CONTENTS

Automated Machine Learning for Entity Matching Tasks

Matteo Paganelli, Francesco Del Buono, Francesco Guerra, Marco Pevarello, Maurizio Vincini
firstname.lastname@unimore.it
University of Modena and Reggio Emilia
Modena, Italy

ABSTRACT

The paper studies the application of automated machine learning approaches (AutoML) for addressing the problem of Entity Matching (EM). This would make the existing, highly effective, Machine Learning (ML) and Deep Learning based approaches for EM usable also by non-expert users, who do not have the expertise to train and tune such complex systems. Our experiments show that the direct application of AutoML systems to this scenario does not provide high quality results. To address this issue, we introduce a new component, the *EM adapter*, to be pipelined with standard AutoML systems, that preprocesses the EM datasets to make them usable by automated approaches. The experimental evaluation shows that our proposal obtains the same effectiveness as the state-of-the-art EM systems, but it does not require any skill on ML to tune it.

1 INTRODUCTION

Machine Learning (ML) has significantly advanced over the past ten years [1]. On one side, the research on Big Data let emerge new challenges and made available scenarios and datasets where to experiment and improve ML techniques. On the other side, the increase of computer processing power, thanks in particular to the use of graphic processing units, enabled ML approaches running in commodity hardware. This led to the development of new ML algorithms and their implementations through frameworks and libraries is extensive and growing [17]. Thus the ML technology moved from an R&D phase, for the exclusive use of specialized laboratories, to a mature phase where it can be adopted in business applications.

Mature technologies have to be easy to use for both non-experts and professionals. One of the main bottlenecks towards a large use of the ML technology is related to the configuration of the systems, where experts are typically needed to set the large number of hyper-parameters. Furthermore, the selection of the algorithm that best performs in a given ML task is based on an experimental evaluation in which the performances of competing approaches are compared. This requires a time-consuming and expensive iterative process in which multiple alternative solutions are tested until an optimal result is achieved.

To address these issues, automated machine learning (AutoML) tools have been proposed. These are user-friendly and easy-to-use systems that provide a unified interface for the automatic selection of the most appropriate ML model/algorithm for a given task and its automatic configuration. Some examples are Auto-WEKA [12], AutoSklearn [9], AutoGluon [8], Auto-Keras [11], H20 AutoML [10], and many other.

This paper analyzes the application of AutoML systems to Entity Matching (EM), i.e. the task of identifying which records in a dataset refer to the same real-world entity [5]. Applications addressing EM tasks are becoming mature technologies and AutoML can promote this maturation for three main reasons. First of all, the state of the art EM systems are based on complex and advanced neural architectures [2, 7, 13, 16, 25] which require a non-trivial configuration process performed by expert users. Therefore, AutoML can make the application of EM techniques possible also to less experienced users. Secondly, the implementation of an EM system based on ML is expensive since it is performed through a time-consuming process that requires the active involvement of domain experts for evaluating the model. Introducing a form of automation would reduce the cost and time for the model deployment. Finally, in business scenarios where annotating data for the training process is costly and the performance evaluation is a critical task, the results obtained with an AutoML system represent a low-cost baseline.

Our experiments showed that the direct application of AutoML systems to the EM task was not effective. One of the main reasons is that such systems have been designed to solve generic ML tasks. Therefore, they are ineffective in dealing with the peculiarities of EM, characterized by an "unusual" data format, where each record is a description of pairs of entities, and there is a high imbalance between the “match" and “non-match" classes to predict.

For this reason, we introduce in this paper a software component, the *EM adapter*, encoding the datasets used in the EM task into a numerical form that make them to be effectively processed by AutoML systems. The *EM adapters* rely on the most recent transformer architectures, such as BERT and variants [6, 14, 19, 24], whose out of the box application to the EM problem has proved to be particularly effective [2]. We show in our experiments that *EM adapters* pipelined with AutoML systems are able to ensure quality performance comparable with the one of EM approaches parametrized by ML experts, thus making possible the use of complex ML techniques to non-expert users.

Summarizing, the main contributions of our paper are:

- An analysis about the application of AutoML systems to the EM task;
- The design of *EM adapters*, components based on pre-trained transformer architectures, to be pipelined with AutoML systems for making them able to effectively perform the EM task;
- An extensive experimentation (including a comparison with other state of the art EM tools) to evaluate the effectiveness of our proposal in different scenarios (reduced training times and different types of data).

The rest of the paper is organized as follows. Section 2 introduces the state-of-the-art techniques developed in the AutoML and EM fields. Section 3 describes the main features of the *EM adapters*, the components we developed for making AutoML systems addressing EM tasks, and Section 4 presents the implementation we developed. Our approach has been experimented as described in Section 5. Finally, in Section 6 we sketched out some conclusions and future work.

2 RELATED WORK

Entity Matching. Despite the effort put by the research community for more than thirty years, EM is still an open challenge. Several techniques have been proposed: they range from rules-based approaches [18, 20, 22] to ML models, that conceive EM as a binary classification problem [4] applied on datasets whose records describe pairs of entities. Recently, Deep Learning (DL) approaches (DeepER [7] and DeepMatcher [16] are the first proposals) have proved to be very effective. They suffer from two main problems: 1) the need for a significant amount of labeled data for their training and 2) a non-trivial configuration. To address the first problem, approaches based on transfer learning and fine-tuning techniques relying on architectures pre-trained on large generalist corpus have been proposed [2, 13, 25]. [25] presents a transfer learning approach to EM leveraging on pre-trained models from large-scale, production knowledge bases. [13] applies a fine-tuning process based on a pre-trained Bert architecture [6] using a limited number of labeled data, whose performance is further improved through the exploitation of data augmentation techniques. An analogous approach is described in [2], in which a larger collection of transformer architectures (Bert, XLNet [24], DistilBERT [19] and RoBERTa [14]) are applied to the EM problem.

Although these systems address the problem of the need for a significant number of labeled data, they require specific skills in modifying, training, and configuring complex neural architectures. In this work, we aim to make an EM architecture accessible also to non-expert users and to generate low-cost baselines for EM tasks.

AutoML AutoML is a recent research topic and consists in the automatic identification of the most effective algorithms to make ML inference requiring minimal human intervention and exploiting a limited computational budget (e.g. in terms of use of CPU and RAM).

The first definition of this problem is given in [12], where AutoML is formalized as a *Combined Algorithm Selection and Hyperparameter optimization (CASH)* problem. The solution proposed in this work is Auto-WEKA: a system designed to automatically select and configure, using methods based on Bayesian optimization, the classification models available within the WEKA[1] ML library. Starting with this pioneering work, numerous AutoML systems have been developed. Among them, AutoSklearn [9] combines a meta-learning technique, to boost a Bayesian optimization, with an ensemble method applied to the models selected in the previous step. AutoGluon [8] uses ensembling and multi-layer stacking techniques to combine multiple models (such as LightGBM boosted trees, CatBoost boosted trees, Random Forests, Extremely Randomized Trees, and k-Nearest Neighbors). In the context of neural networks, the main exponent is Auto-Keras [11], which applies Bayesian optimization in the search for the most efficient neural architecture (the so-called *Neural Architecture Search - NAS*). Another promising AutoML system is H2OAutoML [10], which, in place of Bayesian optimization, uses a combination of fast random search with ensemble staking techniques. Finally, a comparison of state-of-the-art approaches is provided in [21].

To the best of our knowledge, no study has been proposed for the application of AutoML systems to the EM task.

[1]https://www.cs.waikato.ac.nz/~ml/weka/index.html

3 DESIGNING AUTOML FOR EM TASKS

AutoML systems could largely support the application of ML and DL approaches to EM tasks by selecting the best classification models for given datasets and providing the tuning of the parameters. Nevertheless, our experiments (partially reported in Section 5) showed that AutoML systems are not effective in this scenario.

To investigate the reasons for such behavior, we considered AutoML systems as black-box tools and we analyzed if there are peculiarities that make EM, conceived as ML task, a hard issue for these approaches. Firstly, we observed that the datasets used for representing EM tasks describe pairs of entities. The insight that an ML model can learn from them is mainly obtained by comparing the values assumed by pairs of attributes describing the same feature in different entities. Since an entity is described through several features, and pairs of attributes describing the same features have value distributions clearly close, selecting and tuning an ML model can be a complex task. Secondly, the classification problem is highly imbalanced, due to the very large number of unmatching entities with respect to the matching ones.

In this paper, we address the first issue by developing and experimenting with a component, the *EM adapter*, which provides an encoding of EM datasets that effectively supports the training of AutoML systems. We consider the second issue as future work that we intend to address designing data augmentation techniques for building a more balanced dataset to train the AutoML system.

The existing AutoML systems provide limited pre-processing and feature extraction capabilities. Our approach represents a first attempt towards the development of a new generation of AutoML systems able to operate in specific scenarios thanks to data transformation capabilities. The design of the *EM adapter* was based on a preliminary analysis of the main features implemented in the existing approaches. This allowed us to get insights into the design choices to adopt in the creation of the component.

