

Using Descriptions for Explaining Entity Matches

(Discussion Paper)

Matteo Paganelli¹, Paolo Sottovia², Antonio Maccioni³, Matteo Interlandi⁴ and Francesco Guerra¹

¹DIEF - University of Modena and Reggio Emilia, Modena, Italy

²Huawei Research. Munich, Germany

³Independent Researcher, USA

⁴Microsoft Research, Seattle, USA

Abstract

Finding entity matches in large datasets is currently one of the most attractive research challenges. The recent interest of the research community towards Machine and Deep Learning techniques has led to the development of many and reliable approaches. Nevertheless, these are conceived as black-box tools that identify the matches between the entities provided as input. The lack of explainability of the process hampers its application to real-world scenarios where domain experts need to know and understand the reasons why entities can be considered as match, i.e., they represent the same real-world entity. In this paper, we show how *data descriptions*—a set of compact, readable and insightful formulas of boolean predicates—can be used to guide domain experts in understanding and evaluating the results of entity matching processes.

Keywords

data explanation, data exploration, data profiling, outliers

1. Introduction

Entity Matching (EM) is a long-lasting problem in the database research community. Recently, approaches based on Machine Learning (ML) and Deep Learning (DL) have been proposed. They conceive EM as a binary classification problem [1] applied on datasets whose records describe pairs of entities. Since the first proposals (see related work in Section 4), they have proved to be very effective. Nevertheless, the application of ML and DL approaches in real scenarios is hampered not only by the need for a significant amount of labeled data for their configuration and training, but also for the lack of explainability of the results they provide.

We have already introduced and showcased (*data*) *descriptions*, i.e. compact, readable and insightful structures formed by *predicates* expressed on the attribute values and able to effectively explain the content of large and complex datasets, in [2, 3]. We have experimentally

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy


✉ matteo.paganelli@unimore.it (M. Paganelli); paolo.sottovia@huawei.com (P. Sottovia); antonio.maccioni@gmail.com (A. Maccioni); matteo.interlandi@microsoft.com (M. Interlandi); francesco.guerra@unimore.it (F. Guerra)

🌐 <http://morespace.unimore.it/francescoguerra/> (F. Guerra)

🆔 0000-0001-8119-895X (M. Paganelli); 0000-0001-7116-9338 (P. Sottovia); 0000-0002-5756-8321 (M. Interlandi); 0000-0001-6864-568X (F. Guerra)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Entity ID	Song Name	Artist Name	Album Name	Genre	
1	I Ai n't Livin' Long Like This	Chris Janson	Take It to the Bank - EP	Country	e_1
2	I Ai n't Livin' Long Like This	Chris Janson	Take It to the Bank (EP)	Country	e_1
3	I Ai n't Livin' Long Like This	Chris Janson	Take It to the Bank - EP	Country	e_1
4	Billy	Keith Urban	Keith Urban In The Ranch	Country	e_2
5	I Ca n't Go There (Acoustic Version)	Kenny Chesney	No Shoes, No Shirt, No Problems	Country	e_3
6	I Ca n't Go There (Acoustic Version)	Kenny Chesney	No Shoes, No Shirt, No Problems	Country	e_3
7	I Ca n't Go There (Acoustic Version)	Kenny Chesney	No Shoes, No Shirt, No Problems	Urban Cowboy	e_3
8	Afire Love	Ed Sheeran	x	Pop	e_4
9	Afire Love	Ed Sheeran	x	Pop	e_4
10	Afire Love	Ed Sheeran	x	Pop	e_4

Table 1

The Entity Matches from the iTunes-Amazon Dataset

demonstrated the effectiveness of *descriptions* in performing many tasks of data exploration. The aim of this paper is to show how they can be useful in another field, making a (large) dataset describing the result of an EM task understandable at a glance by domain expert users.

