Evaluation of entity resolution approaches on real-world match problems

Hanna Köpcke
WDI-Lab
University of Leipzig
Germany
koepcke@informatik.uni-leipzig.de

Andreas Thor
Database Group
University of Leipzig
Germany
thor@informatik.uni-leipzig.de

Erhard Rahm
WDI-Lab & Database Group
University of Leipzig
Germany
rahm@informatik.uni-leipzig.de

ABSTRACT

Despite the huge amount of recent research efforts on entity resolution (matching) there has not yet been a comparative evaluation on the relative effectiveness and efficiency of alternate approaches. We therefore present such an evaluation of existing implementations on challenging real-world match tasks. We consider approaches both with and without using machine learning to find suitable parameterization and combination of similarity functions. In addition to approaches from the research community we also consider a state-of-the-art commercial entity resolution implementation. Our results indicate significant quality and efficiency differences between different approaches. We also find that some challenging resolution tasks such as matching product entities from online shops are not sufficiently solved with conventional approaches based on the similarity of attribute values.

1. INTRODUCTION

Entity resolution (also referred to as object matching, duplicate identification, record linkage, or reference reconciliation) is a crucial task for data integration and data cleaning [10], [18], [29]. It is the task of identifying entities referring to the same real-world entity. The high importance and difficulty of the entity resolution problem has triggered a huge amount of research on different variations of the problem and numerous approaches have been proposed especially for structured data. Recent surveys include [2], [20], and [24].

Due to the high number and diversity of different entity resolution approaches we see a strong need for comparative evaluations of different schemes. To date most entity resolution approaches have been evaluated individually using diverse methodologies, configurations, and test problems making it difficult to assess the overall quality of each approach, let alone their comparative effectiveness and efficiency. Only few attempts for comparative evaluations of some sub-approaches have been made, e.g., evaluation of different string similarity metrics [11] and of blocking approaches [3]. Some benchmark proposals for entity resolution have been made [28], [32] but they have not yet been implemented or applied.

We have analyzed numerous research publications w.r.t. their evaluation of entity resolution approaches[1] and provide an overview of selected studies in our recent survey paper [20]. While we identified several popular datasets used in the evaluations, e.g., DBLP, Citeseer, Cora, or IMDB, we were not able to derive conclusive results on the relative quality of different entity resolution approaches. This is because the published evaluation results are mostly based on selected subsets or preprocessed versions of these datasets so that the achieved results became incomparable. See, for example, the differently sized Cora-based datasets used in [25], [30], [13], and [12].

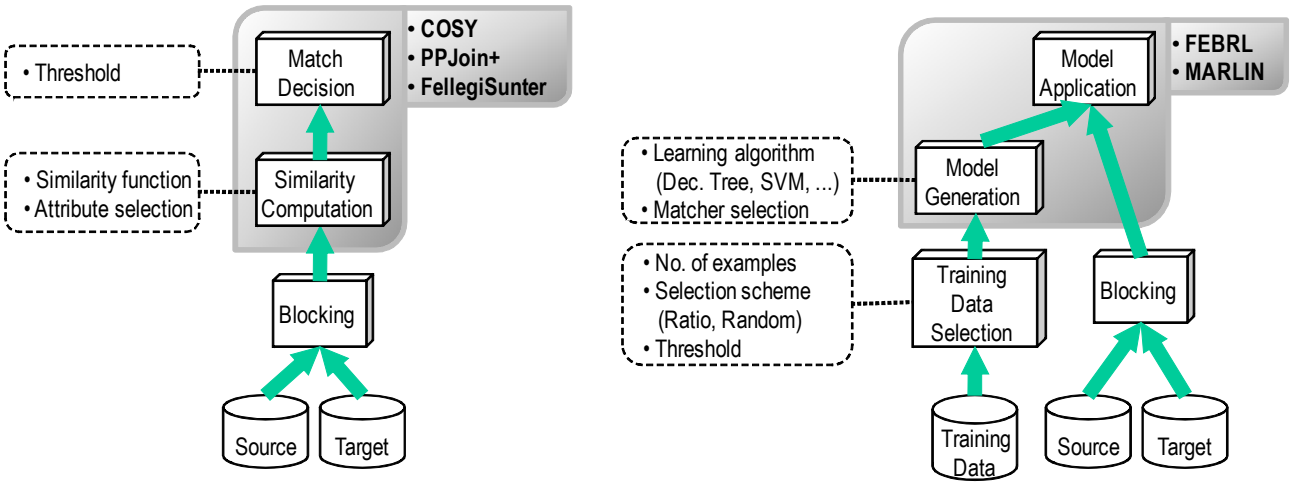
Another difficulty when comparing entity resolution algorithms is that they require different parameters to be set such as the similarity functions for comparing attribute values or similarity thresholds to be exceeded by matching entities. Many proposed approaches also make use of machine learning algorithms requiring specific parameters such as the size and characteristics of training data. Obviously, the chosen algorithm configuration is one of the predominant factors for the resulting match quality and in many published evaluation results significant details of it (e.g., on the used training data) remain unspecified.

In this study, we use a new evaluation framework, FEVER, to comparatively evaluate several previously proposed entity resolution approaches. Main characteristics of our evaluation are:

- The approaches are uniformly evaluated on four real-world match tasks of two domains. In particular we consider matching of product entities from different web shops.
- We consider individual algorithms (PPJoin+) as well as frameworks (FEBRL, MARLIN) offering different approaches. Furthermore we study approaches that do and do not require training data. In addition we consider a state-of-the art commercial entity resolution approach. More than 20 different approaches are evaluated under different parameter settings.
- Our evaluation considers both match quality in terms of precision, recall, and F-measure, as well as efficiency in terms of runtime.

