Profiling Entity Matching Benchmark Tasks

Anna Primpeli
anna@informatik.uni-mannheim.de
Data and Web Science Group
University of Mannheim
Mannheim, Germany

Christian Bizer
chris@informatik.uni-mannheim.de
Data and Web Science Group
University of Mannheim
Mannheim, Germany

ABSTRACT

Entity matching is a central task in data integration which has been researched for decades. Over this time, a wide range of benchmark tasks for evaluating entity matching methods has been developed. This resource paper systematically complements, profiles, and compares 21 entity matching benchmark tasks. In order to better understand the specific challenges associated with different tasks, we define a set of profiling dimensions which capture central aspects of the matching tasks. Using these dimensions, we create groups of benchmark tasks having similar characteristics. Afterwards, we assess the difficulty of the tasks in each group by computing baseline evaluation results using standard feature engineering together with two common classification methods. In order to enable the exact reproducibility of evaluation results, matching tasks need to contain exactly defined sets of matching and non-matching record pairs, as well as a fixed development and test split. As this is not the case for some widely-used benchmark tasks, we complement these tasks with fixed sets of non-matching pairs, as well as fixed splits, and provide the resulting development and test sets for public download. By profiling and complementing the benchmark tasks, we support researchers to select challenging as well as diverse tasks and to compare matching systems on clearly defined grounds.

CCS CONCEPTS

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

KEYWORDS

entity matching; benchmarking; profiling; reproducibility; baseline evaluation

ACM Reference Format:
Anna Primpeli and Christian Bizer. 2020. Profiling Entity Matching Benchmark Tasks. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412781

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412781

1 INTRODUCTION

Entity matching identifies records in one or more data sets that refer to the same real-world entity [4, 5, 9]. Being a central task in data integration, entity matching has been the focus of many research works as well as industrial projects for more than five decades [10].

In order to evaluate and compare different entity matching methods, a wide range of benchmark tasks has been developed and made publicly available. A benchmark task for entity matching consists of the following artifacts: 1) One or more data sets consisting of records describing real-world entities and 2) a set of correspondences stating for all or a subset of all record pairs whether they describe the same real-world entity (match) or different real-world entities (non-match). Many entity matching methods rely on supervised machine learning and the correspondence set is thus often split into training and test sets. Matching the schemata of the data sets is usually considered a separate problem and it is assumed that all correspondences between the attributes of the data sets are known. Figure 1 illustrates the artifacts of an entity matching task along the example of two data sets describing smartphones.

Understanding the difficulty and diversity of benchmark tasks is essential for the meaningful comparison of matching methods and the assessment of their strengths and weaknesses. Existing approaches for categorizing benchmark tasks focus only on the structure of the benchmark data sets but ignore the influence of the correspondence set on the difficulty of a matching task. For instance, Mudgal et al. [17] categorize matching tasks into the categories structured, textual, and dirty. However, textuality and structuredness alone are not enough to understand the specific challenges associated with a matching task: matching two data sets with records having textual descriptions and no corner cases, i.e. all matches have high textual similarity and all non-matches have low textual similarity, is less challenging than matching data sets with textual records and many corner cases. In addition to corner cases, the size of the correspondence set that is available for training also determines the capability of supervised learning methods to excel on a matching task. Therefore, the correspondence set needs to be considered in addition to the data sets for profiling benchmark tasks.

This paper fills this gap by proposing a set of profiling dimensions which capture properties of the data sets as well as the correspondence set. Based on these dimensions, we define five groups of benchmark tasks which capture specific matching challenges. We apply the dimensions as well as the grouping to systematically profile 21 benchmark tasks. The resulting grouping on the one hand verifies the utility of the proposed dimensions and on the other hand unveils the specific challenges associated with the benchmark tasks.

Data set A			
ID	name	price	brand
A1	i-phone 4s	200€	apple
A2	htc one m9	220€	htc

Data set B			
ID	name	price	brand
B1	iphone 4	190€	apple
B2	one m9	210€	htc

Correspondences	
A1-B1	non-match
A2-B2	match
A1-B2	non-match

Figure 1: Artifacts of an Entity Matching Task

In addition, we evaluate the difficulty of the 21 benchmark tasks by computing baseline evaluation results using standard feature engineering together with two widely-used classification methods (SVMs and Random Forests). In this way, we set a lower bound that new supervised matching methods should at least be able to beat. Finally, in order to ensure the exact reproducibility of matching results and the sound comparison of matching methods, we suggest a heuristic for complementing benchmark tasks which do not provide pre-defined non-matches as well as fixed development and test sets. We provide the resulting complemented benchmark tasks for public download[1].

The contributions of our work are summarized as follows:

- We define a set of profiling dimensions for analyzing entity matching tasks which captures characteristics of the correspondence set in addition to the data sets.
- We create groups of benchmark tasks having similar characteristics and associated challenges.
- We evaluate the difficulty of 21 benchmark tasks by establishing baseline evaluation results.
- We complement existing benchmark tasks and make all resulting artifacts publicly available in order to support the reproducibility and comparability of matching results.