Running Example. Let us introduce Table 1 containing a sample of entity matches obtained by applying a generic EM approach to a sample of the *iTunes-Amazon* dataset published in the Magellan library¹ and by transitively extending the results through a connected components algorithm (i.e., according to the literature [4], we consider entities as referring to the same real-world entity when the matching elements form a clique). Each record represents a song with a list of characteristics (Entity Id, Song Name, Artist name, Album Name, and Genre). The extra-column on the right shows the result of the application of the EM process, which is able to recognize four distinct songs from the nine in the Table. A domain expert who needs to validate the result of the EM task has to manually inspect the dataset to figure out the reasons for the computed matches. Data descriptions can support the process since they are able to concisely represent any arbitrary set of data tuples.

Example 1. D_1 is a description of the dataset represented in Table 1 obtained conjoining a list of predicates, each characterizing the same attribute of the dataset with the related values. We called M_E the extra-column recording the identifier of the resulting entity from the matching task. In this example, the description covers the entire dataset (all the tuples are represented by D_1) and is built upon only an attribute of the original dataset (i.e., Song_Name). Moreover, D_1 partitions the dataset into four clusters, each one representing a unique entity resulting from the EM process. The description based on the Song_Name attribute is then useful to understand if the EM Model is able to perform a correct identification of entities: for each entity a different song name is displayed thus confirming the correct subdivision into distinct entities of the dataset entries. Other descriptions for the same dataset are possible, built upon other attributes, dividing the datasets into other partitions and using more attributes for each partition (e.g., the dataset can be grouped by Genre and described with the Album Name).

$$D_1 : (M_E \in \{e_1\}) \wedge \text{Song_Name} \in \{\text{I Ai n't Livin' Long Like This}\} \vee \\ (M_E \in \{e_2\}) \wedge \text{Song_Name} \in \{\text{Billy}\} \vee$$

¹<https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md>

$$(M_E \in \{e_3\} \wedge Song_Name \in \{I\ Ca\ n't\ Go\ There\ (Acoustic\ Version)\}) \vee \\ (M_E \in \{e_4\} \wedge Song_Name \in \{Afire\ Love\})$$

Building the Descriptions: A principled approach. The data descriptions are generated according to three main principles. The first states that *it is more explicative to think of a dataset as the composition of different groups of related tuples*. Attributes of a dataset carry different degrees of relevance for users who want to understand the content of datasets and glean insight from their data. Some attributes have intrinsic value because, for example, they can identify entities in a domain (e.g., the attribute `EntityID` in the dataset of Table 1) or because they allow identifying specific features of the entities (e.g., the attributes `Album Name`, `Artist Name` in the dataset). On the other hand, the importance of certain attributes may depend on the users: on their specific interests motivating the dataset exploration and on their knowledge of the domain. Nevertheless, in this case study the idea is to generate data descriptions representing entity matches, and it is therefore natural to partition the dataset around clusters of matching entities. We call *d-formula* the conjunction of predicates describing a partition. A *description* is formally a non-empty disjunctive-normal formula of d-formulas (one for each partition).

The exploration of a dataset as well as the analysis of entity matches can be conducted for different purposes. In some cases, users want an accurate and complete, yet readable, representation of the whole dataset, able to precisely explain the reasons that led to the generation of those clusters of matching entities. In other cases, a general profile of the dataset is enough for the user, who needs only to broadly know why entity possible match in that domain. The description represents, in these latter cases, an overview that ignores infrequent values. Often, instead, there are users who are interested in finding possible mistakes, i.e., entities in the same clusters with values which are not consistent with the ones of other entities in the same cluster. These values can be infrequent values, or outliers, but can also represent possible mistakes. The second driving principle is that *we can accommodate the multi-faceted goals of data explanation by relaxing the concept of coverage in the descriptions*. Our approach allows users to interactively change the coverage of the expected descriptions (intended as the percentage of the total number of tuples that are true for the description), thus making them able to satisfy all possible needs.