[1] The complete list can be found at http://dbs.uni-leipzig.de/fever

Proceedings of the VLDB Endowment, Vol. 3, No. 1



(a) Non-learning match approaches **(b) Learning-based match approaches**

Figure 1. FEVER match workflows for evaluating existing entity resolution approaches

- We use the FEVER framework to automatically execute the approaches and to find favorable parameter settings in a comparable way. In particular, we always apply the same blocking method to reduce the search space and use a uniform approach for providing training to the machine-learning approaches. For the approaches not based on machine learning we spend the same effort for optimizing parameters such as similarity thresholds.

The rest of the paper is organized as follows: Section 2 describes the use of the FEVER framework to perform the evaluations of entity resolution approaches. The evaluation results are presented and discussed in Section 3. In Section 4 we briefly discuss related evaluation studies, especially of the approaches considered in our study. Finally, we conclude in Section 5.

2. EVALUATION APPROACH

We use the FEVER platform (Framework for EValuating Entity Resolution) [23] to evaluate several match approaches for different match tasks. While FEVER has its own library of match algorithms we do not evaluate this functionality here but use FEVER only to evaluate existing entity resolution approaches from the research community and one vendor. FEVER allows us to automatically execute these algorithms for many different parameter settings in a comparable way as we will discuss in the following for both non-learning and learning-based match approaches.

2.1 Non-learning match approaches

In FEVER, a match approach is specified by a so-called operator tree or workflow that specifies the sequence of processing steps for determining the match result on two input datasets. Figure 1a illustrates the FEVER operator tree that was applied in our evaluation of non-learning match approaches.

For large datasets, it is generally not feasible to exhaustively evaluate the Cartesian product of all input entities. Hence, we first apply a blocking operator to reduce the search space to the most likely matching entity pairs. For comparability, we use a fixed blocking strategy for all non-learning and learning-based match approaches, i.e., blocking is not subject of the evaluation.

The blocking result is input to the non-learning match approaches to be evaluated. In this study all considered match approaches are based on so-called attribute matchers that evaluate the similarity of attribute values based on some similarity function (e.g., an approximate string similarity). The approaches may evaluate only a single matcher (for a specific attribute pair and similarity function) or multiple matchers using different attribute pairs or similarity functions. In the latter case the approaches also need to support a combination of the individual similarities to derive a match decision. In our evaluation, we will always use the same attributes for comparability. Furthermore, all non-learning match approaches apply a threshold-based selection of the matching entity pairs and require the similarity threshold to be provided as a parameter.

For the similarity computation and the threshold-based match decision we used the implementation of the following non-learning match approaches:

- **COSY**: This is a state-of-the-art commercial system for entity resolution. Unfortunately, license restrictions do not allow us to disclose the name of the system. COSY uses its own similarity function that can be applied on one or several attribute pairs. The most important parameter to be provided is the *overall MinimumSimilarity* threshold. An entity pair will be considered a match only if it has a similarity that is greater than or equal to this threshold. Additional *attribute-level similarity thresholds* can optionally be specified for each attribute pair that should be considered in the computation of the entity similarity.
- **PPJoin+** [34] is a single-attribute match approach (similarity join) using sophisticated filtering techniques for improved efficiency. The approach has two parameters that need to be configured. The parameter *function* determines the similarity function used for the join. We will evaluate both supported implementations for the similarity function (Cosine, Jaccard). The parameter *threshold* determines the threshold for the similarity values above which entities are considered to match.

Table 1. Overview of real-world evaluation match tasks

Match task			Source size (#entities)		Mapping size (#correspondences)		
Domain	Attributes	Sources	Source 1	Source 2	Full input mapping (Cartesian product)	Reduced input mapping (blocking result)	perfect result
Bibliographic	- title - authors	DBLP-ACM	2,616	2,294	6 million	494,000	2,224
	- venue - year	DBLP-Scholar	2,616	64,263	168.1 million	607,000	5,347
E-commerce	- product name - description	Amazon-GoogleProducts	1,363	3,226	4.4 million	342,761	1,300
	- manufacturer - price	Abt-Buy	1,081	1,092	1.2 million	164,072	1,097

- **FellegiSunter** [15] is a non-learning approach from the FEBRL framework [9]. For similarity computation we evaluate three of the similarity measures provided by FEBRL (Winkler, Tokenset, Trigram). The approach has an *lower* and *upper similarity threshold* that can be adjusted. Entity pairs with a similarity above the upper classification threshold are classified as matches, pairs with a combined value below the lower threshold are classified as non-matches, and those entity pairs that have a matching weight between the two classification thresholds are classified as possible matches. For our evaluation, we set the lower threshold equal to the upper threshold as we only want a classification into matching and non-matching entity pairs.

An operator tree typically comprises several operators each having several parameters that need to be specified in order to apply the operator tree to a match problem. FEVER allows a systematic evaluation of operator trees for different parameter settings to help finding a suitable configuration [23]. For this study we limit the number of parameters to be set by applying a fixed blocking approach and manually pre-selecting the attributes to be evaluated. We further evaluate the existing similarity functions either on one or two attributes of the input datasets. In both cases we have to specify similarity thresholds on the single attribute or combined attribute similarity. For comparability, we evaluate every match approach for a fixed maximum number, *N*, of settings for the threshold parameters. FEVER supports several methods for selecting the parameter values such as manual (user-defined) and random. For this evaluation, we use the sophisticated and effective gradient descent strategy that iteratively refines a parameter setting by considering the quality of previously generated settings [23].

2.2 Learning-based match approaches

Figure 1b shows the FEVER operator tree applied for the evaluation of learning-based approaches. The execution falls into two phases: model generation and model application. The model generation (left part of the operator tree) requires a training dataset that contains manually labeled correspondences representing matching (similarity value equals 1) and non-matching (0) entity pairs. The learning algorithm applies the specified matchers to the entity pairs in the training data. The learner then uses the resulting similarity values to automatically determine a match strategy model, i.e., combination and parameterization of the specified matchers to derive a match decision for any entity pair. More details on training selection and model generation will be provided below. The second phase (right part of the operator tree) applies the determined model for the real match task (model application) to match a source and target dataset (or to find duplicates within one dataset).