The existing state-of-the-art approaches for solving EM tasks are typically based on neural architectures, and in particular on recurrent neural networks (RNNs), as DeepER and DeepMatcher. The RNN (e.g., a bi–directional recurring network made up of LSTM cells in DeepER) is trained to encode the pairs of entities, input of the network, into multi-dimensional vectors. The application of some similarity function to these vectors makes them a training dataset for a binary classifier. The systems based on these architectures are highly effective thus demonstrating their ability in managing the particular schemas adopted by datasets describing EM tasks.

Insight#1: RNNs are an effective means to represent in a single multi-dimensional vector the values of the attributes describing the same feature of pairs of entities.

Training models based on RNN requires large datasets, thus preventing their use in several business scenarios. To address this issue, approaches have been experimented RNNs trained on large “external" knowledge bases and applied to in-production systems via transfer learning techniques. Transfer learning has been successfully applied to several ML applications, and to tasks

related to the EM as in [25] where the problem was recognizing categories where entity features belong to.

Insight#2: External knowledge-bases can be effectively exploited in ML architectures addressing EM tasks to reduce the need for annotated data.

Transformers [23] have recently gained large popularity in the NLP field. They are deep neural networks that, once trained on a large corpus, are able to learn the semantics of words better than conventional word embedding techniques (e.g., Word2vec [15]). They have also been proved effective in EM tasks [2, 13]. In [13], for example, a fine-tuning process is applied to a pre-trained Bert transformer architecture. The dataset is completely denormalized and, in the resulting text fields, meta-tokens are inserted not to lose the semantics of the schema. In this way, the knowledge through which an ML algorithm learns to discriminate between pairs of matching/non-matching entities is no longer obtained by comparing the values assumed by pairs of attributes, but emerges from the analysis of the record as a whole. A broad experimentation of transformers approaches in the field of EM performed in [2] let emerge two main findings: 1) the transformers can be used for EM out of the box, without the need for a task-specific architecture, and 2) fine-tuning a transformer on an EM task takes relatively little time and requires no particularly capable hardware.

Insight#3: Transformers serialize in numerical vectors the fields of EM datasets describing pairs of entities by exploiting a highly-contextualized analysis of EM data. Their use in EM tasks is a good compromise between simplicity and performance, and their fine-tuning does not require high training times and expensive hardware.

4 IMPLEMENTING AUTOML FOR EM TASKS

The *EM adapter* is the component in charge of computing an encoding of a dataset representing EM to be effectively exploited by an AutoML system. Its functional architecture is based on the insights defined in Section 3 and consists of three main components: the *Tokenizer* which tokenizes the dataset records into one or more token sequences; the *Embedder* which transforms the token sequences into embeddings; and the *Combiner* which generates a single multi-dimensional vector from all embeddings associated the same dataset entry.

Tokenizer. This component transforms an entity pair $(e1, e2)$, described in the records of the EM dataset as a series of attributes $a_{11}, ..., a_{1M}, a_{21}, ..., a_{2M}$, where a_{ij} is the attribute j of the entity e_i, into one or more token sequences. The component defines how to combine the values of the records to obtain tokens that enable the automatic learning of deep relations between attributes describing the same features of different entities. There are three alternative ways for performing this operation: the unstructured, attribute-based, and hybrid tokenization mode. In the unstructured mode, all fields describing the entities are concatenated into a unique sentence and any reference to the dataset schema is lost. In the attribute-based mode, the dataset entry is tokenized at the attribute level and the values of the same attribute for the considered pair of entities are coupled. Through this kind of tokenization, the records in the EM datasets describing pairs of entities are broken down into multiple sub-pairs, one for each attribute. Finally, in the hybrid mode, partial and / or incremental concatenations of the attribute values are performed. A hybrid strategy can for example apply an attribute-level tokenization and then incrementally combine the sub-pairs so that the $i-th$ pair contains the values of the first i attributes, and the last one compares the entire original matching pair. The aforementioned hybrid technique and the attribute-based technique have been experimented in the paper.

Dataset	Type	Datasets	Size	% Match
S-DG	Structured	DBLP-GoogleScholar	28,707	18.63
S-DA		DBLP-ACM	12,363	17.96
S-AG		Amazon-Google	11,460	10.18
S-WA		Walmart-Amazon	10,242	9.39
S-BR		BeerAdvo-RateBeer	450	15.11
S-IA		iTunes-Amazon	539	24.49
S-FZ		Fodors-Zagats	946	11.63
T-AB	Textual	Abt-Buy	9,575	10.74
D-IA	Dirty	iTunes-Amazon	539	24.49
D-DA		DBLP-ACM	12,363	17.96
D-DG		DBLP-GoogleScholar	28,707	18.63
D-WA		Walmart-Amazon	10,242	9.39

Table 1: Magellan Benchmark

Regardless of the specific implementation, the result of this step consists of one or more token sequences for each entry of the original EM dataset.

Embedder. The goal of the *Embedder* is to encode a token sequence into a multi-dimensional vector, i.e., an embedding. The approach usually adopted in the literature starts from a pre-trained word embedding deep learning architecture which is experimented with some token sequences. The embeddings are generally then extracted from these architectures by averaging the last hidden layer for each token, but other techniques have been experimented (the concatenations of the last 4 hidden layers for each token in [6]). In the paper, we experimented five embedders: Bert, DistilBert (DBert), Albert, Roberta and XLNET.

The *Embedder* outputs an embedding for each input token sequence. Recall that, according to the tokenization strategy adopted, an entry in the dataset can generate from one to many embeddings.

Combiner. The embeddings generated from each entry in the EM dataset are summarized in a single multi-dimensional vector by the *Combiner*. The standard approach, experimented in the paper, for performing this task is to calculate an average embedding.

5 EXPERIMENTAL EVALUATION

This section aims to provide an experimental answer to three main questions: 1) How effective are standard AutoML systems in solving EM tasks (Section 5.1); 2) To which extent the performance of AutoML systems applied to the EM task benefit from data preprocessing techniques (Section 5.2); and 3) How much AutoML systems pipelined with the *EM adapters* outperform the current state-of-the-art EM models (Section 5.3).

Datasets. The experiments have been performed against the datasets provided by the Magellan library[2] which are considered as a standard benchmark for the evaluation of EM tasks. In Table 1 we summarize with some statistic measures the 12 datasets analyzed, reporting for each of them the total number of records representing matching entities (fourth column) and the percentage of records associated with a matching label (last column). Each dataset is divided into training, validation, and test sets which were created with 60-20-20 proportions. The code used in the experiments is available at *https://github.com/softlab-unimore/automl-for-em*.

[2]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

	AutoSklearn		AutoGluon		H2OAutoML		DeepMatcher	
	F1	Training time (h)	F1	Training time (h)	F1	Training time (h)	F1	Training time (h)
S-DG	50.65	1.00	**77.85**	4.42	64.74	0.97	94.70	8.50
S-DA	92.79	1.00	**97.62**	2.06	92.51	0.94	98.40	8.50
S-AG	**44.10**	1.00	23.28	1.57	36.88	0.96	69.30	1.50
S-WA	29.28	1.00	19.12	2.48	**31.07**	0.94	66.90	2.17
S-BR	40.00	1.00	0.00	0.07	**43.24**	0.74	72.70	0.08
S-IA	53.33	1.00	50.00	0.13	**59.09**	0.87	88.00	0.25
S-FZ	**100.00**	1.00	71.11	0.10	61.90	0.86	100	0.17
T-AB	26.47	1.00	11.41	1.63	**27.36**	0.94	62.80	3.50
D-IA	**64.00**	1.00	60.87	0.15	62.75	0.87	74.50	0.17
D-DA	54.74	1.00	**89.44**	3.07	67.92	0.94	98.10	4.00
D-DG	46.79	1.00	**69.05**	4.96	43.01	0.97	93.80	8.50
D-WA	25.75	1.00	14.12	2.10	**26.31**	0.97	46.00	2.50

Table 2: Effectiveness of AutoML systems in EM tasks.

5.1 AutoML systems solving EM tasks

The experiment shows the application of standards AutoML systems in solving EM tasks. We selected three well-known approaches, AutoSklearn, AutoGluon, and H2OAutoML. We evaluated their performance as binary classifiers against the chosen datasets. The AutoML systems have been experimented with their default configuration, no parameter tuning has been performed. Only the AutoSklearn system required the application of a data pre-processing step to transform categorical features (which are not managed by this system) into numerical values. We performed this operation via a standard Word2Vec embedding, where the average Word2Vec embedding for each token of non-numeric attributes has been computed and concatenated. The results of this experiment are shown in Table 2, reporting the F1 score and the training time of each AutoML system for each dataset. The last column in the Table shows the scores achieved by DeepMatcher, in the *Hybrid* configuration, which we consider as a baseline[3].

Discussion. The AutoML systems are not effective in solving EM tasks as DeepMatcher. They perform close to DeepMatcher (and with an average F1 score greater than 75%) in two datasets only (S-DA and S-FZ). Even if there are small variations among the performance achieved, there is no AutoML system that clearly outperforms the others: the average F1 scores along all the datasets are close, ranging from 48.66% for AutoGluon, to 51.4% for H2OAutoML, and 52.33% for AutoSklearn. Nevertheless, the time required for training the models was largely varying. H2OAutoML took 1 hour maximum to finalize the training in all the datasets; AutoSklearn limits the training time to one hour by default; the time required for training AutoGluon was larger, more than four hours for the S-DG and D-DG datasets. We performed a further experiment, limiting the training time of AutoGluon to 1 hour. The quality of the results largely decreased and the average F1 score across all datasets dropped to 42.4%.