Finally, the quality of a description is influenced by the subjectivity of the users and the tasks where the description has to be applied. A prolix description can be suitable for a user who wants to interpret a matching model, but, at the same time, be poor for another one who only needs to know the main features of a type of entity. *The third driving principle is to consider user preferences for qualifying descriptions*. We follow this principle by: (1) defining a series of dimensions that characterize the descriptions; (2) we let the users indicate their preferential value for these dimensions. Specifically, we identified three dimensions for characterizing a description from a user perspective: *coverage*, *degree*, and *diversity*. The *coverage* is the percentage of items covered by a description in each partition where the dataset is divided. Through the coverage, the users select from one extreme (i.e., low coverage) if they want to find the peculiarities for each cluster of matching items (i.e., possible mistakes); from the other (i.e. high coverage), if they want to describe the common features for all the entities in a matching group. The *degree* refers to the number of predicates used to describe each cluster of entities. Intuitively, a description with lower *degree* tends to be much easier to read. The *diversity* measures how often attributes are shared in describing different clusters of matching

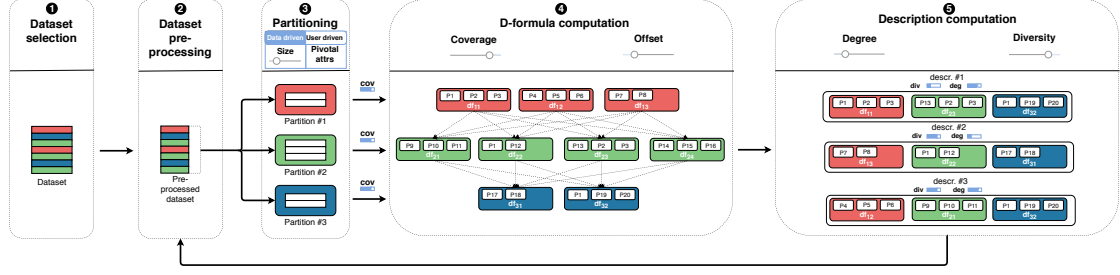


Figure 1: The process for generating descriptions.

entities. While repetitions of a set of attributes in different partitions increase readability, they could reduce the fine-grained representation of partitions.

The rest of the paper is organized as follows. Section 2 summarizes the approach for building the descriptions, Section 3 shows the application to the entity matching task, Section 4 provides some related work, and finally Section 5 sketches out some conclusion.

2. The Approach

The description generation process is interactive: users can specify their preference parameters—namely *coverage*, *degree* and *diversity*—and the system generates the best description according to these preferences. The user can repeatedly vary the preferences and generate further descriptions until the explanation need is fulfilled.

Figure 1 illustrates the 5 steps driving the user in the generation process of descriptions. First of all, the user starts the process by uploading a new dataset (❶). Once the dataset is selected, the user can specify the attributes of interest that must be considered by the system when generating the descriptions (❷, by default all attributes are used). Next, the user is guided through the UI into the 3 major phases in which the actual descriptions are generated.

In the first phase (❸), the input dataset is partitioned such that the tuples that are expected to be described together reside in the same group. The user is in charge to select how to partition the dataset. If the dataset describes the results of an EM task, the partition should be defined on an extra value identifying the clusters of matching entities.

The generation of the d-formulas (conjunction of predicates) happens during the second phase (❹). For each partition, the number of possible d-formulas is exponential over the number of attributes’ values. Generating all possible d-formulas is, therefore, prohibitively expensive. As explained in Section 2.1, we adopt a heuristic procedure that allows us to prune d-formulas that are less relevant for the task at hand. In the last phase (❺), the actual descriptions are computed by combining d-formulas over different partitions. Intuitively, we assemble the top- k descriptions that maximize a scoring function computed on the generated d-formulas that takes into account the user preferences of degree, and diversity. This problem can be solved with a dynamic programming approach: in Section 2.2 we will employ a variant of the Viterbi Algorithm called *LVA* [5] (a.k.a. *List Viterbi*), although any other algorithm can be used.