For model generation, a pre-selected set of matchers is applied to the training data. By comparing similarity values computed by the matchers to the perfect (labeled) match result in the training it is possible to determine (learn) a combination of the most effective matchers and their parameters such as similarity thresholds.

In our evaluation we will compare several existing training-based approaches for model generation and application offered by the following frameworks:

- **FEBRL** [9] (Freely Extensible Biomedical Record Linkage) provides a support vector machine (SVM) implementation for learning suitable matcher combinations. For attribute matching we will evaluate the same three similarity measures than for the non-learning matchers studied for FEBRL.
- **MARLIN** [4] (Multiply Adaptive Record Linkage with INduction) offers two string similarity measures (Edit Distance and Cosine) and several learners, specifically SVM and decision trees. The learners can be used in a single step approach or can be employed for a two-level learning approach. For the two-level approach string similarity measures are first trained for every selected attribute so that they can provide accurate estimates of string distance between values for that attribute. Next, a final decision is learned from similarity metrics applied to each of the individual attributes.

The effectiveness of machine learning approaches is known to depend on the provision of sufficient, suitable, and balanced training data. On the other hand, the number of entity pairs to be labeled affects the manual tuning effort and should thus be small. To address these issues we build upon our evaluation experiences reported in [21] and only consider entity pairs for labeling for which the similarity exceeds a specified threshold *t*. This ensures that the training is not dominated by trivial non-matching entity pairs that are not useful to find effective matcher parameters and matcher combinations. We further strive at providing both matching and non-matching entity pairs by a training selection approach called *Ratio (r,t)*. It uses a ratio parameter *r* from the

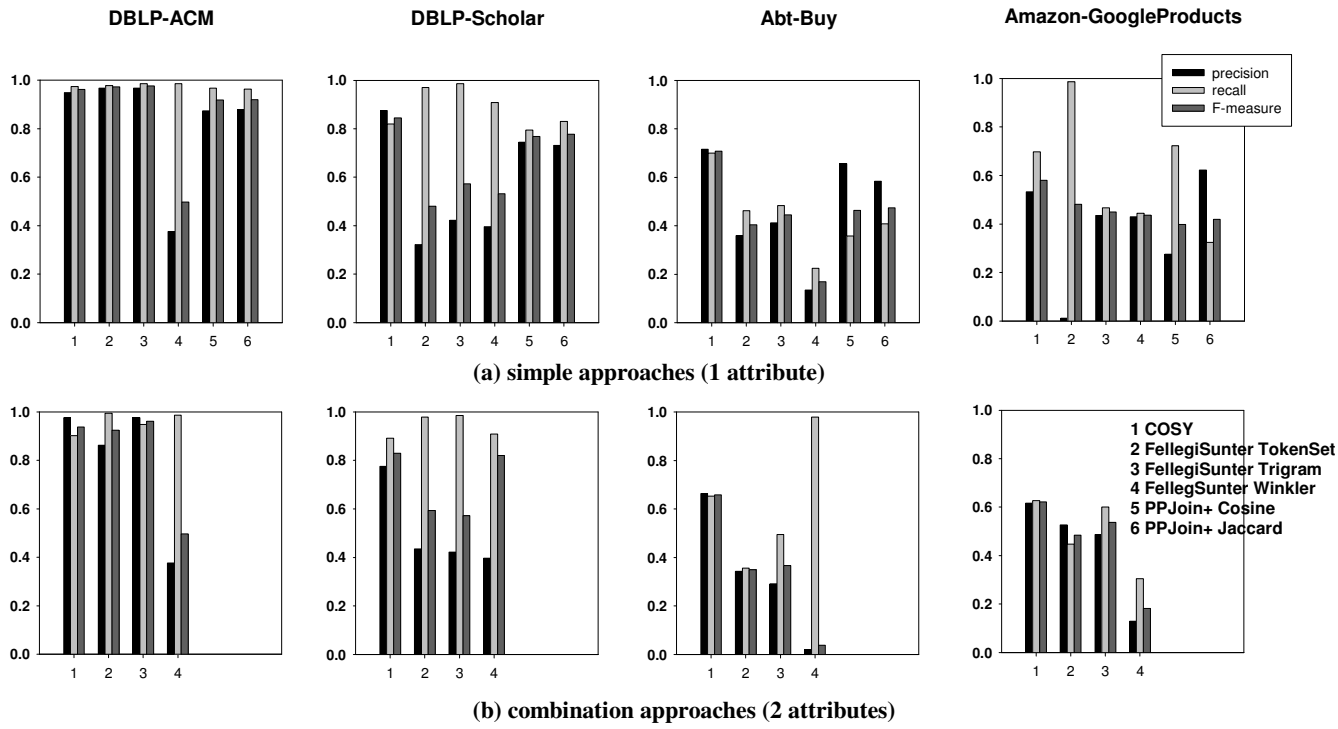


Figure 2: Performance results for non-learning approaches

range 0 to 0.5 indicating the minimal percentage of both matching and non-matching entity pairs. r=0 corresponds to a random strategy that randomly selects entity pairs with a similarity above the threshold t. For r>0 the number of randomly selected entity pairs is reduced so that either the number of matching or non-matching entity pairs satisfy the ratio restriction. For example, r=0.4 guarantees that at least 40% of all training pairs are either matching or non-matching, i.e., at most 60% are non-matching or matching. By ensuring a minimum number of matching/non-matching pairs the ratio approach aims at enhancing the discriminative value of the training data for learning effective match strategies. We have extensively evaluated the Ratio training selection approach and found that setting r=0.4 and t=0.4 with TFIDF is a reliable and effective default configuration. Our evaluation for learning-based matching will thus be based on this configuration.