This paper is structured as follows: Section 2 introduces the benchmark tasks that we analyze throughout the paper. Section 3 describes the employed heuristic for complementing existing benchmark tasks. Section 4 presents the profiling dimensions of matching tasks that we extract and use for grouping. In Section 5, we present the baseline matching methods and report the baseline results for each task. Section 6 discusses the related work in the areas of profiling matching tasks as well as reproducing matching results.

2 BENCHMARK TASKS

Various repositories provide entity matching benchmark tasks for public download in an effort to facilitate the evaluation, reproducibility, and comparability of matching methods. The Database Group of the University of Leipzig[2] has published several benchmark tasks in 2010 which are widely used by the community since then. The Magellan repository[3] is maintained since 2015 by the Data Management Research Group of the University of Wisconsin-Madison. It includes a large collection of benchmark tasks, a subset of which has been created by the same research group while some tasks have been collected from other repositories. The DuDe repository of the Hasso Plattner Institute[4], provides three benchmark tasks originally published by other sources which have been modified to better serve the matching setting [7]. Finally, the Web Data Commons project, which is maintained by the Data and Web Science Group of the University of Mannheim, has published several e-commerce related benchmark tasks[5,6].

We collect and profile 21 benchmark tasks from these four repositories. Table 1 provides information about the selected tasks including references to publications which use the corresponding task. Additionally, we report basic profiling information concerning the amount of data sets from which the records originate, the number of records in the data sets, the amount of matching and non-matching record pairs in the correspondence set, whether fixed development and test splits are provided, the number of attributes having specific data types, and the average density (ratio of non-null values to all values) of the attributes of all records appearing in the correspondence set. We consider an attribute to be of data type long string, if the average length of its values exceeds six words. As shown in Table 1, the tasks are highly diverse and include data sets of different sizes, amounts of attributes, density as well as attribute data types. Most benchmark tasks contain records deriving from two data sets. In contrast, the tasks provided by the Web Data Commons Product repositories include records from up to 269 data sets.

Six matching tasks provide the complete mapping (i.e. all matching record pairs) while no non-matching pairs are offered. For the rest of the tasks a subset of both matching and non-matching record pairs is included in the set of correspondences. For the cases where the complete mapping is provided, non-matching pairs can be generated by calculating the Cartesian product of all records and excluding the matching pairs. Given the size of the data sets, this often results in large numbers of non-matching pairs and thus motivates the usage of blocking techniques [8, 18] to remove obvious non-matches which are not helpful for training and uninteresting for testing. As the benchmark tasks only define matches, different researchers who use these tasks, generate different sets of non-matches which influence the model training [2, 16, 17]. Given that not all of those works publish the complemented sets used for their experiments, it is not possible to exactly reproduce the evaluation of the methods.

Except for the large Web Data Commons tasks, it is uncommon for benchmark tasks to provide fixed splits for model development and model testing. Researchers using the tasks thus split the correspondences into development and test sets by themselves often without providing the details required for reproducing the splits in their papers, such as the sampling tool and random seeds. This hampers the reproducibility of the experimental results.

[1] http://data.dws.informatik.uni-mannheim.de/benchmarkmatchingtasks/index.html
[2] https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution
[3] https://sites.google.com/site/anhaidgroup/useful-stuff/data
[4] https://hpi.de/naumann/projects/data-integration-data-quality-and-data-cleansing/dude.html
[5] http://webdatacommons.org/productcorpus/index.html
[6] http://webdatacommons.org/largescaleproductcorpus/v2/index.html

Table 1: Overview of the Benchmark Tasks

Matching Task [Used in]	#Data Sets	#Records L	#Records R	#Matches	#Non-Matches	Fixed Splits	Attributes #Short Str.	Attributes #Long Str.	Attributes #Num.	Density
Leipzig Database Group										
abt-buy [16, 17]	2	1,081	1,092	1,095	-		0	2	1	0.63
amazon-google [16, 17]	2	1,363	3,226	1,298	-		1	2	1	0.75
dblp-acm [2, 16, 17]	2	2,614	2,294	2,223	-		1	2	1	1.00
dblp-scholar [2, 8, 16, 17]	2	2,616	64,263	5,346	-		1	2	1	0.81
DuDe										
$\text{restaurants}_{\text{Fodors-Zagats}}$ [17]	2	533	331	112	-		5	0	0	1.00
cora [13, 24]	1	1,879		64,578	268,082		11	6	1	0.31
Magellan										
$\text{products}_{\text{Walmart-Amazon}}$ [2, 17]	2	2,554	22,074	1,154	-		6	3	1	0.84
baby products	2	5,085	10,718	108	292		7	2	7	0.42
beer [16, 17]	2	4,345	3,000	68	382		4	0	0	0.96
bikes	2	4,785	9,002	130	320		5	0	3	0.78
$\text{books}_{\text{Goodreads-Barnes}}$	2	3,967	3,700	92	305		6	1	3	0.53
cosmetics	2	6,443	11,026	128	280		2	1	0	0.94
$\text{music}_{\text{iTunes - Amazon}}$ [2, 16, 17]	2	6,906	55,932	132	407		7	0	0	0.99
$\text{restaurants}_{\text{Yellow - Yelp}}$ [16]	2	5,223	11,840	130	270		5	0	1	1.00
Web Data Commons										
phones [21, 22]	17	447	50	258	22,092		22	1	3	0.25
headphones [21, 22]	6	444	51	226	22,418		21	3	3	0.13
tvs [21, 22]	8	428	60	182	25,499		49	9	3	0.07
xlarge_cameras [16, 19]	269	3,665		7,478	35,899	✓	1	3	0	0.51
xlarge_watches [16, 19]	190	4,068		9,564	53,105	✓	1	3	0	0.43
xlarge_computers [16, 19, 20]	235	4,676		9,991	59,571	✓	1	3	0	0.50
xlarge_shoes [16, 19]	120	2,808		4,440	39,088	✓	1	3	0	0.41