Lesson Learned. AutoML systems are not competitive when addressing binary classification problems associated with EM tasks.

5.2 AutoML systems pipelined with EM adapters

In this section, we evaluate the effectiveness of AutoML systems coupled with an *EM adapter*. For this reason, we experimented several adapters, obtained by combining possible implementations for their constituting modules, the tokenizer, the embedder and the combiner, introduced in Section 4. In the following, we report the effectiveness measured in adapters implementing attribute-based and hybrid tokenizers, where the best results are obtained. For each kind of tokenizer, five standard transformer architectures, namely Bert, DistilBert (DBert), Albert, RoBERTa and XLNET have been evaluated[4]. For sake of simplicity, we only consider combiners generating the embeddings from the last hidden layer for each token and averaging it with the layers referring to the other tokens. Tables 3 shows the results of this evaluation for AutoSklearn, AutoGluon and H2OAutoML respectively. The scores of the *EM adapters* were grouped according to the tokenization technique (attribute-based vs hybrid). The F1 scores in bold represent the best result per dataset obtained for the category of tokenizer considered. The values in bold and underlined represent the results of the most effective *EM adapter* for the considered dataset.

Discussion. We observe that (1) the *EM adapters* implementing hybrid tokenizers typically obtain the best performance (only in 3/12 datasets, i.e. S-DA, S-WA and S-FZ, the results are worse); (2) the Albert embedder achieves the best results (i.e., *EM adapter* implementing an Albert embedder is the best solution for 7/12 datasets considering the AutoSklearn system and 8/12 considering AutoGluon and H2OAutoML). Finally, Table 4 reports an overall evaluation of the impact of an *EM adapter* as a preprocessing component for an AutoML system in solving EM tasks. For each dataset and AutoML system, the F1 scores obtained in the absence and presence of an *EM adapter* are reported. The "No EM-Adapter" column is the reference column showing the F1 scores obtained by AutoML systems with no *EM adapter*. The remaining columns show the average F1 score (through the 5 transformer architectures considered) obtained by the adapters implementing attribute-based and hybrid tokenizers. Finally, for each AutoML system, the delta column shows the difference between the F1 score obtained with no *EM adapter* and the average of the two versions including the *EM adapters*. The experiments show that adapters significantly improve the effectiveness of AutoML systems in solving EM tasks in almost all datasets (the datasets S-FZ for AutoSklearn and S-DA for AutoGluon show an anomaly result). The average F1 score increases of 24.96%, 28.02% and 23.6% for AutoSklearn, AutoGluon and H2OAutoML respectively.

Lesson Learned. *EM adapters* largely improve the effectiveness of AutoML systems in addressing EM tasks. The experiments were not able to show a clear winner among the approaches tested. This is a positive result since it means that the AutoML technology can benefit from the application of *EM adapters*.

5.3 EM adapters pipelined with AutoML systems vs ad hoc solutions

The aim of this experiment is to evaluate if an AutoML system pipelined with an *EM adapter* can obtain competitive results with respect to other state-of-the-art EM models. For this evaluation, we consider an AutoML system with an *EM adapter* consisting of a hybrid tokenizer and an Albert embedder, whose combination provided the best performances in the previous experiments.

[3]We used the implementation available at https://github.com/anhaidgroup/deepmatcher

[4]No fine-tuning technique was applied in the experiments.

(a) EM-Adapter with AutoSklearn

	Attr-EM-Adapter					Hybrid-EM-Adapter				
	Bert	**DBert**	**Albert**	**Roberta**	**XLNET**	**Bert**	**DBert**	**Albert**	**Roberta**	**XLNET**
S-DG	92.70	90.49	**92.85**	92.11	91.65	**93.76**	91.92	93.56	92.59	92.77
S-DA	**98.10**	95.74	96.54	97.89	96.76	**96.90**	95.73	96.64	96.12	96.22
S-AG	62.11	59.59	**66.67**	60.07	51.38	66.67	62.60	**68.41**	61.25	58.74
S-WA	58.05	55.73	**67.18**	55.21	56.36	56.93	52.36	**62.17**	48.00	46.92
S-BR	66.67	68.75	**73.33**	70.97	66.67	64.52	78.79	**74.29**	**74.29**	68.97
S-IA	88.46	88.89	83.64	80.77	**90.20**	83.64	87.27	85.71	88.46	**96.30**
S-FZ	97.67	97.78	95.24	**100.00**	93.02	**97.67**	**97.67**	95.24	**97.67**	**97.67**
T-AB	58.44	57.08	**66.37**	56.68	54.65	66.97	58.23	**76.92**	66.36	61.01
D-IA	56.52	58.82	**67.92**	51.85	64.52	80.70	77.97	**91.23**	79.25	87.72
D-DA	92.90	91.58	**96.30**	91.71	93.20	**96.58**	95.41	96.36	95.20	96.00
D-DG	86.73	85.63	**90.29**	86.22	86.67	93.00	91.39	**93.12**	92.76	92.25
D-WA	39.82	45.71	**56.02**	39.17	37.99	51.38	51.10	**62.80**	46.23	44.23

(b) EM-Adapter with AutoGluon

	Attr-EM-Adapter					Hybrid-EM-Adapter				
	Bert	**DBert**	**Albert**	**Roberta**	**XLNET**	**Bert**	**DBert**	**Albert**	**Roberta**	**XLNET**
S-DG	93.35	91.88	**93.53**	92.19	92.76	94.30	92.83	**94.37**	93.25	93.56
S-DA	98.19	97.31	96.85	**98.31**	97.64	96.67	96.31	**96.75**	96.32	96.21
S-AG	57.78	50.14	**64.61**	54.04	45.39	58.06	58.55	**64.89**	53.40	55.83
S-WA	54.07	58.13	**67.04**	52.04	52.34	55.46	49.52	**64.67**	47.88	42.99
S-BR	64.29	66.67	**80.00**	63.64	64.29	64.29	64.29	**81.25**	71.43	75.86
S-IA	84.62	**88.46**	85.19	85.19	79.17	86.79	86.79	92.59	86.79	**98.18**
S-FZ	97.67	**100.00**	97.67	97.67	95.24	97.67	**100.00**	95.24	**100.00**	**100.00**
T-AB	58.49	58.72	**70.95**	56.55	56.76	60.18	64.22	**73.97**	63.69	59.71
D-IA	**69.39**	61.90	63.83	58.33	50.00	80.00	77.55	**87.27**	81.48	82.35
D-DA	93.00	93.12	**96.02**	91.05	92.97	96.54	95.58	**96.76**	96.00	95.88
D-DG	88.78	87.78	**91.73**	87.86	88.45	92.51	92.35	93.10	**93.16**	92.58
D-WA	32.47	44.52	**56.89**	33.79	33.55	51.55	49.35	**65.56**	43.29	39.13

(c) EM-Adapter with H2OAutoML

	Attr-EM-Adapter					Hybrid-EM-Adapter				
	Bert	**DBert**	**Albert**	**Roberta**	**XLNET**	**Bert**	**DBert**	**Albert**	**Roberta**	**XLNET**
S-DG	91.52	90.23	**92.25**	90.08	90.86	92.41	91.87	**93.78**	92.29	92.16
S-DA	**98.08**	96.44	96.72	96.57	96.15	96.03	94.45	**96.96**	94.77	95.58
S-AG	55.02	52.61	**61.86**	59.11	52.81	61.96	59.19	**66.54**	60.32	46.51
S-WA	55.07	53.61	**65.49**	48.28	53.22	49.47	47.89	**59.08**	46.53	45.49
S-BR	70.97	69.23	**78.79**	72.00	52.63	66.67	58.33	**86.67**	70.97	74.07
S-IA	84.62	86.79	84.62	**87.27**	82.61	80.70	92.00	90.57	86.79	**94.34**
S-FZ	87.18	92.68	**97.67**	90.00	84.21	92.68	95.24	92.68	**100.00**	**100.00**
T-AB	50.36	50.11	**65.88**	52.58	51.92	56.19	55.76	**70.53**	57.01	55.50
D-IA	51.16	56.60	70.37	58.18	**80.00**	76.67	80.85	80.77	78.43	**82.35**
D-DA	87.02	89.83	**95.02**	88.86	90.18	94.41	95.51	**96.32**	94.55	94.83
D-DG	83.83	84.10	**89.85**	84.48	83.92	89.87	87.53	**92.34**	91.90	91.93
D-WA	33.40	38.99	**55.37**	30.03	33.59	46.60	47.90	**61.06**	42.79	40.96

Table 3: EM-Adapter effectiveness for AutoML systems.