3. EVALUATION

We first describe the datasets for the four real-life match tasks. In the main part of this section we present and discuss the obtained evaluation results for six non-learning and 15 learning match approaches. In the evaluation we will consider both match quality and runtime efficiency. For match quality we evaluate the usual measures precision, recall, and F-measure.

3.1 Datasets

We consider four match tasks of two application domains (bibliographic and e-commerce data entities). Table 1 provides some statistics on these tasks which are named after the involved web sources. The number of entities per source ranges from about 1,100 to more than 64,000; the size of the Cartesian product for the four tasks ranges from about 1.2 million (Abt-Buy task) to 168.1 million (DBLP-Scholar) entity pairs. We use a fixed blocking strategy for all experiments and evaluated systems to guarantee equal effectiveness and efficiency of the blocking step. Thus blocking is not subject to our evaluation. The blocking strategy employs Trigram on a low string similarity threshold to reduce the search space to the numbers shown in Table 1 (up to 607,000 pairs). To investigate the scalability of the match approaches we evaluate the match runtimes not only on the blocking output but also on the full Cartesian product. To determine the match quality we further created the perfect match results with the cardinalities as shown in Table 1. Selected attributes of the seven data sources are also listed. In this evaluation we focus on matching on the first or the first two attributes listed since they turned out to be most suited for the respective match tasks.

The match tasks were chosen to represent a spectrum of different data characteristics and difficulty levels. The first task is expected to be of low difficulty as it deals with publication entities from two well-structured bibliographic data sources (DBLP, ACM digital library) that are at least partially under manual curation. The selected DBLP and ACM entities cover the same sets of computer science conferences and journals. The second match task requires matching DBLP publications with publications from the entity search engine Google Scholar (Scholar). Scholar automatically extracts its publication entities from full-text documents crawled from the web. This data has many quality problems, in particular duplicate publications, heterogeneous representations of author lists or venue names, misspellings, and extraction errors. To obtain the Scholar data we sent numerous queries on the publication title and venue names and stored the combined query results as our evaluation dataset. The perfect match result was determined manually. We have used the bibliographic data sets already in previous work [31], [21].

Table 2: Execution times (in seconds) for non-learning approaches

1 attribute	DBLP-ACM		DBLP-Scholar		Abt-Buy		Amazon-GoogleProducts	
	blocked	Cartesian	blocked	Cartesian	blocked	Cartesian	blocked	Cartesian
COSY	1	1.6	8.8	224.7	2.7	5.8	6.5	9.6
FellegiSunter TokenSet	2.1	170	10.2	64,057	2.5	17.2	4.6	84
FellegiSunter Trigram	2.5	655	44.7	243,060	25	105	34.1	320
FellegiSunter Winkler	5.7	1,601	164.6	277,200	53.5	364	96.2	1,065
PPJoin+ Cosine	0.4	0.9	3.4	6.9	0.6	2.5	0.5	0.9
PPJoin+ Jaccard	0.4	0.6	3.5	7	0.6	2.5	0.5	0.9
2 attributes								
COSY	35	56	17	434	44	94	28	41
FellegiSunter TokenSet	3	429	17.8	108,896	5.9	43	44	709
FellegiSunter Trigram	3.1	1,512	116	>500,000	58	635	1,940	16,620
FellegiSunter Winkler	7.4	3,602	341	>500,000	135	970	2,833	20,760

The e-commerce tasks deal with sets of related product entities from the online retailers Abt.com, Buy.com (Abt-Buy task), Amazon.com and the product search service of Google accessible through the Google Base Data API (Amazon-GoogleProducts task). In order to obtain the perfect match result we included only product entities with a valid UPC (Universal Product Code) in our datasets which allows a unique identification of a product. Of course, the match strategies to be evaluated could not make use of these UPCs but only of the attributes listed in Table 1 (especially product name and description). This is because in reality many websites do not provide the UPC information so that entity matching cannot rely on these in general. The Abt, Buy, and Amazon datasets were created by selecting products from predefined categories. Based on the Amazon products, the GoogleProducts dataset were generated by sending queries on the product name.

3.2 Evaluation results

We first present match quality and runtime results separately for non-learning and learning-based approaches. Afterwards we briefly compare the two kinds of matchers with each other. The runtime results are determined for a HP Z400 workstation with 2.66 GHz Intel Quad-Core Processor W3520 and 4GB of RAM running 64-bit Windows 7. The evaluated match approaches are implemented in different languages: PPJoin+ is implemented in C++, MARLIN in Java and FEBRL in Python.

3.2.1 Non-learning approaches

Figure 2 shows the match quality (precision, recall, F-measure) results for the four real world match tasks achieved with different non learning approaches. The upper half shows the results for approaches operating on just a single attribute, namely the first attribute listed in Table 1 (publication title for the bibliographic tasks, product name for the e-commerce tasks). The lower half shows the results for approaches combining the similarity for two attributes (the first two attributes listed in Table 1). In both cases we optimized the threshold for the final match decision while all other parameters of the approach were kept constant. Optimization was done with the GradientDescent approach on a test set of 500 object pairs for each match task. For the FellegiSunter approach from the FEBRL framework we considered three different similarity measures, namely Winkler, TokenSet, and Trigram. FEBRL's FellegiSunter approach sums the logarithms of the single similarities. For the COSY approach it is not clear how similarities are combined.

All simple and combined approaches could effectively solve the simple bibliographic match tasks DBLP-ACM (F-measure> 91%), except for the FellegiSunter approach with the Winkler measure which did not even reach an F-measure of 50% because it suffers from a very low precision. The e-commerce match tasks turned out to be much more challenging so that no approach could achieve an F-measure of more than 62% (Amazon-GoogleProducts) or 70% (Abt-Buy). The COSY approach is the top or among the top performing approaches for all match tasks. From the FellegiSunter approaches the configuration using Trigram performed best for all match tasks.

