicate or vice versa. For example, the model could say that products X and Y are not duplicates - however, they would have been if their weights were the same. Counter factual explanations are often human friendly [16] and are an effective tool for debugging classifiers. However, there are two challenges. First, it is possible that there are multiple possible counterfactuals for the same tuple pair [16]. However, identifying the "best" one is quite challenging. The simplest solution is to simply list all of these and let the human make the choice. Alternatively, we could limit the number of counterfactuals each attribute participates in which bounds the number of explanations to the number of attributes. The second challenge is that perturbing feature values realistically for a tuple pair is often challenging. Suppose that two products with weight 1 KG are considered to be a duplicate but could be made a non-duplicate with a different weight. Also suppose that the acceptable weight threshold for the classifier is 0.5 KG (*i.e.,* if the weights are off by more than 0.5 KG, it will be considered as different). A counterfactual that sets the weight of one product to 1.6 is preferable to one that sets it to ∞. Systematic approaches could be easily defined for categorical and numerical values. We can generate counterfactuals by cycling through the domain value of a categorical attribute while using a binary search like idea for numerical attributes. However, automatically finding the minimal perturbation for textual data is quite challenging and requires further research.

2.3 Explanations for Debugging

This is one of the areas where there has been extensive work from the HILDA community [8, 10, 21, 22] and the DB community at large [2, 18]. We have made some progress in extending them to the ER problem. Nevertheless, there are still a number of important problems relevant for ER waiting to be investigated.

Debugging Training Dataset. Consider a tuple pair t that was predicted as a non-duplicate. While using methods such as feature importance or rules could provide some intuition as to why, it does not provide a mechanism to "fix" it if it was wrongly predicted. Often, an incorrect prediction is an artifact of an incomplete training dataset. Some of the common problems in the training dataset include: (i) incorrect labels and (ii) not enough data similar to the tuple t that was mis-predicted. We consider three important sub-problems and provide preliminary solutions that are very effective albeit inefficient.

P1: Are the most important tuple pairs correctly labeled? Suppose that the domain expert has time to validate the labels of K tuple pairs. Then, we need to identify the K tuple pairs that have the most impact on the model and validate them with the expert. Intuitively, we remove the tuple pair from training dataset, retrain the model and measure the impact on the prediction. We use Cook's distance [16] parameterized by prediction outcome to measure the impact of deleting the instance on the model parameters. While deleting individual tuples provides the best results, a natural approach to speed it up is by clustering the tuple pairs and computing Cook's distance for each cluster.

P2: Which training instance(s) caused my tuple pair to be mis-predicted? This is challenging to do effectively for all possible ML models. Heuristically, we use knowledge distillation [7] to compress a blackbox classifier model into a smaller more interpretable model. Specifically, we use a depth bounded decision tree for this purpose where the depth is inversely proportional to the number of tuple pairs the domain expert is willing to examine. Intuitively, the key idea is to partition the data using the decision tree such that if a tuple pair is misidentified, we only need to examine all tuple pairs falling under the path taken by that instance. It is also possible to use other models for partitioning such as [10].

P3: Are there sufficient coverage of training instances? We construct OLAP cubes for both the training and test dataset independently. Then we identify the top-K maximal and non-overlapping cuboids where the fraction of tuple pairs

is very different in the training and test cubes. While simplistic, this approach can easily find common problematic instances such as: (i) lack of relevant training data: identified when a cuboid in the test dataset has larger number of tuples than its corresponding cuboid in the training dataset; This implies that there are insufficient numbers of instances in the training data corresponding to tuple pairs from the testing cuboid, and (ii) covariate shift occurs when there is a difference in the distribution of training and testing datasets. By comparing corresponding cuboids with large differences, a domain expert can validate if this discrepancy is acceptable or expected.

Debugging Classifiers. There has been some interesting work such as [2] that helps users in tracing model performance issues by identifying subsets of data where the model performs poorly. This method automatically identifies subsets of data that are easy to specify, have non-negligible impact on ER mode's F-score (dubbed accuracy in [2]) and carefully avoid false positives.

Comparing ER Models. Finally, it is important to investigate other model metrics. For example, a domain expert would be interested in *differential analysis* where one seeks to compare and contrast two models for the same ER problem through the lens of explanations. Without loss of generality, let model M_1 be better than model M_2. The key idea is to differentiate progression (instances where M_2 was wrong and M_1 is correct) and regression (where M_2 was right but M_1 was wrong). Often, the improvement is not uniform. One could extend [2] to identify the subsets of data where M_1 dramatically outperforms M_2 that provides an easy to understand explanation as to why model M_1 is better. This is often more meaningful to the domain expert than abstract performance metrics such as precision, recall or F-score.

2.4 Explanation Summarization

In this subsection, we describe two ideas incorporated in ExplainER [4] that leverages prior work. However, there are a number of interesting research problems to be tackled. The key issue is that prior explanation methods operate at two extremes: local or global. While local models provide granular information, there are simply too many tuple pairs to inspect. Global models often suffer from loss of fidelity with original ML model or produces too many rules – often in the range 30-50 for benchmark ER datasets. Ideally, we wish to generate a "report" that succinctly summarizes the predictions such that the expert can validate it.

Representative Tuple Pairs. A simple approach is to identify K tuple pairs that when presented to the domain expert, provides a representative overview of the entire model. These must be chosen judiciously by weighing the local/-global importance of features. Interestingly, for many feature importance based methods, the function for computing representativeness is submodular. Hence, it is possible to choose tuple pairs that cover major components (*i.e.,* feature importances) and are not redundant. We use the submodular pick algorithm from [19] for such purposes.