3 TASK COMPLETION

In order to exactly reproduce evaluation results, the correspondence set must contain the same non-matches and it must be split into a fixed development and test set. As explained in section 2, most benchmark tasks do not provide fixed splits.

We complement the correspondence sets of the benchmark tasks that provide a complete mapping but do not provide non-matches using the following heuristic: First, we construct all non-matching record pairs by calculating the Cartesian product of all records and exclude the pairs appearing in the complete mapping. We restrict the amount of non-matching pairs to allow for reasonable duration of feature creation and training by blocking using the domain-dependent label of the records, e.g. title for records describing books or product name for records describing products. The blocks of non-matching pairs are created using relaxed Jaccard with inner Levenshtein distance and a threshold of 0.2. We select all non-matching pairs within each block. To avoid selection bias, we add randomly selected non-matching pairs, the amount of which equals to 25% of the amount of non-matching pairs sampled within the blocks. To allow a high diversity of distinct records appearing in the correspondence set, we allow each record to appear in at most 10 non-matching record pairs. Finally, we apply stratified sampling and split the correspondence sets of the tasks that do not provide fixed splits, into training (70%), optimization (20%), and test (10%) sets.

4 MATCHING TASK PROFILING

This section introduces the profiling dimensions that we use for analyzing the tasks and describes our strategy for selecting attributes that are relevant for entity matching. Afterwards, we present the profiling results for the 21 tasks and create groups of tasks having similar characteristics.

4.1 Relevant Attributes

Attributes that do not contribute to the solution of a matching task are not relevant for understanding the task-related challenges and therefore should be excluded from the profiling.

We identify relevant attributes by learning a Random Forest and by selecting attributes based on the importance of the features of the trained model. For this, we calculate various pairwise record similarity scores for each attribute using a set of data type specific similarity metrics. All details about the feature generation are provided in section 5.1. The feature selection method starts by fitting a Random Forest classifier to the complete feature vector and retrieving the weights of the features, i.e. the feature importances assigned by the trained classifier. The Random Forest classifier is evaluated using four-fold cross validation. The score of this evaluation $F1_n$ is the F1 score that can be reached when all attributes are used for feature generation and learning. Next, the features are sorted in descending order given their weights. We iterate over the sorted features and in each iteration m we reduce the feature vector to

top m features. The reduced feature vector is used for learning a new Random Forest classifier which is evaluated with $F1_m$. We stop iterating once the model learned using a reduced feature vector reaches the quality of the model learned on the complete feature set, i.e. $F1_m >= F1_n$. Algorithm 1 gives the pseudo code of our feature selection strategy.

We identify relevant attributes by projecting the relevant features to single record attributes, e.g. *title_Levenshtein* to *title*. In contrast to key detection, which finds the combinations of attributes that uniquely identify records in a data set, our attribute selection strategy finds attributes whose values when compared pairwise with one or more similarity metrics, help reveal if a pair of records is matching or non-matching.

The coverage of models learned with different combinations of attributes can significantly vary as not all attributes contribute equally to the solution of the matching task [21]. Discovering the set of attributes that encode the most-identifying information, is crucial for the extraction of more focused profiling meta-information. We define these attributes as *top relevant attributes* and approximate their calculation using the following simple rule: the *top relevant attributes* are the attributes that when used for pairwise feature generation and model learning, make up for 95% of the maximum F1 score, $F1_n$. Therefore, we use the same feature selection strategy as described by Algorithm 1 with the only difference that the selection condition (line 8 in the pseudo code) is now $F1_m >= 0.95 \times F1_n$.

Algorithm 1 Feature selection algorithm

Input: $D_n : r \times n$ matrix, $n > 0$
Output: $D' : r \times m$ matrix, $0 < m <= n$

1: $model_n = RandomForest(D_n)$
2: $F1_n = cross\_validate(model_n, D_n)$
3: $sorted\_features = sort\_desc(model_n.feature\_importances)$
4: **for** $m = 1$ *to* n **do**
5: $\quad D' \leftarrow D_n[r, sorted\_features.top(m)]$
6: $\quad model_m = RandomForest(D')$
7: $\quad F1_m = cross\_validate(model_m, D')$
8: $\quad$ **if** $F1_m >= F1_n$ **then break**
9: $\quad$ **end if**
10: **end for**
11: **return** D'

4.2 Profiling Dimensions

We define five profiling dimensions that evolve around specific matching-related challenges: schema complexity, textuality, sparsity, development set size and corner cases. In the following, we present the motivation behind each profiling dimension and explain how it is calculated.