2.1. Building d-formulas

The approach starts building all feasible and relevant d-formulas for each partition. The number of candidate predicates for each partition depends on the size of the active domain of the attributes in the partition. Therefore, the complexity of this phase is $\mathcal{O}\left(\rho \times \prod_{a \in A} (2^{|dom_p(a)|})\right)$, where A is the set of the attributes, ρ is the number of partitions and $|dom_p(a)|$ is the aforementioned size. Given the complexity for generating all possible d-formulas, we follow a heuristic process that generates the most relevant candidate d-formulas only. We adopt two heuristics, one for pruning prolix d-formulas, and one for pruning d-formulas and predicates with undesirable selectivity.

2.1.1. Heuristic 1 - pruning prolix d-formulas

A low degree is specified when the user prefers a small number of predicates and, conversely, a high degree is for users who prefer descriptions with wide d-formulas. We push the degree intent of the users into the process of building d-formulas in order to early-prune d-formulas that do not meet such a requirement. This limits the number of predicates to evaluate, and it improves the efficiency of the process.

2.1.2. Heuristic 2 - pruning d-formulas with undesirable selectivity.

The coverage parameter (cov) indicates the desired percentage of tuples that make a d-formula true. Heuristic 2 transforms the coverage value indicated by the user input parameter value into an interval $[cov_l, cov_u]$ of admissible coverage values. The width of the interval is proportional to the coverage itself given that $cov_l = cov - offset(cov)$, $cov_u = cov + offset(cov)$ and $offset(cov) = 0.14 * cov^2$. As for Heuristic 1, we push the coverage interval into the process of generating d-formulas for early pruning.

2.1.3. Generating d-formulas.

The approach adopted for generating all possible and relevant d-formulas takes in input the list of partitions, the list of dataset attributes, the desired coverage, and degree. The d-formulas of each partition are computed separately. First, we generate the *atomic predicates* (i.e., predicates with only one value). Atomic predicates are generated only if they are within the expected coverage as defined by Heuristic 2. Then the atomic predicates are combined together to generate conjunctions of predicates. The resulting d-formulas, for each partition, are then returned as output.

When combining the predicates together, an internal routine makes sure that only those valid combinations (i.e., d-formulas) for Heuristic 2 are actually returned. The combination of the input predicates is organized in a lattice which is dynamically generated using a loop. In each iteration, we generate one level of the lattice (the n -th) by combining pairs of the previously generated predicates into one d-formula. Only combinations of predicates whose coverage is within the interval defined by Heuristic 2 are kept for the next level of the lattice. The generation of the lattice ends when the threshold, as per Heuristic 1, is reached or when it is no longer possible to generate predicates.

2.2. Building top-k descriptions

Selecting the best d-formulas in isolation does not necessarily lead to the best descriptions. We need to search for the optimal set of d-formulas across partitions that all together best fit the users' needs. To achieve this, we use dynamic programming: our current implementation is based on the *List Viterbi Algorithm* [5]. Viterbi Algorithm and its variants require to model the search space as a trellis. We build a vertical slice for each partition. The nodes in the slice are the d-formulas we have found in the previous phase. Nodes of a slice are connected to all the nodes of the next slice via weighted edges. A path represents a list of d-formulas, with at most one d-formula for partition. The algorithm is used to find the best full path.

Viterbi needs an objective function w to score a path, which intuitively represents the score of the (intermediate) descriptions we are computing.

We introduce a scoring function which measures (1) the adherence between the features of the description and the expectation of the user (expressed with a preference on level of coverage, degree and diversity); and (2) the goodness of the chosen attribute in describing the partitions. Other techniques can be evaluated. For example, we experimented also with the Smooth Local Search [6] with no improvement in the performance.

3. Applying Descriptions to the EM scenario

When we need to explain entity matches, it is natural to create descriptions with a d-formula for each cluster of matching entities. The use of descriptions enables the computation of different types of explanation, obtained by varying the preferences of coverage, degree and diversity specified by the user.

What do these entities represent? When users are interested in understanding the main features of the entities in the dataset, they have to select settings with high coverage values. These configurations generate descriptions where d-formulas aim to represent the clusters of matching entities as whole units. Descriptions with high degree tend to create d-formulas with more predicates. Descriptions with high diversity tend to use different attributes in the predicate lists of different d-formulas.