Using two attributes (first two of Table 1) is not always more effective than using one attribute because it is difficult to find a good similarity combination. All approaches become worse for the easy bibliographic match task DBLP-ACM. COSY becomes worse for DBLP-ACM, DBLP-Scholar, and ABT-BUY. PPJoin+ could not be evaluated on two attributes because no combination approach is provided.

Table 2 lists the execution times for the considered non-learning approaches for the blocked input as well as the Cartesian product of the considered match tasks. The table shows significant differences between the approaches already for the blocked input. The evaluation of the Cartesian product tests the scalability and leads to huge differences. PPJoin+ and COSY achieved very fast execution times and could even achieve acceptable run times for the Cartesian product. PPJoin+ implements an intelligent pruning of the search space and is uniformly the fastest approach for all

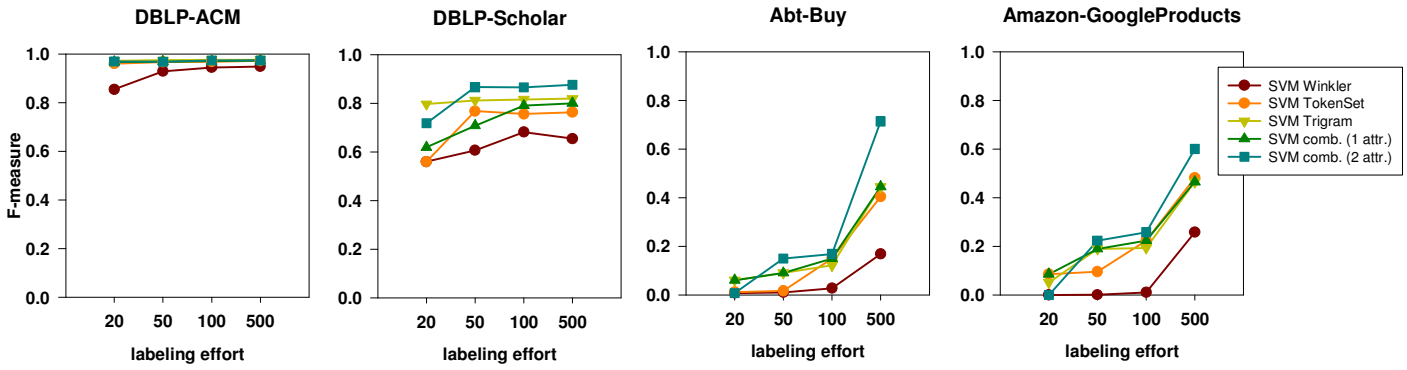
match tasks with execution times between less than a second to at most seven seconds. The small increase of at most a factor of 2 for evaluating the Cartesian product proves the excellent scalability of PPJoin+. In this respect it also outperforms COSY that noticeably slows down for the Cartesian product evaluation of DBLP-Scholar (almost 4 minutes vs. 9 seconds for the blocked input).

The considered FEBRL approaches were mostly much slower than COSY and PPJoin+, on the Cartesian products by orders of magnitude. This may be influenced by the Python-based implementation of FEBRL. FellegiSunter using the Winkler similarity turned out to be not only the least effective but also by far the slowest of all non-learning match approaches. On the blocked input, FEBRL with tokenset similarity is almost as fast as COSY.