Schema Complexity (SC) This dimension refers to the amount of attributes that contribute to solving a matching task. A high schema complexity suggests a larger amount of underlying matching patterns and thus might be harder for a learner to solve. We approximate schema complexity by the number of relevant attributes of a matching task, calculated with the heuristic introduced in section 4.1.

Textuality (TX) Attribute values that consist of long sequences of words, e.g. the title of a product offer in e-commerce, often contain information that could also have been represented using multiple attributes. Such attributes are challenging for matching methods as they need to either separate the attributes in pre-processing or apply similarity metrics that can deal with below 1st normal form values. We calculate the textuality of a matching task as the average length value in words split by white-space, of the top relevant attributes of the records appearing in the set of correspondences.

Sparsity (SP) Having missing values in a supervised learning setting is a well recognized challenge for machine learning. In a classification setting with a non-dense feature vector, the learning algorithm has to learn multiple classification functions with each function covering a subset of the data points having the same missing features [12]. We calculate sparsity as the ratio of missing attribute values to all attribute values of the relevant attributes of records that appear in the correspondence set of the matching task.

Development Set Size (DS) The difficulty of a matching task also depends on the amount of correspondences that are available for learning and selecting matching models. This is an instantiation of the general observation that the performance of a classifier is affected by the size of the training set, while small sample sizes tend to cause classification models to overfit the data used for training and optimization [1]. We calculate the development set size as the amount of matching and non-matching record pairs in the training and optimization sets of a matching task.

Corner Cases (CC) The dimension of corner cases aims to capture the amount of record pairs that are non-matching but their attribute values are similar and the ones that have attributes values of low similarity but are matching. This dimension is crucial for understanding the difficulty of a matching task. In a trivial matching setting with no corner cases, the matching record pairs lie far from the non-matching record pairs in the hypersurface of the vector space and can be thus perfectly separated with a decision boundary. A less trivial matching task, includes record pairs that are close to the decision boundary. For these corner cases the learning algorithm needs to make further adjustments.

We use this observation and the simple case of a linear classifier for approximating the amount of corner cases of a matching task which we compute with the following heuristic: for every record pair in the correspondence set we calculate an aggregated similarity score by averaging the pairwise similarity score-based feature values, the computation of which is detailed in section 5.1, of the top relevant attributes. Feature values that cannot be computed because of missing corresponding attribute values of one or both records of the pair, are excluded from this aggregation. With every record pair being represented by its aggregated similarity score, we iteratively search in steps of 0.01 for the threshold value that can best separate the matching from the non-matching pairs, i.e. maximizes the F1 score. Once the optimal threshold is found, we measure the amount of matching pairs lying in the non-matching zone (false negatives) and the amount of non-matching pairs lying in the matching zone (false positives). To account for the class imbalance problem in entity matching, the corner cases dimension is calculated as the ratio of corner cases in relation to the amount of matching pairs: $\frac{\#false\_positives + \#false\_negatives}{\#matching\_pairs}$.

Table 2: Relevant Attributes and Profiling Results
SP = Sparsity, SC = Schema Complexity, TX = Textuality, DS = Development Set Size, CC = Corner Cases

Matching Task	Relevant Attributes	% of all	Profiling Dimensions SP	SC	TX	DS	CC
Group 1: Dense Data, Simple Schema							
beer	[[beer_name], brew_factory, ABV, style]	1.00	0.06	4	5.32	405	0.32
bikes	[[color, bike_name], price, km_driven]	0.44	0.00	4	5.37	405	0.09
$\text{music}_{\text{iTunes-Amazon}}$	[[song_name, time], album_name]	0.42	0.00	3	6.6	485	0.08
$\text{restaurants}_{\text{Yellow-Yelp}}$	[[phone], name, address]	0.50	0.00	3	2	360	0.05
$\text{restaurants}_{\text{Fodors-Zagats}}$	[[phone], name, address]	0.60	0.00	3	1.02	600	0.03
Group 2: Sparse Data, Complex Schema							
phones	[[phone_type,memory, display_size], color, mpn, ..]	0.38	0.39	10	4.8	20,137	0.42
headphones	[[model], mpn, impedance, sensitivity, accessories]	0.18	0.63	5	1.5	20,401	0.23
tvs	[[model], mpn, weight, height, width]	0.11	0.64	7	1.04	23,138	0.36
Group 3: Small and Difficult							
baby products	[[title, ext_id, SKU],colors, category, ..]	0.50	0.32	8	9.2	360	0.81
$\text{books}_{\text{Goodreads-Barnes}}$	[[pagecount, title], publisher, ISBN13, format, ..]	0.70	0.24	7	7.86	357	0.53
cosmetics	[[color, description], price]	1.00	0.07	3	9.36	367	0.27
Group 4: Textual Data, Few Corner Cases							
dblp-acm	[[title], year, authors]	0.75	0.00	3	7.65	42,079	0.02
dblp-scholar	[[title], authors]	0.50	0.02	2	7.8	74,689	0.07
$\text{products}_{\text{Walmart-Amazon}}$	[[title, modelno], long_description, brand, price]	0.50	0.08	5	10.52	14,036	0.27
cora	[[authors, volume, pages, year, title] , ..]	0.38	0.26	7	15.97	299,726	0.23
Group 5: Textual Data, Many Corner Cases							
abt-buy	[[name], description, price]	1.00	0.23	3	8.69	6,452	0.69
amazon-google	[[name, price, description]]	0.75	0.03	3	130.22	7,604	0.68
xlarge_cameras	[[title, description], brand, specTable]	1.00	0.38	4	98.71	42,277	0.78
xlarge_watches	[[title, description], specTable, brand]	1.00	0.43	4	109.79	61,569	0.64
xlarge_computers	[[title, description], brand, specTable]	1.00	0.44	4	51.69	68,462	0.86
xlarge_shoes	[[title, description]]	0.50	0.17	2	80.85	42,429	0.98