Example 2. D_1 in Example 1 is a description with high coverage, low degree and low diversity. All clusters of matching entities shown in Table 1 are represented and the user can understand that the values of *Song Name* qualify the entities. D_2 is an example of a description with high coverage, low degree and high diversity. In this case, the approach tries to select different attributes for describing each partition.

$$D_2 : (M_E \in \{e_1\} \wedge \text{Song Name} \in \{\text{I Ai n't Livin' Long Like This}\}) \vee \\ (M_E \in \{e_2\} \wedge \text{Artist Name} \in \{\text{Keith Urban}\}) \vee \\ (M_E \in \{e_3\} \wedge \text{Song Name} \in \{\text{I Ca n't Go There (Acoustic Version)}\}) \vee \\ (M_E \in \{e_4\} \wedge \text{Genre} \in \{\text{Pop}\})$$

Descriptions with high degree tend to use the largest number of possible attributes for each partition. Clearly, degree and diversity are related dimensions and when the dataset has a low dimensionality as in Table 1, the generated descriptions cannot accomplish both the requirement.

Is this a mistake? When users are interested in discovering possible mistakes in clusters of matching entities, they need to generate low coverage descriptions, that apply only to a small portion for each partition. As before for high coverage descriptions, degree and diversity are used to qualify the d-formulas generated.

Example 3. D_3 is a description with low coverage, low degree and low diversity. It describes a unique partition, i.e., the cluster of matching entities e_3 . In this cluster, the entity with *Entity id* 7 has a genre which is different from the ones of the other entities in the same cluster. This can be a possible mistake that the domain expert needs to check. Note that D_3 shows the information about cluster e_3 , since it is not possible to generate low coverage d-formulas for the other partitions.

$D_3 : M_E \in \{e_3\} \wedge Song_Name \in \{I\ Ca\ n't\ Go\ There\ (Acoustic\ Version)\} \wedge Genre \in \{Urban\ Cowboy\}$

In this case, the expert will probably validate the cluster, since all songs in e_3 actually refer to the same real-world item. Nevertheless, let us suppose that the EM model erroneously generates a cluster of matching entities which is the concatenation of the entities in e_1 and e_2 . D_4 is a low coverage description for this new cluster. In this case, the item reported by this description will probably be evaluated by an expert as a mistake, since the song with *Entity id* 4 is completely different from the other ones.

$D_4 : M_E \in \{e_1 + e_2\} \wedge Song_Name \in \{Billy\} \wedge Genre \in \{Country\}$

4. Related Work

Describing datasets. Explanation systems help users in gaining knowledge on the behavior of systems, experiments or query answers [7], and “black box” complex models [8]. Explanations typically assume the form of association rules, decision lists and decision sets [9]. Our approach performs data explanation since it creates partitions from a dataset and builds rules that provide users with an explanation of their content. The paper develops a subgroup discovery technique for performing data explanation and exploration on structured datasets [10].

Explainable Entity Matching. Entity matching [11] represents one of the main steps of data integration and has been under study for several years. Many techniques have been proposed: from the more traditional rule-based approaches to the most recent machine learning and deep learning methods. Rule-based approaches [12] are intrinsically interpretable, however, the identification of the most effective set of matching rules is a complex and non-trivial task [13]. EM approaches based on Deep Learning (e.g., DeepER [14], DeepMatcher [15], DITTO [16] and many others [17]) have been demonstrated particularly effective. Nevertheless, they require a significant amount of annotated data, they need a complex configuration, and there is no direct interpretation of their behavior, affecting their usability in business environments. There are typically two alternative approaches for providing explanation of AI techniques [18]: 1) applying explanation systems to interpret their behavior a-posteriori or 2) building models that are interpretable by design, i.e. models that base their decisions on humanly interpretable “structures / components”. The main approaches in EM (e.g., LIME [19] and SHAP [20], Mojito [21] and Landmark Explanation [22]) belong to the first area.

5. Conclusion

This discussion paper shows how *data descriptions* can be used for evaluating an EM task. In particular, descriptions can be used to generate an overview of the main features of the entities in the dataset or for discovering outliers, which can be the symptom of a mistake in the approach. Interested readers can find further details of the approach and a deep evaluation in [2, 3].