	AutoSklearn				AutoGluon				H2OAutoML			
	No EM-Adapter	**Attr-EM-Adapter**	**Hybrid-EM-Adapter**	**Δ**	**No EM-Adapter**	**Attr-EM-Adapter**	**Hybrid-EM-Adapter**	**Δ**	**No EM-Adapter**	**Attr-EM-Adapter**	**Hybrid-EM-Adapter**	**Δ**
S-DG	50.65	91.96	**92.92**	41.79	77.85	92.74	**93.66**	15.35	64.74	90.99	**92.50**	27.00
S-DA	92.79	**97.01**	96.32	3.87	97.62	**97.66**	96.45	-0.56	92.51	**96.79**	95.56	3.66
S-AG	44.10	59.96	**63.53**	17.65	23.28	54.39	**58.15**	32.98	36.88	56.28	**58.90**	20.71
S-WA	29.28	**58.50**	53.28	26.61	19.12	**56.72**	52.10	35.30	31.07	**55.13**	49.69	21.34
S-BR	40.00	69.28	**72.17**	30.72	0.00	67.77	**71.42**	69.60	43.24	68.72	**71.34**	26.79
S-IA	53.33	86.39	**88.28**	34.00	50.00	84.52	**90.23**	37.38	59.09	85.18	**88.88**	27.94
S-FZ	**100.00**	96.74	97.19	-3.04	71.11	97.65	**98.58**	27.01	61.90	90.35	**96.12**	31.33
T-AB	26.47	58.64	**65.90**	35.80	11.41	60.29	**64.36**	50.92	27.36	54.17	**59.00**	29.23
D-IA	64.00	59.93	**83.37**	7.65	60.87	60.69	**81.73**	10.34	62.75	63.26	**79.81**	8.79
D-DA	54.74	93.14	**95.91**	39.79	89.44	93.23	**96.15**	5.25	67.92	90.18	**95.13**	24.74
D-DG	46.79	87.11	**92.50**	43.01	69.05	88.92	**92.74**	21.78	43.01	85.23	**90.71**	44.96
D-WA	25.75	43.74	**51.15**	21.69	14.12	40.24	**49.78**	30.89	26.31	38.27	**47.86**	16.76

Table 4: Impact of EM-Adapter on AutoML performance. Bold values indicate the best configuration for a specific AutoML system; underlined values the best results for the considered dataset.

This system was then compared with the *Hybrid* variant of DeepMatcher. Table 5 reports the results of this comparison. We performed two experiments, in the first we limited the training time of the AutoML systems to 1 hour. In the second experiment, we set that time to 6 hours. The offset between the average effectiveness of AutoML systems and DeepMatcher is shown for each configuration.

Discussion. AutoML systems limited to 1 hour of training show better effectiveness than DeepMatcher in 5/12 datasets (i.e. S-BR, S-IA, T-AB , D-IA and D-WA, with an average increase of F1 equal to 9%). In the remaining cases, they generate slightly lower results, with an average F1 difference of 3.2%. However, we notice that AutoSklearn used the entire training time budget, but AutoGluon and H2OAutoML took on average a training time of 0.61 h and 0.76 h respectively. Furthermore, DeepMatcher has been trained for less than 1 hour in only 4/12 datasets. If we consider a tolerance threshold of 2%, the EM-adapted AutoML systems are comparable or outperform DeepMatcher in 9/12 datasets. This trend is further confirmed when we limit the training time to 6 hours. The average F1 score increases of 12% in the 5 datasets where the AutoML systems obtain the best results. Assuming as before a 2% tolerance threshold in F1 scores, EM-adapted AutoML systems are comparable or outperform DeepMatcher in 11/12 datasets. Also in this case, only AutoSklearn used the entire time budget for training, while AutoGluon and H2OAutoML took an average of 3.68 hours and 3.24 hours respectively (compared to 2.92 hours of DeepMatcher).

Lesson Learned. AutoML systems pipelined with *EM adapters* perform as or greater than state-of-the-art EM tools.

	DeepMatcher (Hybrid)		EM-Adapted AutoMLs (1h time budget)				EM-Adapted AutoMLs (6h time budget)			
	F1	Time (h)	AutoSklearn	AutoGluon	H2OAutoML	Δ	AutoSklearn	AutoGluon	H2OAutoML	Δ
S-DG	94.70	8.50	93.56	92.98	**93.78**	-1.26	94.02	**94.16**	93.82	-0.70
S-DA	98.40	3.75	96.64	96.75	**96.96**	-1.62	**97.08**	96.85	**97.08**	-1.40
S-AG	69.30	1.50	**68.41**	59.03	66.54	-4.64	68.41	64.53	**69.24**	-1.90
S-WA	66.90	2.17	**62.17**	58.01	59.08	-7.15	65.16	**66.12**	63.50	-1.97
S-BR	72.70	0.08	74.29	81.25	**86.67**	8.03	76.47	81.25	**86.67**	8.76
S-IA	88.00	0.25	85.71	**92.59**	90.57	1.62	91.23	92.59	**94.12**	4.65
S-FZ	100.00	0.17	**95.24**	**95.24**	92.68	-5.61	**97.67**	**97.67**	95.24	-3.14
T-AB	62.80	3.50	**76.92**	69.83	70.53	9.63	76.92	76.88	**77.06**	14.16
D-IA	74.50	0.17	**91.23**	87.27	80.77	11.92	**91.23**	**91.23**	82.35	13.77
D-DA	98.10	4.00	96.36	**97.12**	96.32	-1.50	96.36	97.12	**97.33**	-1.16
D-DG	93.80	8.50	**93.12**	92.59	92.34	-1.12	**93.53**	93.38	93.21	-0.43
D-WA	46.00	2.50	**62.80**	57.05	61.06	14.30	**67.77**	62.50	65.44	19.23

Table 5: Effectiveness of EM-Adapter with AutoML systems compared to DeepMatcher. Values in bold indicate the best configuration for a training time budget configuration.

6 CONCLUSION

The adoption of AutoML systems would make machine learning and deep learning based approaches for addressing EM tasks usable also for non-expert people. The direct application of AutoML systems to the EM problem is not possible. The reason is mainly due to the schema adopted by the datasets representing EM tasks whose records encode pairs of entities, and to the classification problem which is highly imbalanced. In this paper, we address the first problem, by introducing the *EM adapter* component which transforms the records of datasets representing entity pairs into a form which is effectively processable by AutoML systems. Our experiments show that this approach achieves a performance similar to the one of EM-task specific systems (but it does not require expert users to tune it). The future work will try to improve the performance (1) by introducing data augmentation techniques for creating more balanced training datasets for the AutoML systems; (2) by experimenting techniques for improving the embeddings (via "local embeddings" [3], generated taking into account the current dataset, and/or performing a fine-tune of the existing techniques).

ACKNOWLEDGEMENT

This work was partially funded by SBDIO I4.0 (https://www.sbdioi40.it/), an industrial research project funded by the POR FESR Emilia Romagna 2014-2020 as part of the Smart Specialization Strategy (S3).

REFERENCES

[1] Gary Anthes. 2017. Artificial intelligence poised to ride a new wave. *Commun. ACM* 60, 7 (2017), 19–21.
[2] Ursin Brunner and Kurt Stockinger. 2020. Entity Matching with Transformer Architectures - A Step Forward in Data Integration. In *EDBT*. OpenProceedings.org, 463–473.
[3] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *SIGMOD Conference*. ACM, 1335–1349.
[4] Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer.
[5] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2020. An Overview of End-to-End Entity Resolution for Big Data. 53, 6, Article 127 (Dec. 2020), 42 pages. https://doi.org/10.1145/3418896
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.
[7] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proc. VLDB Endow.* 11, 11 (2018), 1454–1467.
[8] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander J. Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *CoRR* abs/2003.06505 (2020).
[9] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2019. Auto-sklearn: Efficient and Robust Automated Machine Learning. In *Automated Machine Learning*. Springer, 113–134.
[10] H2O.ai. 2017. *H2O AutoML*. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html H2O version 3.30.0.1.
[11] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In *KDD*. ACM, 1946–1956.
[12] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. 2019. Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA. In *Automated Machine Learning*. Springer, 81–95.
[13] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (Sept. 2020), 50–60. https://doi.org/10.14778/3421424.3421431
[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).
[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR (Workshop Poster)*.
[16] 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 Conference*. ACM, 19–34.
[17] Giang Nguyen, Stefan Dlugolinsky, Martin Bobák, Viet D. Tran, Álvaro López García, Ignacio Heredia, Peter Malík, and Ladislav Hluchý. 2019. Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artif. Intell. Rev.* 52, 1 (2019), 77–124.
[18] Matteo Paganelli, Paolo Sottovia, Francesco Guerra, and Yannis Velegrakis. 2019. TuneR: Fine Tuning of Rule-based Entity Matchers. In *CIKM*. ACM, 2945–2948.
[19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019).
[20] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing Entity Matching Rules by Examples. *Proc. VLDB Endow.* 11, 2 (2017), 189–202.
[21] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan E. Hines, C. Bayan Bruss, and Reza Farivar. 2019. Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools. In *ICTAI*. IEEE, 1471–1479.
[22] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity Matching: How Similar Is Similar. *Proc. VLDB Endow.* 4, 10 (2011), 622–633.
[23] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP (Demos)*. Association for Computational Linguistics, 38–45.
[24] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*. 5754–5764.
[25] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. In *WWW*. ACM, 2413–2424.