4.3 Profiling and Grouping the Matching Tasks

We identify the relevant attributes, calculate the values of the five profiling dimensions from each task, and create groups of tasks having similar characteristics. Table 2 presents the grouped benchmark tasks along with the attributes that are relevant for matching, the ratio of relevant attributes to all attributes, as well as the values of the five profiling dimensions sparsity (SP), schema complexity (SC), textuality (TX), development set size (DS) and corner cases (CC). The relevant attributes are listed in descending order of the corresponding feature importances. The inner brackets indicate the top relevant attributes.

Looking at the relevant attributes, we observe that in only 6 out of the 21 tasks all attributes were selected as relevant using the selection strategy described in section 4.1. For the remaining 15 tasks, a subset of the attributes is sufficient for reaching the same F1 score which would result from using all attributes.

Small development sets can lead to models that overfit the data. This is relevant for the bikes, $\text{books}_{\text{Goodreads-Barnes}}$, and cosmetics tasks. For these tasks, attributes which are more identifying, such as the book title and the bike model name, receive a lower importance weight in comparison to less identifying ones, such as the number of pages and the bike colour.

The profiling results show patterns which we use to create five groups of matching tasks. The grouping can help researchers to select interesting tasks for evaluating matching methods and to better understand and interpret matching results, i.e. with which challenges a matching method can successfully deal. In the following we present the five groups:

Group 1: Dense Data, Simple Schema. This group comprises of matching tasks with low schema complexity (≤ 4 relevant attributes), high density (> 0.94) and short attribute values (< 7 words). The following benchmark tasks have dense data and simple schema: $\text{restaurants}_{\text{Fodors-Zagats}}$, $\text{restaurants}_{\text{Yellow-Yelp}}$, $\text{music}_{\text{iTunes-Amazon}}$, bikes and beer. These tasks are expected to be easy to solve for traditional machine learning algorithms.

Group 2: Sparse Data, Complex Schema. In group 2 belong matching tasks that have non-dense attributes (< 0.60) with short values (< 5 words) and high schema complexity (≥ 5 relevant attributes). Under this group fall the following tasks: phones, headphones, and tvs. The matching methods used for evaluating these tasks need to especially address the challenge of low data density [21].

Group 3: Small and Difficult. In this group belong the matching tasks that have challenging characteristics, like high schema complexity and textuality, but only provide a small number of matches and non-matches which a classifier can use for learning

Datatype Detection		
	Title	long string
	Price $	numeric

A_1	Title	kodak single use 35mm blak
	Price $	12.50

A_2	Title	kodak power flash single use camera compact film 35mm black
	Price $	12

Feature Vector A_1-A_2	all	Title						Price $
	Cosine tf-idf	Cosine tf-idf	Levenshtein	Jaccard	Relaxed Jaccard	Containment	Overlap	Abs. Difference
	0.67	0.73	0.44	0.36	0.5	0.8	0.0	0.5

Figure 2: Feature Vector Creation Example

Table 3: Parameter Ranges of the Grid Search

RF	estimators [10, 100, 500]	max. depth [5, 10, 50, None]	min. leaf size [1, 3, 5]
SVM	C log(-2, 5), $\alpha = 10$	gamma $[1^{-6}, 1^{-4}, 1^{-2}, 1, 10]$	kernel [rbf, linear*]

and model selection. The small size of the training set makes it hard for classifiers to adapt to the different challenges and may lead to overfitting. The benchmark tasks that fall into this group are: $\text{books}_{\text{Goodreads-Barnes}}$, cosmetics, and baby products.