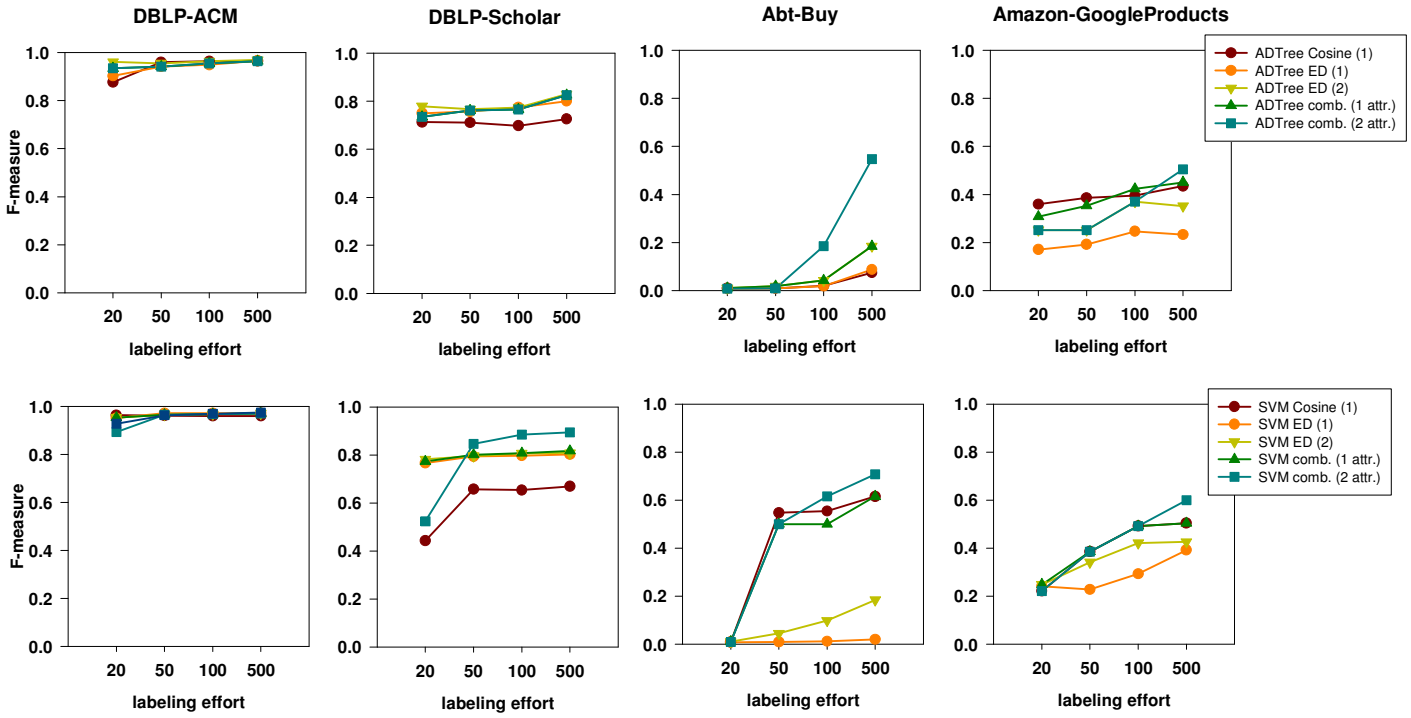
3.2.2 Learning-based approaches

Figure 3 shows the F-measure results for the four real-world match tasks achieved with different learning-based approaches from FEBRL and MARLIN and different labeling efforts (x-axis). The labeling effort varies between 20 and 500 entity pairs, i.e., we consider only comparatively small training sizes and thus a limited amount of labeling effort. The F-measure results are averaged over 10 runs.

All results in Figure 3 refer to matching on the first or the first two attributes listed in Table 1 (publication title and authors for the bibliographic tasks, product name and product description for the e-commerce tasks) with different similarity functions. Figure 3a shows the results for the SVM learner of FEBRL that was applied for the same three similarity functions (TokenSet, Trigram, and Winkler) as for the non-learning case. In addition we



(a) FEBRL



(b) MARLIN

Figure 3: Evaluation results for learning-based approaches

Table 3: Execution times (in seconds) for learning-based approaches

		DBLP-ACM		DBLP-Scholar		Abt-Buy		Amazon-GoogleProducts	
		blocked	Cartesian	blocked	Cartesian	blocked	Cartesian	blocked	Cartesian
FEBRL	SVM TokenSet	3	244	20.0	249,364	8	23	14	124
	SVM Trigram	5	859	79.0	250,920	25	127	46	415
	SVM Winkler	8	2,022	196.5	295,800	62	409	110	1,225
	SVM comb. (1 attr.)	13	2,400	275	>500,000	83	590	154	1,481
	SVM comb. (2 attr.)	99	4,320	482	>500,000	232	1,364	196	36,090
MARLIN	ADTree ED (1)	3	329	76	10,090	22	64	41	161
	ADTree ED (2)	5	582	96	17,427	37	119	57	244
	ADTree Cosine (1)	5	157	71	301	1	89	2	98
	ADTree comb. (1 attr.).	7	951	104	28,476	40	340	95	373
	ADTree comb. (2 attr.)	12	1,553	324	46,501	551	3,456	10,299	41,615
	SVM ED (1)	5	633	117	257,982	28	231	66	333
	SVM ED (2)	7	979	146	445,575	44	192	80	465
	SVM Cosine (1)	4	267	41	7,696	10	143	26	186
	SVM comb. (1 attr.)	9	1,336	157	>600,000	68	552	127	498
	SVM comb. (2 attr.)	20	2,196	375	>900,000	324	3,747	13,768	55,632

use the SVM for two combined match strategies using all three similarity measures either on one or on two attributes. Figure 3b shows the results for MARLIN separated by the employed learner, first for MARLIN's decision tree implementation ADTree followed by the SVM results. For both learners we applied the two similarity measures Edit Distance and Cosine. EditDistance was used in the single-step as well as the two-step learning approach. Cosine was just applied in the single-step approach as it has limitations in the two-step implementation as mentioned by the authors in [4]. We also tested combined match strategies using the two similarity measures either on one or on two attributes for single-step learning. In total, 15 different learning-based approaches are considered.

For the easy bibliographic match task DBLP-ACM, we observe that both FEBRL and MARLIN are able to achieve stable results already for very small training sizes of 20 labeled entity pairs with all evaluated approaches. For the more challenging bibliographic match task DBLP-Scholar, for both FEBRL and MARLIN the SVM strategies combining several matchers on two attributes perform best and achieve F-measure results of 88-89%. The best one-attribute strategies are the combined SVM approaches and SVM using trigram (for FEBRL) or EditDistance (MARLIN). All approaches have substantial difficulties with the e-commerce match tasks, especially for training sizes smaller than 500 entity pairs. The best match quality is always achieved for the combined strategies using all similarity measures on two attributes, followed by the combined approach on one attribute. This underlines that the learners are able to effectively find a combination of several matchers. The decision tree learner of MARLIN is mostly inferior to the SVM-based results. The SVM learner of MARLIN performs slightly better than the one of FEBRL for smaller training sizes. However, for 500 training pairs both SVM learners perform similarly well and achieve a top F-measure of about 71% for Abt-Buy and (only) 60% for Amazon-GoogleProducts. From the single similarity approaches FEBRL with trigram and MARLIN with cosine similarity performed best for the e-commerce tasks. For MARLIN, the 2-step learning for Edit-Distance was always better than the 1-step approach but still too ineffective for the e-commerce tasks. Here the rather long product names and product descriptions tend to favor token-based similarity measures such as cosine, trigram, or the unsupported TF/IDF similarity.

There are huge differences between the approaches regarding execution time as can be seen in Table 3. In general, the execution times for the considered learning-based approaches are significantly worse than for the non-learning approaches. Nearly all learning-based approaches do not scale with larger input sets and are unable to match sufficiently fast on the Cartesian product. For the largest match task DBLP-Scholar execution times of hours to days are needed, the most effective combined approaches exceeded our limit of 500,000 seconds. On the blocked datasets, the approach with the fastest execution time for all match tasks is the FEBRL approach with the TokenSet Cosine measure. The combined match approaches on two attributes take the longest time for blocking, too. They are more than a factor 2 slower than the other learning-based approaches and (except for DBLP-ACM) requires execution times in the order of minutes to hours.