CONTENTS

Automated Machine Learning for Entity Matching Tasks

Matteo Paganelli, Francesco Del Buono, Francesco Guerra, Marco Pevarello, Maurizio Vincini
firstname.lastname@unimore.it
University of Modena and Reggio Emilia
Modena, Italy

ABSTRACT

The paper studies the application of automated machine learning approaches (AutoML) for addressing the problem of Entity Matching (EM). This would make the existing, highly effective, Machine Learning (ML) and Deep Learning based approaches for EM usable also by non-expert users, who do not have the expertise to train and tune such complex systems. Our experiments show that the direct application of AutoML systems to this scenario does not provide high quality results. To address this issue, we introduce a new component, the *EM adapter*, to be pipelined with standard AutoML systems, that preprocesses the EM datasets to make them usable by automated approaches. The experimental evaluation shows that our proposal obtains the same effectiveness as the state-of-the-art EM systems, but it does not require any skill on ML to tune it.

1 INTRODUCTION

Machine Learning (ML) has significantly advanced over the past ten years [1]. On one side, the research on Big Data let emerge new challenges and made available scenarios and datasets where to experiment and improve ML techniques. On the other side, the increase of computer processing power, thanks in particular to the use of graphic processing units, enabled ML approaches running in commodity hardware. This led to the development of new ML algorithms and their implementations through frameworks and libraries is extensive and growing [17]. Thus the ML technology moved from an R&D phase, for the exclusive use of specialized laboratories, to a mature phase where it can be adopted in business applications.

Mature technologies have to be easy to use for both non-experts and professionals. One of the main bottlenecks towards a large use of the ML technology is related to the configuration of the systems, where experts are typically needed to set the large number of hyper-parameters. Furthermore, the selection of the algorithm that best performs in a given ML task is based on an experimental evaluation in which the performances of competing approaches are compared. This requires a time-consuming and expensive iterative process in which multiple alternative solutions are tested until an optimal result is achieved.

To address these issues, automated machine learning (AutoML) tools have been proposed. These are user-friendly and easy-to-use systems that provide a unified interface for the automatic selection of the most appropriate ML model/algorithm for a given task and its automatic configuration. Some examples are Auto-WEKA [12], AutoSklearn [9], AutoGluon [8], Auto-Keras [11], H20 AutoML [10], and many other.

This paper analyzes the application of AutoML systems to Entity Matching (EM), i.e. the task of identifying which records in a dataset refer to the same real-world entity [5]. Applications addressing EM tasks are becoming mature technologies and AutoML can promote this maturation for three main reasons. First of all, the state of the art EM systems are based on complex and advanced neural architectures [2, 7, 13, 16, 25] which require a non-trivial configuration process performed by expert users. Therefore, AutoML can make the application of EM techniques possible also to less experienced users. Secondly, the implementation of an EM system based on ML is expensive since it is performed through a time-consuming process that requires the active involvement of domain experts for evaluating the model. Introducing a form of automation would reduce the cost and time for the model deployment. Finally, in business scenarios where annotating data for the training process is costly and the performance evaluation is a critical task, the results obtained with an AutoML system represent a low-cost baseline.

Our experiments showed that the direct application of AutoML systems to the EM task was not effective. One of the main reasons is that such systems have been designed to solve generic ML tasks. Therefore, they are ineffective in dealing with the peculiarities of EM, characterized by an "unusual" data format, where each record is a description of pairs of entities, and there is a high imbalance between the “match" and “non-match" classes to predict.

For this reason, we introduce in this paper a software component, the *EM adapter*, encoding the datasets used in the EM task into a numerical form that make them to be effectively processed by AutoML systems. The *EM adapters* rely on the most recent transformer architectures, such as BERT and variants [6, 14, 19, 24], whose out of the box application to the EM problem has proved to be particularly effective [2]. We show in our experiments that *EM adapters* pipelined with AutoML systems are able to ensure quality performance comparable with the one of EM approaches parametrized by ML experts, thus making possible the use of complex ML techniques to non-expert users.

Summarizing, the main contributions of our paper are:

- An analysis about the application of AutoML systems to the EM task;
- The design of *EM adapters*, components based on pre-trained transformer architectures, to be pipelined with AutoML systems for making them able to effectively perform the EM task;
- An extensive experimentation (including a comparison with other state of the art EM tools) to evaluate the effectiveness of our proposal in different scenarios (reduced training times and different types of data).

The rest of the paper is organized as follows. Section 2 introduces the state-of-the-art techniques developed in the AutoML and EM fields. Section 3 describes the main features of the *EM adapters*, the components we developed for making AutoML systems addressing EM tasks, and Section 4 presents the implementation we developed. Our approach has been experimented as described in Section 5. Finally, in Section 6 we sketched out some conclusions and future work.

2 RELATED WORK

Entity Matching. Despite the effort put by the research community for more than thirty years, EM is still an open challenge. Several techniques have been proposed: they range from rules-based approaches [18, 20, 22] to ML models, that conceive EM as a binary classification problem [4] applied on datasets whose records describe pairs of entities. Recently, Deep Learning (DL) approaches (DeepER [7] and DeepMatcher [16] are the first proposals) have proved to be very effective. They suffer from two main problems: 1) the need for a significant amount of labeled data for their training and 2) a non-trivial configuration. To address the first problem, approaches based on transfer learning and fine-tuning techniques relying on architectures pre-trained on large generalist corpus have been proposed [2, 13, 25]. [25] presents a transfer learning approach to EM leveraging on pre-trained models from large-scale, production knowledge bases. [13] applies a fine-tuning process based on a pre-trained Bert architecture [6] using a limited number of labeled data, whose performance is further improved through the exploitation of data augmentation techniques. An analogous approach is described in [2], in which a larger collection of transformer architectures (Bert, XLNet [24], DistilBERT [19] and RoBERTa [14]) are applied to the EM problem.

Although these systems address the problem of the need for a significant number of labeled data, they require specific skills in modifying, training, and configuring complex neural architectures. In this work, we aim to make an EM architecture accessible also to non-expert users and to generate low-cost baselines for EM tasks.

AutoML AutoML is a recent research topic and consists in the automatic identification of the most effective algorithms to make ML inference requiring minimal human intervention and exploiting a limited computational budget (e.g. in terms of use of CPU and RAM).

The first definition of this problem is given in [12], where AutoML is formalized as a *Combined Algorithm Selection and Hyperparameter optimization (CASH)* problem. The solution proposed in this work is Auto-WEKA: a system designed to automatically select and configure, using methods based on Bayesian optimization, the classification models available within the WEKA[1] ML library. Starting with this pioneering work, numerous AutoML systems have been developed. Among them, AutoSklearn [9] combines a meta-learning technique, to boost a Bayesian optimization, with an ensemble method applied to the models selected in the previous step. AutoGluon [8] uses ensembling and multi-layer stacking techniques to combine multiple models (such as LightGBM boosted trees, CatBoost boosted trees, Random Forests, Extremely Randomized Trees, and k-Nearest Neighbors). In the context of neural networks, the main exponent is Auto-Keras [11], which applies Bayesian optimization in the search for the most efficient neural architecture (the so-called *Neural Architecture Search - NAS*). Another promising AutoML system is H2OAutoML [10], which, in place of Bayesian optimization, uses a combination of fast random search with ensemble staking techniques. Finally, a comparison of state-of-the-art approaches is provided in [21].

To the best of our knowledge, no study has been proposed for the application of AutoML systems to the EM task.

[1]https://www.cs.waikato.ac.nz/~ml/weka/index.html

3 DESIGNING AUTOML FOR EM TASKS

AutoML systems could largely support the application of ML and DL approaches to EM tasks by selecting the best classification models for given datasets and providing the tuning of the parameters. Nevertheless, our experiments (partially reported in Section 5) showed that AutoML systems are not effective in this scenario.

To investigate the reasons for such behavior, we considered AutoML systems as black-box tools and we analyzed if there are peculiarities that make EM, conceived as ML task, a hard issue for these approaches. Firstly, we observed that the datasets used for representing EM tasks describe pairs of entities. The insight that an ML model can learn from them is mainly obtained by comparing the values assumed by pairs of attributes describing the same feature in different entities. Since an entity is described through several features, and pairs of attributes describing the same features have value distributions clearly close, selecting and tuning an ML model can be a complex task. Secondly, the classification problem is highly imbalanced, due to the very large number of unmatching entities with respect to the matching ones.

In this paper, we address the first issue by developing and experimenting with a component, the *EM adapter*, which provides an encoding of EM datasets that effectively supports the training of AutoML systems. We consider the second issue as future work that we intend to address designing data augmentation techniques for building a more balanced dataset to train the AutoML system.

The existing AutoML systems provide limited pre-processing and feature extraction capabilities. Our approach represents a first attempt towards the development of a new generation of AutoML systems able to operate in specific scenarios thanks to data transformation capabilities. The design of the *EM adapter* was based on a preliminary analysis of the main features implemented in the existing approaches. This allowed us to get insights into the design choices to adopt in the creation of the component.