Group 4: Textual Data, Few Corner Cases. This group contains matching tasks with textual relevant attributes (> 7 words) and a low to very low containment of corner cases (≤ 0.27). The challenge imposed by the textuality dimension becomes trivial in the absence of corner cases. The benchmark matching tasks that fall under this category are: $\text{products}_{\text{Walmart-Amazon}}$, dblp-acm, dblp-scholar, and cora. The lower the containment in corner cases is, the less interesting a textual matching task is for evaluating state-of-the-art matching methods.

Group 5: Textual Data, Many Corner Cases This group contains matching tasks with high textuality and many corner cases. The benchmark tasks that can be categorized under this group are: xlarge_cameras, xlarge_watches, xlarge_computers, xlarge_shoes, amazon-google and abt-buy. For these tasks, embedding-based matching methods have shown to achieve better results than methods relying on symbolic features [17, 19, 20].

5 BASELINE EVALUATION

Considering baseline results is crucial for judging the difficulty of benchmark tasks. Baseline results set the lower limit of the predictive quality that a new matching method needs to achieve on a specific task and indicates if there is room for improvement for more sophisticated matching methods. In this section, we describe the matching methods that we employ for calculating baseline results. Afterwards, we apply the methods to the 21 benchmark tasks, present the baseline results, and discuss the difficulty of the tasks in relation to the task groups.

5.1 Feature Creation

We use symbolic features which we calculate using data type specific similarity metrics, similarly to the Magellan matching system [14]. As a first step, we therefore need to detect the data type of each attribute. Our data type detection heuristic distinguishes between three data types: short string, long string, and numeric. The long string data type is assigned to those attributes whose values have an average length larger than 6 words. In the case that more than one data types are detected for the same attribute, we assign long string, if long string appears in the list of detected data types, otherwise we assign short string.

Given the detected data type, multiple data type specific similarity metrics are used for constructing the feature vector. Feature values of data type short string are compared using the following similarity metrics: Levenshtein, Jaccard on the token level, Jaccard with inner Levenshtein, exact similarity, and containment similarity. The containment similarity is calculated on word level for any string value with more than one word, otherwise it is calculated on the token level. For long strings, the similarity metrics used for short strings are applied while Jaccard is calculated on the word level and additionally the cosine similarity with tf-idf weighting is computed. For numeric attributes, the absolute difference is computed. In addition to the attribute-specific similarity scores, we also concatenate all attributes values of a record and calculate an overall similarity score using cosine similarity with tf-idf weighting over the concatenated values.

All similarity scores are scaled to the range of [0,1]. In the case that the value of an attribute is missing for one or for both records of the record pair, we assign the out of range score -1. This allows any classifier to consider all record pairs without dropping or replacing the missing values. Figure 2 shows an example of the feature vector creation for two records A_1 and A_2 describing cameras.

5.2 Classification Methods

We employ two widely-used supervised classification techniques for learning baseline matchers: Support Vector Machines (SVM) and Random Forests (RF). We optimize the hyperparameters shown in Table 3 using grid search. The asterisk indicates that the linear kernel has not been used in combination with all other parameter values in the grid search like the rbf kernel, but only in one setting with default C and gamma values. For the non-optimized parameters we use the default values of the python scikit-learn library,

Table 4: Baseline Results and Comparison to Related Work

Profiling Group	Matching Task	SVM			Random Forest			Δ_{RF-SVM}	Best F1 Result in Related Work
		P	R	F1	P	R	F1		
Group 1: Dense Data, Simple Schema	beer	1.00	0.86	0.92	1.00	1.00	1.00	+0.08	0.78 [17]
	bikes	0.92	0.92	0.92	0.92	0.92	0.92	0.00	-
	$\text{music}_{\text{iTunes-Amazon}}$	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.94 [16]
	$\text{restaurants}_{\text{Yellow -Yelp}}$	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00 [17]
	$\text{restaurants}_{\text{Fodors-Zagats}}$	1.00	0.91	0.95	1.00	1.00	1.00	+0.05	0.97 [16]
Group 2: Sparse Data, Complex Schema	phones	0.85	0.88	0.86	0.85	0.88	0.86	0.00	0.84 [21]
	headphones	0.89	0.77	0.83	0.95	0.82	0.88	+0.05	0.94 [21]
	tvs	0.93	0.78	0.85	0.94	0.89	0.91	+0.06	0.83 [21]
Group 3: Small and Difficult	baby products	0.70	0.64	0.67	0.68	0.55	0.63	-0.04	-
	$\text{books}_{\text{Goodreads-Barnes}}$	0.80	0.89	0.84	0.73	0.89	0.80	-0.04	-
	cosmetics	1.00	0.77	0.87	0.90	0.69	0.78	-0.09	-
Group 4: Textual Data, Few Corner Cases	dblp-acm	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.98 [17]
	dblp-scholar	0.99	0.99	0.99	0.99	0.99	0.99	0.00	0.94 [17]
	$\text{products}_{\text{Walmart-Amazon}}$	0.97	0.87	0.92	0.93	0.89	0.93	+0.01	0.89 [11]
	cora	0.99	0.98	0.99	1.00	1.00	1.00	+0.01	1.00 [24]
Group 5: Textual Data, Many Corner Cases	abt-buy	0.96	0.71	0.81	0.95	0.77	0.85	+0.04	0.62 [17]
	amazon-google	0.79	0.73	0.76	0.82	0.76	0.79	+0.03	0.71 [16]
	xlarge_cameras	0.71	0.61	0.65	0.75	0.67	0.71	+0.06	0.93 [16]
	xlarge_watches	0.86	0.71	0.78	0.82	0.73	0.81	+0.03	0.96 [16]
	xlarge_computers	0.74	0.67	0.70	0.78	0.78	0.78	+0.08	0.97 [20]
	xlarge_shoes	0.83	0.43	0.57	0.82	0.38	0.52	-0.05	0.94 [19]

