Entity Resolution on Camera Records without Machine Learning

SIGMOD 2020 Contest Finalist Paper

Luca Zecchini
Università degli Studi
di Modena e Reggio Emilia
Italy
zecchini.luca@libero.it

Giovanni Simonini
Università degli Studi
di Modena e Reggio Emilia
Italy
simonini@unimore.it

Sonia Bergamaschi
Università degli Studi
di Modena e Reggio Emilia
Italy
sonia.bergamaschi@unimore.it

ABSTRACT

This paper reports the runner-up solution to the ACM SIGMOD 2020 programming contest, whose target was to identify the specifications (i.e., records) collected across 24 e-commerce data sources that refer to the same real-world entities.

First, we investigate the machine learning (ML) approach, but surprisingly find that existing state-of-the-art ML-based methods fall short in such a context—not reaching 0.49 F-score. Then, we propose an efficient solution that exploits annotated lists and regular expressions generated by humans that reaches a 0.99 F-score. In our experience, our approach was not more expensive than the dataset labeling of match/non-match pairs required by ML-based methods, in terms of human efforts.

CCS CONCEPTS

• **Information systems** → **Entity resolution**.

KEYWORDS

Entity Matching, Entity Resolution, Data Integration, Data Wrangling

1 INTRODUCTION

1.1 Competition Description

This paper reports our solution to the ACM SIGMOD 2020 programming contest, which resulted runner-up. During the contest, we were asked to solve an *Entity Resolution* (ER) problem; in particular, to perform ER on a provided dirty dataset of specifications (i.e., records) referring to cameras, extracted from different sources—with the help of an available labelled dataset, if needed. In this context, the aim of ER was to identify all the specifications (i.e., records) which referred to the same real-world camera model. The solutions were evaluated according to the F-measure (harmonic mean of precision and recall, rounded up to two decimal places) reached on a secret evaluation dataset.

Dataset Description. The provided dataset, called D, is composed of 29,787 specifications collected from 24 e-commerce websites. Each specification is a list of *name-value* pairs describing a camera that is being sold on the website. As illustrated in Table 1, the distribution of the specifications across the sources is not uniform, with few sources contributing with most of the specifications (*www.ebay.com* and *www.alibaba.com*).

SOURCE	SPECS	SOURCE	SPECS
buy.net	358	www.henrys.com	181
cammarkt.com	198	www.ilgs.net	102
www.alibaba.com	7,972	www.mypriceindia.com	347
www.buzzillions.com	832	www.pcconnection.com	211
www.cambuy.com.au	118	www.pricedekho.com	366
www.camerafarm.com.au	120	www.price-hunt.com	327
www.canon-europe.com	164	www.priceme.co.nz	740
www.ebay.com	14,274	www.shopbot.com.au	516
www.eglobalcentral.co.uk	571	www.shopmania.in	630
www.flipkart.com	157	www.ukdigitalcameras.co.uk	129
www.garricks.com.au	130	www.walmart.com	195
www.gosale.com	1,002	www.wexphotographic.com	147

Table 1: Specifications extracted from each source

As for the attributes, Page Title is the only one present in each specification, while none of the other 4,660 attributes appears in all the sources or, in most cases, in all the specifications from the same source. Furthermore, the attributes suffer from problems of homonymy (same name but different meaning) and synonymy (same meaning but different names). Finally, many details about additional accessories are provided (e.g., lens, bag, tripod, etc.) even if they do not contribute to the identification of the entities—the same for many other attributes, such as the color.

Labelled Data. In addition to D (the complete dataset), a labelled dataset to train a model in case of a machine learning (ML) solution is provided. Differently from D, this dataset contains a list of specifications id pairs and the related binary label: 1, which denotes a match (i.e., both specifications are related to the same real-world camera model), and 0, which denotes a non-match.

The labelled dataset is provided in two versions: Y, generated from 306 specifications and available since the start of the competition, and W, which is larger (generated from 908 specifications) and includes all the couples already present in Y, made available only later. Table 2 reports the size of the labelled datasets and their internal distribution of matching and non-matching couples.