The existing state-of-the-art approaches for solving EM tasks are typically based on neural architectures, and in particular on recurrent neural networks (RNNs), as DeepER and DeepMatcher. The RNN (e.g., a bi–directional recurring network made up of LSTM cells in DeepER) is trained to encode the pairs of entities, input of the network, into multi-dimensional vectors. The application of some similarity function to these vectors makes them a training dataset for a binary classifier. The systems based on these architectures are highly effective thus demonstrating their ability in managing the particular schemas adopted by datasets describing EM tasks.

Insight#1: RNNs are an effective means to represent in a single multi-dimensional vector the values of the attributes describing the same feature of pairs of entities.

Training models based on RNN requires large datasets, thus preventing their use in several business scenarios. To address this issue, approaches have been experimented RNNs trained on large “external" knowledge bases and applied to in-production systems via transfer learning techniques. Transfer learning has been successfully applied to several ML applications, and to tasks

related to the EM as in [25] where the problem was recognizing categories where entity features belong to.

Insight#2: External knowledge-bases can be effectively exploited in ML architectures addressing EM tasks to reduce the need for annotated data.

Transformers [23] have recently gained large popularity in the NLP field. They are deep neural networks that, once trained on a large corpus, are able to learn the semantics of words better than conventional word embedding techniques (e.g., Word2vec [15]). They have also been proved effective in EM tasks [2, 13]. In [13], for example, a fine-tuning process is applied to a pre-trained Bert transformer architecture. The dataset is completely denormalized and, in the resulting text fields, meta-tokens are inserted not to lose the semantics of the schema. In this way, the knowledge through which an ML algorithm learns to discriminate between pairs of matching/non-matching entities is no longer obtained by comparing the values assumed by pairs of attributes, but emerges from the analysis of the record as a whole. A broad experimentation of transformers approaches in the field of EM performed in [2] let emerge two main findings: 1) the transformers can be used for EM out of the box, without the need for a task-specific architecture, and 2) fine-tuning a transformer on an EM task takes relatively little time and requires no particularly capable hardware.

Insight#3: Transformers serialize in numerical vectors the fields of EM datasets describing pairs of entities by exploiting a highly-contextualized analysis of EM data. Their use in EM tasks is a good compromise between simplicity and performance, and their fine-tuning does not require high training times and expensive hardware.

4 IMPLEMENTING AUTOML FOR EM TASKS

The *EM adapter* is the component in charge of computing an encoding of a dataset representing EM to be effectively exploited by an AutoML system. Its functional architecture is based on the insights defined in Section 3 and consists of three main components: the *Tokenizer* which tokenizes the dataset records into one or more token sequences; the *Embedder* which transforms the token sequences into embeddings; and the *Combiner* which generates a single multi-dimensional vector from all embeddings associated the same dataset entry.

Tokenizer. This component transforms an entity pair $(e1, e2)$, described in the records of the EM dataset as a series of attributes $a_{11}, ..., a_{1M}, a_{21}, ..., a_{2M}$, where a_{ij} is the attribute j of the entity e_i, into one or more token sequences. The component defines how to combine the values of the records to obtain tokens that enable the automatic learning of deep relations between attributes describing the same features of different entities. There are three alternative ways for performing this operation: the unstructured, attribute-based, and hybrid tokenization mode. In the unstructured mode, all fields describing the entities are concatenated into a unique sentence and any reference to the dataset schema is lost. In the attribute-based mode, the dataset entry is tokenized at the attribute level and the values of the same attribute for the considered pair of entities are coupled. Through this kind of tokenization, the records in the EM datasets describing pairs of entities are broken down into multiple sub-pairs, one for each attribute. Finally, in the hybrid mode, partial and / or incremental concatenations of the attribute values are performed. A hybrid strategy can for example apply an attribute-level tokenization and then incrementally combine the sub-pairs so that the $i-th$ pair contains the values of the first i attributes, and the last one compares the entire original matching pair. The aforementioned hybrid technique and the attribute-based technique have been experimented in the paper.

Dataset	Type	Datasets	Size	% Match
S-DG	Structured	DBLP-GoogleScholar	28,707	18.63
S-DA		DBLP-ACM	12,363	17.96
S-AG		Amazon-Google	11,460	10.18
S-WA		Walmart-Amazon	10,242	9.39
S-BR		BeerAdvo-RateBeer	450	15.11
S-IA		iTunes-Amazon	539	24.49
S-FZ		Fodors-Zagats	946	11.63
T-AB	Textual	Abt-Buy	9,575	10.74
D-IA	Dirty	iTunes-Amazon	539	24.49
D-DA		DBLP-ACM	12,363	17.96
D-DG		DBLP-GoogleScholar	28,707	18.63
D-WA		Walmart-Amazon	10,242	9.39

Table 1: Magellan Benchmark

Regardless of the specific implementation, the result of this step consists of one or more token sequences for each entry of the original EM dataset.

Embedder. The goal of the *Embedder* is to encode a token sequence into a multi-dimensional vector, i.e., an embedding. The approach usually adopted in the literature starts from a pre-trained word embedding deep learning architecture which is experimented with some token sequences. The embeddings are generally then extracted from these architectures by averaging the last hidden layer for each token, but other techniques have been experimented (the concatenations of the last 4 hidden layers for each token in [6]). In the paper, we experimented five embedders: Bert, DistilBert (DBert), Albert, Roberta and XLNET.

The *Embedder* outputs an embedding for each input token sequence. Recall that, according to the tokenization strategy adopted, an entry in the dataset can generate from one to many embeddings.

Combiner. The embeddings generated from each entry in the EM dataset are summarized in a single multi-dimensional vector by the *Combiner*. The standard approach, experimented in the paper, for performing this task is to calculate an average embedding.

5 EXPERIMENTAL EVALUATION

This section aims to provide an experimental answer to three main questions: 1) How effective are standard AutoML systems in solving EM tasks (Section 5.1); 2) To which extent the performance of AutoML systems applied to the EM task benefit from data preprocessing techniques (Section 5.2); and 3) How much AutoML systems pipelined with the *EM adapters* outperform the current state-of-the-art EM models (Section 5.3).

Datasets. The experiments have been performed against the datasets provided by the Magellan library[2] which are considered as a standard benchmark for the evaluation of EM tasks. In Table 1 we summarize with some statistic measures the 12 datasets analyzed, reporting for each of them the total number of records representing matching entities (fourth column) and the percentage of records associated with a matching label (last column). Each dataset is divided into training, validation, and test sets which were created with 60-20-20 proportions. The code used in the experiments is available at *https://github.com/softlab-unimore/automl-for-em*.

[2]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

	AutoSklearn		AutoGluon		H2OAutoML		DeepMatcher	
	F1	Training time (h)	F1	Training time (h)	F1	Training time (h)	F1	Training time (h)
S-DG	50.65	1.00	**77.85**	4.42	64.74	0.97	94.70	8.50
S-DA	92.79	1.00	**97.62**	2.06	92.51	0.94	98.40	8.50
S-AG	**44.10**	1.00	23.28	1.57	36.88	0.96	69.30	1.50
S-WA	29.28	1.00	19.12	2.48	**31.07**	0.94	66.90	2.17
S-BR	40.00	1.00	0.00	0.07	**43.24**	0.74	72.70	0.08
S-IA	53.33	1.00	50.00	0.13	**59.09**	0.87	88.00	0.25
S-FZ	**100.00**	1.00	71.11	0.10	61.90	0.86	100	0.17
T-AB	26.47	1.00	11.41	1.63	**27.36**	0.94	62.80	3.50
D-IA	**64.00**	1.00	60.87	0.15	62.75	0.87	74.50	0.17
D-DA	54.74	1.00	**89.44**	3.07	67.92	0.94	98.10	4.00
D-DG	46.79	1.00	**69.05**	4.96	43.01	0.97	93.80	8.50
D-WA	25.75	1.00	14.12	2.10	**26.31**	0.97	46.00	2.50

Table 2: Effectiveness of AutoML systems in EM tasks.

5.1 AutoML systems solving EM tasks

The experiment shows the application of standards AutoML systems in solving EM tasks. We selected three well-known approaches, AutoSklearn, AutoGluon, and H2OAutoML. We evaluated their performance as binary classifiers against the chosen datasets. The AutoML systems have been experimented with their default configuration, no parameter tuning has been performed. Only the AutoSklearn system required the application of a data pre-processing step to transform categorical features (which are not managed by this system) into numerical values. We performed this operation via a standard Word2Vec embedding, where the average Word2Vec embedding for each token of non-numeric attributes has been computed and concatenated. The results of this experiment are shown in Table 2, reporting the F1 score and the training time of each AutoML system for each dataset. The last column in the Table shows the scores achieved by DeepMatcher, in the *Hybrid* configuration, which we consider as a baseline[3].

Discussion. The AutoML systems are not effective in solving EM tasks as DeepMatcher. They perform close to DeepMatcher (and with an average F1 score greater than 75%) in two datasets only (S-DA and S-FZ). Even if there are small variations among the performance achieved, there is no AutoML system that clearly outperforms the others: the average F1 scores along all the datasets are close, ranging from 48.66% for AutoGluon, to 51.4% for H2OAutoML, and 52.33% for AutoSklearn. Nevertheless, the time required for training the models was largely varying. H2OAutoML took 1 hour maximum to finalize the training in all the datasets; AutoSklearn limits the training time to one hour by default; the time required for training AutoGluon was larger, more than four hours for the S-DG and D-DG datasets. We performed a further experiment, limiting the training time of AutoGluon to 1 hour. The quality of the results largely decreased and the average F1 score across all datasets dropped to 42.4%.