version 0.22.1. Finally, we use fixed random seeds for allowing the reproducibility of the baseline results by setting the random_state parameter of the scikit-learn classification models to 1.

5.3 Baseline Results

We evaluate the baseline matching methods using split validation and use precision, recall and F1 score on the positive class (matching) as evaluation metrics. We present the baseline results in Table 4. In addition we show the best reported result found in related work from matching systems using supervised learning while matching systems that use other types of learning (e.g. active learning or semi-supervised learning) are excluded from this comparison [13, 18, 22]. The interpretation of the comparison to the results reported in related work should be made with attention to the differing, unfixed train, optimization, and test sets.

Comparing the results of the two classifiers, we see that the SVM performs worse than the Random Forest for 17 out of 21 tasks. However, this is not the case for the xlarge_shoes, baby products, cosmetics and $\text{books}_{\text{Goodreads-Barnes}}$ tasks, for which the SVM outperforms the Random Forest by 0.04 to 0.09 in F1 score. In addition, six benchmark tasks are perfectly solved by the baseline method (F1=1.00), indicating that there is no room for improvement for methods using full supervision.

Considering the grouping of the tasks, we can see that each profiling group imposes varying difficulty levels to our baseline method. More concretely, F1 scores between 0.92 and 1.00 are achieved for the tasks belonging to *Group 1: Dense Data, Simple Schema*. For solving the tasks of this group, only a small amount of attributes needs to be considered by the learning algorithm. The matching patterns are simple which we verify by looking at the average depth of the trees of the best Random Forest estimator per task, ranging between 4.5 and 4.98. On the other hand, the tasks of *Group 2: Sparse Data, Complex Schema* are more difficult to solve, with the F1 scores being lower than 0.91, as a result of the higher sparsity and schema complexity. These characteristics require the matching algorithm to learn complex matching patterns. The average depth of the trees of the best Random Forest estimators for the tasks of Group 2 is 10.

Our baseline methods perform poorly for the tasks of *Group 3: Small and Difficult*, with F1 scores ranging between 0.67 and 0.87. We assume the reason of the poor performance to be on the one hand, the inherent challenging characteristics of the tasks, such as high textuality, a significant amount of corner cases and high schema complexity and on the other hand, the small amount of training data. These factors likely cause the model to overfit the training data and not generalize well.

The F1 scores achieved using the Random Forest classifier for the tasks of *Group 4: Textual Data, Few Corner Cases* are above 0.93, indicating that they have a low level of difficulty despite their high textuality. The small amount of corner cases makes the tasks trivial. For each of the tasks dblp-acm and dbpl-scholar, which contain almost no corner cases, there is only one top relevant feature: *title_cosine_tfidf* for the dblp-acm task and *title_relaxed_jaccard* for the dblp-scholar. This indicates that condensing the title attribute into a single similarity score is enough for solving both tasks. The other tasks of Group 4 contain a few corner cases but have the following characteristics which we hypothesize help the model to generalize well and achieve good performance: the cora task has a very large development set and the $\text{products}_{\text{Walmart-Amazon}}$ task has a model number attribute which can be identifying for resolving product records.

The performance of our baseline method drops for the matching tasks of *Group 5: Textual Data, Many Corner Cases* and ranges between 0.57 and 0.85. These tasks have many corner cases and highly textual relevant attributes, such as product titles and descriptions. The weak performance can be explained from the inability of the similarity-based feature vector to adequately summarize the textual data. In contrast to our symbolic, similarity-based features, the related works that report significantly higher F1 scores on these tasks, use embeddings and deep neural network-based matching methods which have been shown to perform better on textual data [17, 19, 20].

6 RELATED WORK

There are many survey articles comparing entity matching methods [5, 15] but there are hardly any works comparing and categorizing entity matching benchmark tasks. To the best of our knowledge, the work of Mudgal et al. [17] is the only one which profiles the structure of relational data sets used for matching and distinguishes three groups of tasks: structured (short attribute values), textual, and dirty, with the last group being created by removing with 50% probability the values of certain attributes and injecting them to other attributes. The profiling is conducted on the basis of the number of attributes, the length and the presence or absence of artificial noise in the attribute values. In our work, we show that concentrating only on data set characteristics is not sufficient for understanding the challenges associated with the matching tasks, as the difficulty of textual tasks is for instance highly affected by the amount of corner cases in the correspondence set. For tasks involving mostly short attributes, schema complexity, density, and development set size together determine the difficulty of the task. In contrast to entity matching, there exist survey papers on benchmark tasks for the closely related domain of entity linking [23]. In our work, we fill this gap and profile benchmark tasks for entity matching.