DATASET	COUPLES	MATCHES	NON-MATCHES
Dataset Y	46,665	3,582	43,083
Dataset W	297,651	44,039	253,612

Table 2: Characteristics of labelled datasets

The data sources are dirty, meaning that it is possible to find matches even among specifications coming from the same source. Moreover, the transitive closure is applied on the matches: if e_1 matches with e_2 and e_3, then also e_2 matches with e_3.

Evaluation Process. The competition's participants were asked to identify all the matching pairs of D. A portion of the ground truth of D was known only to the organizer, which returned the F-measure on that portion as feedback of each submitted solution. The last version submitted before the contest deadline was used to determine the final ranking. After the deadline, the top teams had to provide their code and a guide to execute it, in order to validate and certify the score (through manual inspection and a reproducibility test of the solution) and to measure its execution time; this happened in a network-isolated container with defined specifications (4x30 GHz processor, 16 GB main memory, 128 GB storage, Linux OS). Both training (not included in final execution time) and execution had to respect a maximum time limit each of 12 hours.

1.2 Our Solution and Paper Organization

Our solution is based on the definition of human-crafted rules and lists to detect the information about brands and models in the specification titles—all titles contain this information—and to normalize them in order to perform an equality join completing the ER process.

In our experience, the definition of brand-based rules obtained through the human study of data was not more expensive than labelling a general training dataset for a state-of-the-art ML-based solution, which cannot capture with the same accuracy all the brand-dependent patterns needed for detecting matches—i.e., the ML-based solution achieves lower F-score.

The remainder of the paper is organized as follows: Section 2 reviews the related work; Section 3 shows how the ML approach performs on our problem; Section 4 presents our solution and the results in the challenge; finally, the lesson learned is reported in the Section 5.

2 RELATED WORK

ER (Entity Resolution, a.k.a.: Entity Matching, Duplicate Detection, Record Linkage) has been studied for decades [6], but is still a relevant research problem, since it significantly relies on human effort for labelling training data, for generating rules, for blocking, and many other related tasks—see [1, 3, 4, 9, 12, 15] for a survey.

Frameworks have been proposed to support practitioners to solve this task within data preparation pipelines [5, 7, 11, 13]. Among all the proposed solutions for structured in the literature, the latest research outcomes indicate the Machine Learning (ML) approaches as most promising, achieving the best performance in publicly available benchmarks when training data is available. In particular, the state-of-the-art ER algorithms based on ML (and deep learning) are *Magellan* [5] and *DeepMatcher* [8], respectively.

Magellan is an ER tool that allows to perform ER by using supervised learning techniques. In practice, in the training phase it takes as input a set of labelled examples of matching and non-matching pairs of specifications and trains a binary classifier. Then, in the inference phase, it predicts the labels of unseen pairs of specifications, i.e., performing the ER on the unlabelled part of the dataset. Of course, a ML classifier needs feature engineering for processing pairs of specifications; thus, Magellan employs similarity functions computed on pairs of corresponding attribute values (e.g., Jaccard and cosine similarity, edit distance, etc.) as features of each specification pair. As classification algorithms, it implements all the most-common and best-performing ones, such as *decision tree*, *random forest*, *SVM*, *naïve Bayes*.

Magellan may be sensitive to the definition of the features (e.g., by selecting some similarity functions rather than others or by using different sizes for the tokenization of the attributes); hence, it might be hard sometimes to tune. On the other hand, by employing deep learning, this step is basically avoided: features are automatically extracted by the neural net. Moreover, the final results are often better, with the right neural net architecture. This is essentially the idea behind *DeepMatcher*. In particular, it provides four different kinds of architecture for EM tasks: *(i) SIF*, which determines a match or non-match by considering the words present in each attribute value pair, without caring about their order; *(ii) RNN*, which considers the sequences of words; *(iii) Attention*, which considers the alignment of words, without caring about word order; *(iv) Hybrid*, which cares about the alignment of sequences of words, selected as default model.

Regular expressions have been used in the ER context for discovering transformations for the entity reconciliation process (e.g., to uniform the model representations among different specifications of the same camera) [2]. This is known as the *golden record* approach and exploits labelled data to synthesize regular expressions that can be used to transform strings in a canonical way, for all the attributes of a record—which becomes the *golden record* at the end of the transformations. Our solution exploits regular expressions to isolate and normalize substrings that refer to a camera model, but allows brand-dependent transformations, which do not generalize to the whole dataset as the golden-record approach assumes.