MDL Summarization. In many domains such as banks or e-commerce, individual attributes are often associated with an OLAP hierarchy. It is possible to generate concise summaries by using these OLAP hierarchies. Suppose we are given the top-K most importance features for each tuple pair. This can be considered as a binary matrix where rows correspond to tuples while columns correspond to features. A cell for tuple pair t and attribute A has a value of 1 if this A is in the top-K features for t. Given this setting, one can identify a small set of "regions" defined by the OLAP hierarchy that covers the entries with value of 1. These regions are human understandable as they are defined using OLAP hierarchies and provide a bird's eye view of how explanations and tuple pairs interact. For this purpose, we implemented the algorithm from [12] in ExplainER.

2.5 Scaling Up Explanation Algorithms

The current generation of local and global explanation algorithms for tabular data are not very scalable. Local methods such as LIME or Anchors could take as much as few seconds to generate explanations. Similarly, global methods such as Interpretable decision sets [11] or Bayesian Rule Lists (BRL) [13] could require as much as few hours to obtain a concise set of rules. Given that ER problems operate in the realm of quadratic complexity (recall blocking gives $C \subseteq D \times D$), this could be a formidable challenge. Naive parallelization or distributed computing is often insufficient and requires novel algorithmic breakthroughs. We now briefly describe some preliminary approaches to speedup popular explanation methods.

Scaling up LIME. Recall that LIME takes in a feature vector F for tuple pair t, makes many perturbations of F, applies the classifier on each perturbation and applies K-Lasso to identify the most important features. Suppose we have another tuple pair t' with feature vector F' that is very similar to t. Intuitively, it must be possible to reuse many of the perturbations of F for F'. This reduces the expensive steps of generating perturbations and then applying the classifier on them. Of course, this notion of materialization and reuse of perturbations would not be surprising to a database researcher. While intuitive, this approach of retrieving and reusing appropriate materialized perturbations requires solving numerous practical issues.

Scaling up BRL. BRL [13] is a promising approach that produces rule lists that are accurate and concise. However, the flexibility is obtained at the expense of computation time. The key idea to speeding up global methods is to come up with an appropriate objective function for global models that could be approximated. In the context of BRL, it already has an elegant posterior distribution over rule lists that allows principled approximation which dramatically prunes the search space [23]. Extending this idea to other global methods for ER is an intriguing research challenge.

Impact of Blocking. An orthogonal approach for scalability is to simply reduce the number of instances on which the explanation algorithms are run on. Blocking is a natural avenue for this as it reduces the set of candidate pairs to be evaluated by the classifier. An intriguing question is how does blocking impact explanations? Similarly, is it possible to devise "explanation aware" blocking where tuple pairs in the same block would most likely have similar explanations. In this case, one need to only apply explanation algorithms on a representative tuple pair from each block.

3 CONCLUSION

In this paper, we argued that there is a large gap and a golden opportunity between current research on explainability and the practitioners of entity resolution. We identified a taxonomy of interesting research problems and proposed preliminary solutions. While our discussion was anchored on ER, many of the problems could naturally be extended to other data integration and database problems. Our proposed approaches are a promising start but more needs to be done. Given the oncoming tide of ML in databases, it is important that the database community investigate these research problems and smoothen such transition for practitioners.

REFERENCES

[1] B. Carter, J. Mueller, S. Jain, and D. Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. *AISTATS*, 2019.

[2] Y. Chung, T. Kraska, N. Polyzotis, and S. E. Whang. Slice finder: Automated data sclicing for model validation. *arXiv preprint arXiv:1807.06068*, 2018.

[3] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[4] A. Ebaid, S. Thirumuruganathan, A. Elmagarmid, M. Ouzzani, and W. Aref. Explainer – entity resolution explanations. In *ICDE Demo*. IEEE, 2019.

[5] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[6] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 601–612. ACM, 2014.

[7] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[8] M. Kahng, D. Fang, and D. H. P. Chau. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 1. ACM, 2016.

[9] B. Kim and F. Doshi-Velez. Interpretable machine learning: the fuss, the concrete and the questions. *ICML Tutorial on interpretable machine learning*, 2017.

[10] S. Krishnan and E. Wu. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 4. ACM, 2017.

[11] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684. ACM, 2016.

[12] L. V. Lakshmanan, R. T. Ng, C. X. Wang, X. Zhou, and T. J. Johnson. The generalized mdl approach for summarization. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 766–777. VLDB Endowment, 2002.

[13] B. Letham, C. Rudin, T. H. McCormick, D. Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[14] Z. C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[15] N. G. Marchant and B. I. Rubinstein. In search of an entity resolution oasis: optimal asymptotic sequential importance sampling. *Proceedings of the VLDB Endowment*, 10(11):1322–1333, 2017.

[16] C. Molnar et al. Interpretable machine learning: A guide for making black box models explainable. *Christoph Molnar, Leanpub*, 2018.

[17] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM, 2018.

[18] F. Panahi, W. Wu, A. Doan, and J. F. Naughton. Towards interactive debugging of rule-based entity matching. In *EDBT*, pages 354–365, 2017.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[20] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] P. Tamagnini, J. Krause, A. Dasgupta, and E. Bertini. Interpreting black-box classifiers using instance-level visual explanations. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 6. ACM, 2017.

[22] P. Varma, D. Iter, C. De Sa, and C. Ré. Flipper: A systematic approach to debugging training sets. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 5. ACM, 2017.

[23] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3921–3930. JMLR. org, 2017.