Lesson Learned. AutoML systems are not competitive when addressing binary classification problems associated with EM tasks.

5.2 AutoML systems pipelined with EM adapters

In this section, we evaluate the effectiveness of AutoML systems coupled with an *EM adapter*. For this reason, we experimented several adapters, obtained by combining possible implementations for their constituting modules, the tokenizer, the embedder and the combiner, introduced in Section 4. In the following, we report the effectiveness measured in adapters implementing attribute-based and hybrid tokenizers, where the best results are obtained. For each kind of tokenizer, five standard transformer architectures, namely Bert, DistilBert (DBert), Albert, RoBERTa and XLNET have been evaluated[4]. For sake of simplicity, we only consider combiners generating the embeddings from the last hidden layer for each token and averaging it with the layers referring to the other tokens. Tables 3 shows the results of this evaluation for AutoSklearn, AutoGluon and H2OAutoML respectively. The scores of the *EM adapters* were grouped according to the tokenization technique (attribute-based vs hybrid). The F1 scores in bold represent the best result per dataset obtained for the category of tokenizer considered. The values in bold and underlined represent the results of the most effective *EM adapter* for the considered dataset.

Discussion. We observe that (1) the *EM adapters* implementing hybrid tokenizers typically obtain the best performance (only in 3/12 datasets, i.e. S-DA, S-WA and S-FZ, the results are worse); (2) the Albert embedder achieves the best results (i.e., *EM adapter* implementing an Albert embedder is the best solution for 7/12 datasets considering the AutoSklearn system and 8/12 considering AutoGluon and H2OAutoML). Finally, Table 4 reports an overall evaluation of the impact of an *EM adapter* as a preprocessing component for an AutoML system in solving EM tasks. For each dataset and AutoML system, the F1 scores obtained in the absence and presence of an *EM adapter* are reported. The "No EM-Adapter" column is the reference column showing the F1 scores obtained by AutoML systems with no *EM adapter*. The remaining columns show the average F1 score (through the 5 transformer architectures considered) obtained by the adapters implementing attribute-based and hybrid tokenizers. Finally, for each AutoML system, the delta column shows the difference between the F1 score obtained with no *EM adapter* and the average of the two versions including the *EM adapters*. The experiments show that adapters significantly improve the effectiveness of AutoML systems in solving EM tasks in almost all datasets (the datasets S-FZ for AutoSklearn and S-DA for AutoGluon show an anomaly result). The average F1 score increases of 24.96%, 28.02% and 23.6% for AutoSklearn, AutoGluon and H2OAutoML respectively.

Lesson Learned. *EM adapters* largely improve the effectiveness of AutoML systems in addressing EM tasks. The experiments were not able to show a clear winner among the approaches tested. This is a positive result since it means that the AutoML technology can benefit from the application of *EM adapters*.

5.3 EM adapters pipelined with AutoML systems vs ad hoc solutions

The aim of this experiment is to evaluate if an AutoML system pipelined with an *EM adapter* can obtain competitive results with respect to other state-of-the-art EM models. For this evaluation, we consider an AutoML system with an *EM adapter* consisting of a hybrid tokenizer and an Albert embedder, whose combination provided the best performances in the previous experiments.

[3]We used the implementation available at https://github.com/anhaidgroup/deepmatcher

[4]No fine-tuning technique was applied in the experiments.

(a) EM-Adapter with AutoSklearn

	Attr-EM-Adapter					Hybrid-EM-Adapter				
	Bert	**DBert**	**Albert**	**Roberta**	**XLNET**	**Bert**	**DBert**	**Albert**	**Roberta**	**XLNET**
S-DG	92.70	90.49	**92.85**	92.11	91.65	**93.76**	91.92	93.56	92.59	92.77
S-DA	**98.10**	95.74	96.54	97.89	96.76	**96.90**	95.73	96.64	96.12	96.22
S-AG	62.11	59.59	**66.67**	60.07	51.38	66.67	62.60	**68.41**	61.25	58.74
S-WA	58.05	55.73	**67.18**	55.21	56.36	56.93	52.36	**62.17**	48.00	46.92
S-BR	66.67	68.75	**73.33**	70.97	66.67	64.52	78.79	**74.29**	**74.29**	68.97
S-IA	88.46	88.89	83.64	80.77	**90.20**	83.64	87.27	85.71	88.46	**96.30**
S-FZ	97.67	97.78	95.24	**100.00**	93.02	**97.67**	**97.67**	95.24	**97.67**	**97.67**
T-AB	58.44	57.08	**66.37**	56.68	54.65	66.97	58.23	**76.92**	66.36	61.01
D-IA	56.52	58.82	**67.92**	51.85	64.52	80.70	77.97	**91.23**	79.25	87.72
D-DA	92.90	91.58	**96.30**	91.71	93.20	**96.58**	95.41	96.36	95.20	96.00
D-DG	86.73	85.63	**90.29**	86.22	86.67	93.00	91.39	**93.12**	92.76	92.25
D-WA	39.82	45.71	**56.02**	39.17	37.99	51.38	51.10	**62.80**	46.23	44.23

(b) EM-Adapter with AutoGluon

	Attr-EM-Adapter					Hybrid-EM-Adapter				
	Bert	**DBert**	**Albert**	**Roberta**	**XLNET**	**Bert**	**DBert**	**Albert**	**Roberta**	**XLNET**
S-DG	93.35	91.88	**93.53**	92.19	92.76	94.30	92.83	**94.37**	93.25	93.56
S-DA	98.19	97.31	96.85	**98.31**	97.64	96.67	96.31	**96.75**	96.32	96.21
S-AG	57.78	50.14	**64.61**	54.04	45.39	58.06	58.55	**64.89**	53.40	55.83
S-WA	54.07	58.13	**67.04**	52.04	52.34	55.46	49.52	**64.67**	47.88	42.99
S-BR	64.29	66.67	**80.00**	63.64	64.29	64.29	64.29	**81.25**	71.43	75.86
S-IA	84.62	**88.46**	85.19	85.19	79.17	86.79	86.79	92.59	86.79	**98.18**
S-FZ	97.67	**100.00**	97.67	97.67	95.24	97.67	**100.00**	95.24	**100.00**	**100.00**
T-AB	58.49	58.72	**70.95**	56.55	56.76	60.18	64.22	**73.97**	63.69	59.71
D-IA	**69.39**	61.90	63.83	58.33	50.00	80.00	77.55	**87.27**	81.48	82.35
D-DA	93.00	93.12	**96.02**	91.05	92.97	96.54	95.58	**96.76**	96.00	95.88
D-DG	88.78	87.78	**91.73**	87.86	88.45	92.51	92.35	93.10	**93.16**	92.58
D-WA	32.47	44.52	**56.89**	33.79	33.55	51.55	49.35	**65.56**	43.29	39.13

(c) EM-Adapter with H2OAutoML

	Attr-EM-Adapter					Hybrid-EM-Adapter				
	Bert	**DBert**	**Albert**	**Roberta**	**XLNET**	**Bert**	**DBert**	**Albert**	**Roberta**	**XLNET**
S-DG	91.52	90.23	**92.25**	90.08	90.86	92.41	91.87	**93.78**	92.29	92.16
S-DA	**98.08**	96.44	96.72	96.57	96.15	96.03	94.45	**96.96**	94.77	95.58
S-AG	55.02	52.61	**61.86**	59.11	52.81	61.96	59.19	**66.54**	60.32	46.51
S-WA	55.07	53.61	**65.49**	48.28	53.22	49.47	47.89	**59.08**	46.53	45.49
S-BR	70.97	69.23	**78.79**	72.00	52.63	66.67	58.33	**86.67**	70.97	74.07
S-IA	84.62	86.79	84.62	**87.27**	82.61	80.70	92.00	90.57	86.79	**94.34**
S-FZ	87.18	92.68	**97.67**	90.00	84.21	92.68	95.24	92.68	**100.00**	**100.00**
T-AB	50.36	50.11	**65.88**	52.58	51.92	56.19	55.76	**70.53**	57.01	55.50
D-IA	51.16	56.60	70.37	58.18	**80.00**	76.67	80.85	80.77	78.43	**82.35**
D-DA	87.02	89.83	**95.02**	88.86	90.18	94.41	95.51	**96.32**	94.55	94.83
D-DG	83.83	84.10	**89.85**	84.48	83.92	89.87	87.53	**92.34**	91.90	91.93
D-WA	33.40	38.99	**55.37**	30.03	33.59	46.60	47.90	**61.06**	42.79	40.96

Table 3: EM-Adapter effectiveness for AutoML systems.