3 HOW DID THE ML-APPROACH PERFORM IN THIS SETTING?

As mentioned in Section 1.1, the data sources from which the specifications were extracted have highly heterogeneous schemas, the attributes have ambiguous names and most of them are used only in a handful of specifications. This makes the schema alignment a hard task itself. Furthermore, in our experiments we were not able to find any attributes that would appreciably help in the resolution of the ER problem. Thus, we decided not to explore this route further—i.e., we do not perform a schema alignment. As a consequence, EM with *Magellan* and *DeepMatcher* is performed on the attribute Page Title, which is always present and contains in most cases the two elements sufficient to uniquely identify a camera: the brand and the model. Notice that both *Magellan* and *DeepMatcher* can operate only with an aligned schema, thus a different approach would not be possible.

We preprocess the labelled dataset to the format required by the libraries and set the attribute Page Title to lowercase, then we generate train, validation and test sets respecting a 3:1:1 ratio. All of these sets contain matches and non-matches following the same distribution, which reflects the one of the complete dataset.

Because of the extremely variable length of the attribute values, the results obtained by *Magellan* on Y are extremely poor (best F-measure of 0.40 with naïve Bayes). This because only a portion of the titles is relevant to determine a match. Since in most cases the useful information (Brand and Model) is located among the first

words of Page Title, the normalization is realized by truncating the string, considering only the first n words, with the best value of n determined in an empirical way as 4. Table 3 and Table 4 show the results obtained on the datasets Y and W, with the best performing model (RNN by *DeepMatcher*) chosen to face the challenge.

CLASSIFIER	PRECISION	RECALL	F-SCORE
Decision Tree	92.45%	90.64%	91.54%
Random Forest	93.91%	88.27%	91.00%
RNN	**97%**	**95%**	**96%**
Attention	100%	73%	84%
Hybrid	98%	73%	84%
SIF	46%	31%	37%

Table 3: Results on dataset Y

CLASSIFIER	PRECISION	RECALL	F-MEASURE
Random Forest	98.90%	95.03%	96.93%
SVM	98.29%	93.52%	95.85%
Logistic Regression	96.56%	89.10%	92.68%
Decision Tree	97.72%	87.10%	92.11%
Linear Regression	97.47%	79.27%	87.43%
Naïve Bayes	70.35%	94.13%	80.52%
RNN	**99.59%**	**96.96%**	**98.26%**

Table 4: Results on dataset W

Moving to the complete dataset requires to consider all the pairs of tuples generated by the Cartesian product (887,265,369 pairs of tuples), an amount that even considering the reduction due to the deletion of identities and reflexive pairs is not computationally affordable, thus *blocking* is needed.

We employ blocking to reduce the number of candidates, in particular we find that *token blocking* [10, 14, 16] on the truncated version of the attribute Page Title is a good choice. In practice, *token blocking* indexes two specifications in the same block if they share one of the first four words in their Page Title. Then, all possible pairs of specifications within each block are considered as candidates. This blocking strategy produces 3,914 blocks, yielding 54,000,932 candidate pairs, and on the labelled ones (i.e., dataset W), it achieves a precision of 0.28 and a recall of 0.99, making it a suitable candidate set for being processed in an acceptable amount of time.

In order to improve precision and recall, we also employ a *list of frequent useless words* to be deleted from Page Title, increasing the probability of finding relevant information among the 4 maintained words. This list is generated by looking for the most shared words and detecting the ones which do not express meaningful information, before focusing on more specific cases. Further improvements can be given by the use of aliases, to solve the problem of popular synonyms (e.g., `"fuji"` and `"fujifilm"`).

3.1 Results with the ML approach

On D the best performing configuration we were able to find could not go beyond a F-measure of 0.47. In detail, it achieves a recall of 0.85 but a very poor precision of 0.32, due to a serious problem of false positives—not solvable through a simple useless words removal.