The reproducibility of research results is an important issue across all research communities. Therefore, initiatives such as SIGMOD reproducibility[7] invite researchers to make their experiments repeatable by sharing all relevant artifacts. In addition, there exist multiple campaigns that contribute to the meaningful evaluation of different systems, such as the SemEval workshop[8] with a focus on NLP tasks, the SemWebEval [3], and the OAEI contest[9] which both aim at evaluating tasks relevant to the Semantic Web community. Such campaigns provide benchmarks and fixed evaluation procedures. The OAEI contest publishes among others, schema and instance matching tasks for evaluating ontology matching systems. In contrast to the benchmark tasks that we profile in our work which contain relational data sets, these tasks use RDF data that includes class and property hierarchies which should be considered by successful matching systems. For such tasks a different set of profiling dimensions becomes relevant which has been analyzed in the work of Daskalaki et al. [6], such as the data set creation method, i.e. whether the data is real or synthetic, as well as the schema similarity of the data sets.

[7] http://db-reproducibility.seas.harvard.edu/
[8] http://alt.qcri.org/semeval2020/
[9] http://oaei.ontologymatching.org

7 CONCLUSION

We defined five dimensions for profiling entity matching tasks capturing characteristics of both the data sets as well as the correspondence set of the task. On the basis of these dimensions, we profiled 21 benchmark tasks and grouped them into five groups entailing similar matching challenges. In an effort to enhance the comparability and reproducibility of matching results, we complemented the benchmark tasks with fixed development and test sets which we make publicly available. Using the complemented benchmark tasks, we calculated baseline results using two matching methods and identified the difficulty level of the tasks in each group. Our results aim to help researchers select diverse tasks for showing the strengths and weaknesses of their matching systems.

REFERENCES

[1] M. H. G. Anthony and Norman Biggs. 1997. *Computational learning theory*. Vol. 30. Cambridge University Press.
[2] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures-a step forward in data integration. In *EDBT*. 463–473.
[3] Davide Buscaldi, Aldo Gangemi, and Diego Reforgiato Recupero (Eds.). 2018. *Semantic Web Challenges - 5th SemWebEval Challenge at ESWC 2018, Heraklion, Greece, June 3-7, 2018, Revised Selected Papers*.
[4] Peter Christen. 2012. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer.
[5] Vassilis Christophides et al. 2019. End-to-End Entity Resolution for Big Data: A Survey. *arXiv:1905.06397 [cs]* (2019).
[6] Evangelia Daskalaki et al. 2016. Instance matching benchmarks in the era of Linked Data. *Journal of Web Semantics* 39 (2016), 1 – 14.
[7] Uwe Draisbach and Felix Naumann. 2010. DuDe: The Duplicate Detection Toolkit. In *QDB Workshop*.
[8] Muhammad Ebraheem et al. 2018. Distributed Representations of Tuples for Entity Resolution. In *VLDB*. 1454–1467.
[9] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1 (2007), 1–16.
[10] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.
[11] Chaitanya Gokhale et al. 2014. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*. 601–612.
[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
[13] Jungo Kasai et al. 2019. Low-resource Deep Entity Resolution with Transfer and Active Learning. *arXiv preprint arXiv:1906.08042* (2019).
[14] Pradap Konda et al. 2016. Magellan: Toward Building Entity Matching Management Systems over Data Science Stacks. *PVLDB* 13 (2016), 1581–1584.
[15] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *PVLDB* (2010), 484–493.
[16] Yuliang Li et al. 2020. Deep Entity Matching with Pre-Trained Language Models. *arXiv:2004.00584 [cs]* (2020).
[17] Sidharth Mudgal et al. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *SIGMOD*. 19–34.
[18] George Papadakis et al. 2020. Domain- and Structure-Agnostic End-to-End Entity Resolution with JedAI. *ACM SIGMOD Record* 48, 4 (2020), 30–36.
[19] Ralph Peeters et al. 2020. Using schema. org Annotations for Training and Maintaining Product Matchers. In *WIMS*.
[20] Ralph Peeters, Christian Bizer, and Goran Glavaš. 2020. Intermediate Training of BERT for Product Matching. In *DI2KG Workshop @ VLDB*.
[21] Petar Petrovski and Christian Bizer. 2020. Learning expressive linkage rules from sparse data. *Semantic Web* 11 (2020), 549–567.
[22] Anna Primpeli and Christian Bizer. 2019. Robust active learning of expressive linkage rules. In *WIMS*. 1–7.
[23] Marieke van Erp et al. 2016. Evaluating Entity Linking: An Analysis of Current Benchmark Datasets and a Roadmap for Doing a Better Job. In *LREC*. 4373–4379.
[24] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity matching: how similar is similar. *VLDB Endow.* 4, 10 (2011), 622–633.