	AutoSklearn				AutoGluon				H2OAutoML			
	No EM-Adapter	**Attr-EM-Adapter**	**Hybrid-EM-Adapter**	**Δ**	**No EM-Adapter**	**Attr-EM-Adapter**	**Hybrid-EM-Adapter**	**Δ**	**No EM-Adapter**	**Attr-EM-Adapter**	**Hybrid-EM-Adapter**	**Δ**
S-DG	50.65	91.96	**92.92**	41.79	77.85	92.74	**93.66**	15.35	64.74	90.99	**92.50**	27.00
S-DA	92.79	**97.01**	96.32	3.87	97.62	**97.66**	96.45	-0.56	92.51	**96.79**	95.56	3.66
S-AG	44.10	59.96	**63.53**	17.65	23.28	54.39	**58.15**	32.98	36.88	56.28	**58.90**	20.71
S-WA	29.28	**58.50**	53.28	26.61	19.12	**56.72**	52.10	35.30	31.07	**55.13**	49.69	21.34
S-BR	40.00	69.28	**72.17**	30.72	0.00	67.77	**71.42**	69.60	43.24	68.72	**71.34**	26.79
S-IA	53.33	86.39	**88.28**	34.00	50.00	84.52	**90.23**	37.38	59.09	85.18	**88.88**	27.94
S-FZ	**100.00**	96.74	97.19	-3.04	71.11	97.65	**98.58**	27.01	61.90	90.35	**96.12**	31.33
T-AB	26.47	58.64	**65.90**	35.80	11.41	60.29	**64.36**	50.92	27.36	54.17	**59.00**	29.23
D-IA	64.00	59.93	**83.37**	7.65	60.87	60.69	**81.73**	10.34	62.75	63.26	**79.81**	8.79
D-DA	54.74	93.14	**95.91**	39.79	89.44	93.23	**96.15**	5.25	67.92	90.18	**95.13**	24.74
D-DG	46.79	87.11	**92.50**	43.01	69.05	88.92	**92.74**	21.78	43.01	85.23	**90.71**	44.96
D-WA	25.75	43.74	**51.15**	21.69	14.12	40.24	**49.78**	30.89	26.31	38.27	**47.86**	16.76

Table 4: Impact of EM-Adapter on AutoML performance. Bold values indicate the best configuration for a specific AutoML system; underlined values the best results for the considered dataset.

This system was then compared with the *Hybrid* variant of DeepMatcher. Table 5 reports the results of this comparison. We performed two experiments, in the first we limited the training time of the AutoML systems to 1 hour. In the second experiment, we set that time to 6 hours. The offset between the average effectiveness of AutoML systems and DeepMatcher is shown for each configuration.

Discussion. AutoML systems limited to 1 hour of training show better effectiveness than DeepMatcher in 5/12 datasets (i.e. S-BR, S-IA, T-AB , D-IA and D-WA, with an average increase of F1 equal to 9%). In the remaining cases, they generate slightly lower results, with an average F1 difference of 3.2%. However, we notice that AutoSklearn used the entire training time budget, but AutoGluon and H2OAutoML took on average a training time of 0.61 h and 0.76 h respectively. Furthermore, DeepMatcher has been trained for less than 1 hour in only 4/12 datasets. If we consider a tolerance threshold of 2%, the EM-adapted AutoML systems are comparable or outperform DeepMatcher in 9/12 datasets. This trend is further confirmed when we limit the training time to 6 hours. The average F1 score increases of 12% in the 5 datasets where the AutoML systems obtain the best results. Assuming as before a 2% tolerance threshold in F1 scores, EM-adapted AutoML systems are comparable or outperform DeepMatcher in 11/12 datasets. Also in this case, only AutoSklearn used the entire time budget for training, while AutoGluon and H2OAutoML took an average of 3.68 hours and 3.24 hours respectively (compared to 2.92 hours of DeepMatcher).

Lesson Learned. AutoML systems pipelined with *EM adapters* perform as or greater than state-of-the-art EM tools.

	DeepMatcher (Hybrid)		EM-Adapted AutoMLs (1h time budget)				EM-Adapted AutoMLs (6h time budget)			
	F1	Time (h)	AutoSklearn	AutoGluon	H2OAutoML	Δ	AutoSklearn	AutoGluon	H2OAutoML	Δ
S-DG	94.70	8.50	93.56	92.98	**93.78**	-1.26	94.02	**94.16**	93.82	-0.70
S-DA	98.40	3.75	96.64	96.75	**96.96**	-1.62	**97.08**	96.85	**97.08**	-1.40
S-AG	69.30	1.50	**68.41**	59.03	66.54	-4.64	68.41	64.53	**69.24**	-1.90
S-WA	66.90	2.17	**62.17**	58.01	59.08	-7.15	65.16	**66.12**	63.50	-1.97
S-BR	72.70	0.08	74.29	81.25	**86.67**	8.03	76.47	81.25	**86.67**	8.76
S-IA	88.00	0.25	85.71	**92.59**	90.57	1.62	91.23	92.59	**94.12**	4.65
S-FZ	100.00	0.17	**95.24**	**95.24**	92.68	-5.61	**97.67**	**97.67**	95.24	-3.14
T-AB	62.80	3.50	**76.92**	69.83	70.53	9.63	76.92	76.88	**77.06**	14.16
D-IA	74.50	0.17	**91.23**	87.27	80.77	11.92	**91.23**	**91.23**	82.35	13.77
D-DA	98.10	4.00	96.36	**97.12**	96.32	-1.50	96.36	97.12	**97.33**	-1.16
D-DG	93.80	8.50	**93.12**	92.59	92.34	-1.12	**93.53**	93.38	93.21	-0.43
D-WA	46.00	2.50	**62.80**	57.05	61.06	14.30	**67.77**	62.50	65.44	19.23

Table 5: Effectiveness of EM-Adapter with AutoML systems compared to DeepMatcher. Values in bold indicate the best configuration for a training time budget configuration.

6 CONCLUSION

The adoption of AutoML systems would make machine learning and deep learning based approaches for addressing EM tasks usable also for non-expert people. The direct application of AutoML systems to the EM problem is not possible. The reason is mainly due to the schema adopted by the datasets representing EM tasks whose records encode pairs of entities, and to the classification problem which is highly imbalanced. In this paper, we address the first problem, by introducing the *EM adapter* component which transforms the records of datasets representing entity pairs into a form which is effectively processable by AutoML systems. Our experiments show that this approach achieves a performance similar to the one of EM-task specific systems (but it does not require expert users to tune it). The future work will try to improve the performance (1) by introducing data augmentation techniques for creating more balanced training datasets for the AutoML systems; (2) by experimenting techniques for improving the embeddings (via "local embeddings" [3], generated taking into account the current dataset, and/or performing a fine-tune of the existing techniques).

ACKNOWLEDGEMENT

This work was partially funded by SBDIO I4.0 (https://www.sbdioi40.it/), an industrial research project funded by the POR FESR Emilia Romagna 2014-2020 as part of the Smart Specialization Strategy (S3).

REFERENCES

[1] Gary Anthes. 2017. Artificial intelligence poised to ride a new wave. *Commun. ACM* 60, 7 (2017), 19–21.
[2] Ursin Brunner and Kurt Stockinger. 2020. Entity Matching with Transformer Architectures - A Step Forward in Data Integration. In *EDBT*. OpenProceedings.org, 463–473.
[3] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *SIGMOD Conference*. ACM, 1335–1349.
[4] Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer.
[5] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2020. An Overview of End-to-End Entity Resolution for Big Data. 53, 6, Article 127 (Dec. 2020), 42 pages. https://doi.org/10.1145/3418896
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.
[7] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proc. VLDB Endow.* 11, 11 (2018), 1454–1467.
[8] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander J. Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *CoRR* abs/2003.06505 (2020).
[9] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2019. Auto-sklearn: Efficient and Robust Automated Machine Learning. In *Automated Machine Learning*. Springer, 113–134.
[10] H2O.ai. 2017. *H2O AutoML*. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html H2O version 3.30.0.1.
[11] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In *KDD*. ACM, 1946–1956.
[12] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. 2019. Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA. In *Automated Machine Learning*. Springer, 81–95.
[13] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (Sept. 2020), 50–60. https://doi.org/10.14778/3421424.3421431
[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).
[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR (Workshop Poster)*.
[16] 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 Conference*. ACM, 19–34.
[17] Giang Nguyen, Stefan Dlugolinsky, Martin Bobák, Viet D. Tran, Álvaro López García, Ignacio Heredia, Peter Malík, and Ladislav Hluchý. 2019. Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artif. Intell. Rev.* 52, 1 (2019), 77–124.
[18] Matteo Paganelli, Paolo Sottovia, Francesco Guerra, and Yannis Velegrakis. 2019. TuneR: Fine Tuning of Rule-based Entity Matchers. In *CIKM*. ACM, 2945–2948.
[19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019).
[20] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing Entity Matching Rules by Examples. *Proc. VLDB Endow.* 11, 2 (2017), 189–202.
[21] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan E. Hines, C. Bayan Bruss, and Reza Farivar. 2019. Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools. In *ICTAI*. IEEE, 1471–1479.
[22] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity Matching: How Similar Is Similar. *Proc. VLDB Endow.* 4, 10 (2011), 622–633.
[23] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP (Demos)*. Association for Computational Linguistics, 38–45.
[24] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*. 5754–5764.
[25] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. In *WWW*. ACM, 2413–2424.