The best result has been achieved by employing *DeepMatcher* with a RNN architecture.

3.2 Why does the ML approach fall short?

The problem of false positives is linked to the nature of the dataset: the matching can be determined only on little brand-dependent details (e.g., the variation of a single letter or digit). Because of its size, the labelled dataset can cover only a little portion of all possible relevant cases which can be met considering all specifications (furthermore, a lot of brands and models of D do not even appear in W). So, the similarity patterns learned by *Magellan* and *DeepMatcher* on a few brands and models cannot be effective on the whole dataset, since it is impossible to generalize them.

4 A REGEX APPROACH

Our solution is based on *regular expressions* (*regex*), developed as a variation of the blocking method used in the previous approach. In particular, exploring the data it is possible to notice that in most cases the model is expressed as a string composed by both letters and digits, so it can be retrieved using a regular expression, while the number of brands with a high distribution is quite limited, manageable through a (human-generated) list.

In most cases, by performing simple data cleansing operations (e.g., by removing special characters and white spaces) the extraction of the first alpha-numerical string allows to detect the right model. Using this in combination with a list of the most popular brands, in 19,513 out of the 29,787 specifications, both the brand and the model are detected. In the following we describe the complete procedure performed to get the final, complete result.

4.1 The Procedure

The camera specifications are read into a dictionary with an identifier `ID` and the attribute Page Title, considered in its entirety, in lowercase, and replacing all the punctuation characters with an idle character (except for the character "-", substituted by the empty string because it is often used inside model names). Then, Page Title is transformed in a list by tokenization, and the elements that have aliases are replaced with their basic form by employing an apposite human-generated dictionary—e.g., different or misspelled versions used to indicate the same brand, like `"fuji"` for `"fujifilm"` or `"cannon"` for `"canon"`. Afterwards, the list is used for the brand retrieval phase, to identify the brand of each camera—also for this phase a human-generated list, containing popular brand names, is employed.

For extracting the model type (needed for the actual ER) four lists for each brand are defined: the `prefixes` list and the `suffixes` list, which contain the recurrent elements that can appear separately from the model strings and that must be considered part of the model name; the `model` list, which is composed by those only alphabetical or only numerical words (so, they would not be detected by the defined regex system without the use of this list) that must be considered as models; the `exceptions` list, which contains words that are both alphabetical and numerical but that must be skipped because they do not represent a model, using also an additional list of suffixes that denote measures (e.g., "`mm`","`gb`", `etc.`). Finally,

an `equivalences` dictionary is employed to conform the different versions of a model name choosing a standard one.

If a prefix (suffix), identified through the prefix (suffix) list, appears in the list of tokens representing Page Title, it is concatenated to its next (previous) token. Scanning the resulting list, a model can be detected and assigned to the current specification if a token is both alphabetical and numerical (detected through the defined regular expression) and does not appear in the `exceptions` list, or if it is only alphabetical or only numerical and appears in the `model` list—the first detected model is assigned to the specification.

Once extracted the model, further brand-based operations can be performed to normalize it: some prefixes and suffixes are modified or removed (e.g., the suffixes that indicate the colors and that would cause false negatives); some models may have synonyms depending on the geographic area (e.g., *Canon EOS 450D* is sold in North America as *EOS Rebel XSi* and in Japan as *EOS Kiss X2*), so they must be normalized to a single standard in order to avoid false negatives; some models need the retrieval of additional information for the disambiguation (e.g., *EOS 5D Mark II*, *EOS 5D Mark III*, and *EOS 5D Mark IV* must be considered as different models): the models which are sensitive to this problem (e.g., *5D*) are stored in an additional *brand-related* list. If a model appears in this list, the information related to its edition (e.g., *mark* for Canon) in the attribute Page Title is retrieved and attached to the model name, in order to avoid false positives.

Thus, the result of the process is that a new attribute is added to each specification: `brand_n_model`. This attribute contains the concatenation of the extracted brand and model—of course, if they have been both detected.

If the detection was successful, the specification is appended to the `solved_specs` list, otherwise to `unsolved_specs` list.