References

- [1] P. Christen, *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, Data-Centric Systems and Applications, Springer, 2012.
- [2] M. Paganelli, P. Sottovia, A. Maccioni, M. Interlandi, F. Guerra, Explaining data with descriptions, *Inf. Syst.* 92 (2020).
- [3] M. Paganelli, P. Sottovia, A. Maccioni, M. Interlandi, F. Guerra, Understanding data in the blink of an eye, in: *CIKM*, ACM, 2019, pp. 2885–2888.
- [4] D. Firmani, B. Saha, D. Srivastava, Online entity resolution using an oracle, *Proc. VLDB Endow.* 9 (2016).
- [5] N. Seshadri, C.-E. W. Sundberg, List viterbi decoding algorithms with applications, *IEEE Transactions on Communications* 42 (1994).
- [6] U. Feige, V. S. Mirrokni, J. Vondrák, Maximizing non-monotone submodular functions, *SIAM J. Comput.* 40 (2011) 1133–1153.
- [7] A. Meliou, S. Roy, D. Suciu, Causality and explanations in databases, *PVLDB* 7 (2014) 1715–1716.
- [8] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, *CSUR* 51 (2018) 93.
- [9] H. Lakkaraju, S. H. Bach, J. Leskovec, Interpretable decision sets: A joint framework for description and prediction, in: *SIGKDD*, 2016, pp. 1675–1684. URL: <http://doi.acm.org/10.1145/2939672.2939874>.
- [10] M. Atzmueller, Subgroup discovery, *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 5 (2015) 35–49.
- [11] G. Papadakis, G. Mandilaras, L. Gagliardelli, G. Simonini, E. Thanos, G. Giannakopoulos, S. Bergamaschi, T. Palpanas, M. Koubarakis, Three-dimensional entity resolution with jedai, *Information Systems* 93 (2020).
- [12] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, N. Tang, Synthesizing entity matching rules by examples, *Proc. VLDB Endow.* 11 (2017) 189–202.
- [13] M. Paganelli, P. Sottovia, F. Guerra, Y. Velegrakis, Tuner: Fine tuning of rule-based entity matchers, in: *CIKM*, ACM, 2019, pp. 2945–2948.
- [14] M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, N. Tang, Distributed representations of tuples for entity resolution, *Proc. VLDB Endow.* 11 (2018) 1454–1467.
- [15] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Deep learning for entity matching: A design space exploration, in: *SIGMOD Conference*, ACM, 2018, pp. 19–34.
- [16] Y. Li, J. Li, Y. Suhara, A. Doan, W.-C. Tan, Deep entity matching with pre-trained language models, *Proc. VLDB Endow.* 14 (2020) 50–60. URL: <https://doi.org/10.14778/3421424.3421431>. doi:10.14778/3421424.3421431.
- [17] U. Brunner, K. Stockinger, Entity matching with transformer architectures - A step forward in data integration, in: *EDBT*, OpenProceedings.org, 2020, pp. 463–473.
- [18] M. Du, N. Liu, X. Hu, Techniques for interpretable machine learning, *Commun. ACM* 63 (2020) 68–77.
- [19] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?" explaining the predictions of any classifier, in: *Proceedings of the 22nd ACM SIGKDD*, 2016, pp. 1135–1144.
- [20] A. Ghorbani, J. Y. Zou, Data shapley: Equitable valuation of data for machine learning, in: *ICML*, volume 97 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 2242–2251.
- [21] V. D. Cicco, D. Firmani, N. Koudas, P. Meriardo, D. Srivastava, Interpreting deep learning models for entity resolution: an experience report using LIME, in: *aiDM@SIGMOD*, ACM, 2019, pp. 8:1–8:4.
- [22] A. Baraldi, F. D. Buono, M. Paganelli, F. Guerra, Using landmarks for explaining entity matching models, in: *EDBT*, OpenProceedings.org, 2021, pp. 451–456.