3.2.3 Non-learning vs. learning-based

Table 4 shows a brief summary of the maximum F-measure results achieved for each of the considered non-learning as well as learning-based approaches. For three of four tasks the commercial

Table 4: Summary of evaluation results (F-measure in %, top values are underlined)

	DBLP-ACM		DBLP-Scholar		Abt-Buy		Amazon-GoogleProducts	
	1 attr	2 attr	1 attr	2 attr	1 attr	2 attr	1 attr	2 attr
COSY	96.2	93.8	84.5	82.9	70.7	65.8	62.1	62.2
FEBRL FellegiSunter	97.6	96.2	57.2	81.9	44.5	36.7	48.4	53.8
PPJoin+	91.9	-	77.8	-	47.4	-	41.9	-
FEBRL SVM comb.	97.3	97.6	81.9	87.6	44.5	71.3	46.5	60.1
MARLIN ADTree comb	96.4	96.4	82.6	82.9	18.4	54.8	45.0	50.5
MARLIN SVM comb.	96.4	97.4	82.6	89.4	54.8	70.8	50.5	59.9

COSY approach performs best for matching on one attribute. However for two match tasks its quality degrades when using two attributes. The learning-based approaches, on the other hand, always improve for matching on two attributes compared to only one attribute underlining their potential to effectively combine different match criteria. SVM learning was most effective and the FEBRL and MARLIN implementations perform similarly well for training size 500. They achieve the top F-measure for three of the four match tasks for matching on two attributes. The learning-based approaches from FEBRL perform better than the non-learning FEBRL (FellegiSunter) approach, especially when considering two attributes. The good quality of the learning-based approaches on two attributes comes at the expense of significantly higher execution times. With a single matcher on just one attribute the learning-based approaches could not exploit their potential to combine several matchers and thus turned out to be inferior to the non-learning approaches considering both match quality and execution times.

The relatively low match quality for the e-commerce task asks for further improvements, e.g., by considering additional similarity measures such as TFIDF and/or further attributes and spending more training effort on learning.

4. RELATED WORK

There has been a large body of research on entity resolution and its variations. Recent surveys include [2], [20], and [24]. Most previous studies used a single match approach like threshold-based attribute matching (similarity join [5]), clustering [26], or context-based matching [1], [12], [33].

Most published entity resolution evaluations also focus on individual approaches and use diverse methodologies, measures, and test problems making it difficult to assess the quality of each approach, not to mention their comparative effectiveness and efficiency. There have been few attempts for more comparative evaluations, e.g., comparative evaluations of different string similarity metrics [11], [19], blocking approaches [3], [14], and clustering algorithms [17]. Standardized benchmarks for object matching are useful for comparative evaluations; first proposals exist [28], [32] but have not yet been implemented or applied.

Existing matching frameworks such as FEBRL and MARLIN have been used in several evaluation scenarios for non-learning matchers. The authors of [8] present a comparison of FEBRL's string similarity functions for personal name data. The evaluation results demonstrate that there is no single best name matching approach and the type of personal name data has to be considered when selecting a matching technique.

Blocking techniques as provided by matching frameworks are also subject to comparative evaluation. For example, [27] compares a learned blocking scheme to a hand-crafted blocking strategy implemented in MARLIN. The results indicate that learned blocking schemes can achieve a significant higher reduction rate by a comparable pairs completeness.

The decision model is a crucial aspect for match techniques that combine different similarity measures and, thus, several approaches have been compared individually to the techniques implemented in FEBRL and MARLIN. A matching approach based on genetic programming is presented in [5]. The automatically generated similarity function can improve the match quality in comparison to FEBRL's FellegiSunter method. [15] introduces an enhanced clustering-based decision model of entity resolution. The comparison to the probabilistic decision model of FEBRL shows that it can achieve similar accuracy but with smaller training data. Finally, the authors of [7] compare their programmatic matching techniques to the SVM implementation of MARLIN. They report that the recall values of their operator trees are comparable to that of the SVM for their evaluation settings. Unfortunately, without considering further quality measures such as precision or F-measure and comparable evaluation settings the generality of such findings remains open.

Most previous evaluations of learning-based approaches have provided only limited information on how training examples where acquired and how many were necessary to achieve the stated results making it difficult to judge whether good results were due to the approach or clever (time-consuming) manual training data selection. The authors of [5] present a first study on evaluation and training-set construction for training-based approaches. We present a study of different generic methods for automatically selecting training data to combine and configure several matching techniques in [21].

In [22] we used the same test data sets as in this paper to evaluate our own training-based approaches in comparison with COSY. For all match tasks the learning-based approaches improved F-

measure compared to COSY and the results reported here. The learning-based approaches in [22] combine the results of eight matchers (four similarity measures on two attributes); the best match quality was achieved by a multi-learner approach combining the results of several learners.

5. SUMMARY AND OUTLOOK

We presented a comprehensive and comparable evaluation of existing implementations of non-learning as well as learning-based entity resolution approaches on challenging real-world match tasks. Our evaluations reveal big differences regarding match quality and execution times.

It turned out that the commercial implementation COSY is very effective and efficient for matching on one attribute. However, it was not always able to effectively use more than one attribute for improved match quality. The learning-based match strategies using SVM, on the other hand, outperformed the non-learning approaches for the combined usage of several matchers on more than one attribute. While the SVM approaches effectively solve simple bibliographic match tasks with little training, more training is needed for the challenging e-commerce tasks (500 training pairs in our evaluation). Furthermore, the combined learning-based approaches could only be executed on blocked datasets and required the highest execution times of all match strategies.