Once concluded this first step of extracting brand and model from the specifications, the following step is to determine the matching pairs of specifications. An inverted index built from the `solved_specs` list generates clusters representing entities, by grouping elements according to the perfect match of the attribute `brand_n_model`. Then, the specifications in the `unsolved_specs` list are considered as matches only if the content of the attribute Page Title is exactly the same: once again, inverted index is build to generate clusters.

We finally generate the list of matching pairs by emitting all possible pairs of specifications from each identified cluster. We need this step to submit the final solution and compute its F-score.

Notice that all the lists and rules are human-crafted; yet, in our experience, this required human labor is no more expensive than the amount of work required for labelling a training dataset of match/non-match pairs for *Magellan* or *DeepMatcher*.

4.2 Results

After all refinements applied to the method, the final submission reached the top F-measure of 0.99, with precision equal to 0.99 and recall to 0.98. Other 4 teams were able to reach the top result, so the tie was broken according to the execution time of the solution, concluding the contest as runner-up.

5 CONCLUSION

While looking for a good solution for the contest's ER problem, we investigated the limits of state-of-the-art machine learning (Magellan) and deep learning (DeepMatcher) methods for ER. These methods are able to achieve good results when matches can be identified by means of similarity-based features, but in a lot of real-world scenarios it may happen that matching is based on small variations that make the generalization on the entire dataset of the learned patterns impossible—this is the case of camera models, which have tiny brand-dependent variations to distinguish a camera (an entity) to another. In fact, we found that ER on camera specifications requires human-designed rules and lists (prefixes and suffixes management, exceptions, etc.), which existing ML methods are not able to synthesize. Yet, in our experience, these rules and lists are no more expensive to build than a labelled dataset of match/non-match pairs required by ML-based methods. This suggests that when approaching an ER problem, to start collecting match/non-match labels might not be the first thing to do, and might not be necessary at all.

REFERENCES

[1] Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer.

[2] Dong Deng, Wenbo Tao, Ziawasch Abedjan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Guoliang Li, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Unsupervised String Transformation Learning for Entity Consolidation. In *ICDE 2019*.

[3] AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration.* Morgan Kaufmann.

[4] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 1–16.

[5] Yash Govind et al. 2019. Entity Matching Meets Data Science: A Progress Report from the Magellan Project. In *SIGMOD 2019*.

[6] Ivan P Fellegi and Alan B Sunter. 1969. A theory for record linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[7] Luca Gagliardelli, Giovanni Simonini, Domenico Beneventano, and Sonia Bergamaschi. 2019. SparkER: Scaling Entity Resolution in Spark. In *EDBT 2019, Lisbon, Portugal, March 26-29, 2019.* 602–605.

[8] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *SIGMOD 2018*.

[9] Felix Naumann and Melanie Herschel. 2010. *An Introduction to Duplicate Detection.* Morgan & Claypool Publishers.

[10] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederée, and Wolfgang Nejdl. 2013. A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces. *IEEE Trans. Knowl. Data Eng.* 25, 12 (2013), 2665–2682.

[11] 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. *Inf. Syst.* 93 (2020), 101565.

[12] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–42.

[13] El Kindi Rezig, Lei Cao, Michael Stonebraker, Giovanni Simonini, Wenbo Tao, Samuel Madden, Mourad Ouzzani, Nan Tang, and Ahmed K. Elmagarmid. 2019. Data Civilizer 2.0: A Holistic Framework for Data Preparation and Analytics. *Proc. VLDB Endow.* 12, 12 (2019), 1954–1957.

[14] Giovanni Simonini, Sonia Bergamaschi, and H. V. Jagadish. 2016. BLAST: a Loosely Schema-aware Meta-blocking Approach for Entity Resolution. *Proc. VLDB Endow.* 9, 12 (2016), 1173–1184.

[15] Giovanni Simonini, Luca Gagliardelli, Sonia Bergamaschi, and H. V. Jagadish. 2019. Scaling entity resolution: A loosely schema-aware approach. *Inf. Syst.* 83 (2019), 145–165.

[16] Giovanni Simonini, George Papadakis, Themis Palpanas, and Sonia Bergamaschi. 2018. Schema-Agnostic Progressive Entity Resolution. In *ICDE 2018*.