The best scalability was observed for the very fast single-attribute PPJoin+ implementation which was even faster than COSY and can be applied on the unblocked Cartesian product (execution time of at most 7 s). Hence scalability to large test cases needs to be better addressed in future approaches, especially for learning-based approaches.

The e-commerce tasks turned out to be quite challenging for all approaches and could not be effectively solved. More sophisticated methods are needed there.

6. ACKNOWLEDGEMENTS

We thank Peter Christen, Mikhal Bilenko, and Wei Wang for kindly making the evaluated approaches available to us and giving us valuable feedback. We also thank the anonymous reviewers for their helpful comments.

7. REFERENCES

[1] Ananthakrishna, R., Chaudhuri, S., and Ganti, V.: Eliminating Fuzzy Duplicates in Data Warehouses. In *Proc. of VLDB*, 2002

[2] Batini, C., and Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques, Data-Centric Systems and Applications, Springer, 2006

[3] Baxter, R., Christen, P, and Churches, T.: A comparison of fast blocking methods for record linkage. In *Proc of ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003

[4] Bilenko, M. and Mooney, R. J.: Adaptive duplicate detection using learnable string similarity measures. In *Proc. of ACM SIGKDD*, 2003

[5] Bilenko, M. and Mooney, R. J.: On Evaluation and Training-Set Construction for Duplicate Detection. In *Proc. of Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003

[6] de Carvalho, M. G., Gonçalves, M. A., Laender, A. H., and da Silva, A. S.: Learning to deduplicate. In *Proc. of JCDL*, 2006

[7] Chaudhuri, S., Chen, B.-C., Ganti,V., and Kaushik, R.: Example-driven design of efficient record matching queries. In *Proc. of VLDB*, 2007

[8] Christen, P.: A Comparison of Personal Name Matching: Techniques and Practical Issues. *Technical Report*, Australian National University, 2006

[9] Christen, P.: FEBRL: a freely available record linkage system with a graphical user interface. In *Proc. of HDKM*, 2008

[10] Cohen, W. W., Kautz, H. A., and McAllester, D. A.: Hardening soft information sources. In *Proc. of Workshop on Information Quality in Information Systems (IQIS)*, 2005

[11] Cohen, W. W., Ravikumar, P., and Fienberg, S. E.: A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proc. of Workshop on Information Integration on the Web (IIWeb)*, 2003

[12] Culotta, A., and McCallum, A.: Joint deduplication of multiple record types in relational data. In *Proc. of CIKM*, 2005

[13] Dong, X., Halevy, A., and Madhavan, J.: Reference reconciliation in complex information spaces. In *Proc. of ACM SIGMOD*, 2005

[14] Elfeky, M. G., Elmagarmid, A.K., and Verykios, V.S.: TAILOR: A Record Linkage Tool Box. In *Proc. of ICDE*, 2002

[15] Fellegi, I. P., and Sunter, A. B.: A theory for record linkage. Journal of the American Statistical Association 64 (328), 1969

[16] Gu, L., and Baxter, R.: Decision Models for Record Linkage. In *Proc. of AusDM*, 2006

[17] Hassanzadeh, O., Chiang, F., Lee, H. C., and Miller, R. J.: Framework For Evaluating Clustering Algorithms In Duplicate Detection. In *Proc. of VLDB*, 2009

[18] Hernandez, M. A., and Stolfo, S. J.: The Merge/Purge Problem for Large Databases. In *Proc. of ACM SIGMOD*, 1995.

[19] Heuser, C. A., Krieser, F. N., and Orengo, V. M.: SimEval: a tool for evaluating the quality of similarity functions. In *Proc. of Conference on Conceptual Modeling*, 2007

[20] Köpcke, H., and Rahm, E.: Frameworks for Entity Matching: A Comparison. *Data & Knowledge Engineering*, 96(2), 2010

[21] Köpcke, H., and Rahm, E.: Training Selection for Tuning Entity Matching. In *Proc. of QDB/MUD workshop*, 2008

[22] Köpcke, H., Thor, A., and Rahm, E.: Learning-Based Approaches for Matching Web Data Entities. IEEE Internet Computing, pp. 23-31, July/August, 2010

[23] Köpcke, H., Thor, A., and Rahm, E.: Comparative evaluation of entity resolution approaches with FEVER. In *Proc. of VLDB*, 2009 (Demo paper)

[24] Koudas, N., Sarawagi, S., and Srivastava, D.: Record linkage: Similarity measures and algorithms. In *Proc of ACM SIGMOD*, 2006

[25] Lu, Q., and Getoor, L.: Link-based Classification using Labeled and Unlabeled Data. In *Proc of ICML*, 2003

[26] McCallum, A., Nigam, K., and Ungar, L. H.: Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In *Proc. of ACM SIGKDD*, 2000

[27] Michelson, M., and Knoblock, C. A.: Learning blocking schemes for record linkage. In *Proc. of AAAI*, 2006

[28] Neiling, M., Jurk, S., Lenz, H.-J., and Naumann, F.: Object identification quality. In *Proc. of DQCIS*, 2003

[29] Rahm, E., and Do, H.-H.: Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin, 23(4)*, 2000

[30] Singla, P., and Domingos, P.: Object Identification with Attribute-Mediated Dependences. In *Proc. of PKDD*, 2005

[31] Thor, A., and Rahm, E.: MOMA - A Mapping-based Object Matching System. In *Proc. of CIDR*, 2007

[32] Weis, M., Naumann, F., and Brosy, F.: A Duplicate Detection Benchmark for XML (and Relational) Data. In *Proc. of Workshop on Information Quality for Information Systems (IQIS)*, 2006

[33] Weis, N. and Naumann, F.: DogmatiX tracks down Duplicated in XML. In *Proc. of ACM SIGMOD*, 2005

[34] Xiao, C., Wang, W., Lin, X., and Yu, J. X.: Efficient Similarity Joins for Near Duplicate Detection. In *Proc. of WWW*, 2008