Deep Entity Matching: Challenges and Opportunities

YULIANG LI, JINFENG LI, YOSHIHIKO SUHARA, JIN WANG, WATARU HIROTA, and WANG-CHIEW TAN, Megagon Labs

Entity matching refers to the task of determining whether two different representations refer to the same real-world entity. It continues to be a prevalent problem for many organizations where data resides in different sources and duplicates the need to be identified and managed. The term "entity matching" also loosely refers to the broader problem of determining whether two heterogeneous representations of *different entities* should be associated together. This problem has an even wider scope of applications, from determining the subsidiaries of companies to matching jobs to job seekers, which has impactful consequences.

In this article, we first report our recent system Ditto, which is an example of a modern entity matching system based on pretrained language models. Then we summarize recent solutions in applying deep learning and pre-trained language models for solving the entity matching task. Finally, we discuss research directions beyond entity matching, including the promise of synergistically integrating blocking and entity matching steps together, the need to examine methods to alleviate steep training data requirements that are typical of deep learning or pre-trained language models, and the importance of generalizing entity matching solutions to handle the broader entity matching problem, which leads to an even more pressing need to explain matching outcomes.

CCS Concepts: • **Information systems → Entity resolution**; **Deduplication**; • **Computing methodologies → Neural networks**;

Additional Key Words and Phrases: Entity matching, entity resolution, data integration, deep learning, pre-trained language models

ACM Reference format:
Yuliang Li, Jinfeng Li, Yoshihiko Suhara, Jin Wang, Wataru Hirota, and Wang-Chiew Tan. 2021. Deep Entity Matching: Challenges and Opportunities. *J. Data and Information Quality* 13, 1, Article 1 (January 2021), 17 pages.
https://doi.org/10.1145/3431816

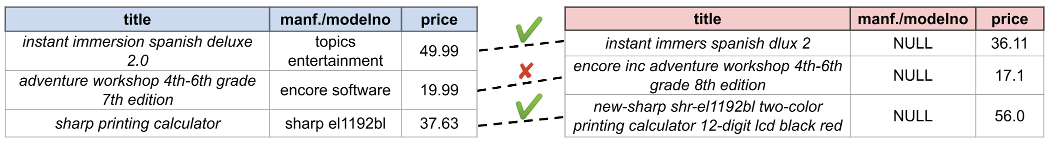
1 INTRODUCTION

Entity matching (EM, also known as *entity resolution*, *record linkage*, *duplicate detection*, or *reference reconciliation*)[1] refers to the problem of determining whether two different representations refer

[1]According to [23], the name itself needs entity resolution.

Authors' addresses: Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, and W.-C. Tan (corresponding author), Megagon Labs, 444 Castro Street Suite 720, Mountain View, CA 94041; emails: {yuliang, jinfeng, yoshi, jin, wataru}@megagon.ai.

1936-1955/2021/01-ART1 $15.00
https://doi.org/10.1145/3431816



title	manf./modelno	price
instant immersion spanish deluxe 2.0	topics entertainment	49.99
adventure workshop 4th-6th grade 7th edition	encore software	19.99
sharp printing calculator	sharp el1192bl	37.63

title	manf./modelno	price
instant immers spanish dlux 2	NULL	36.11
encore inc adventure workshop 4th-6th grade 8th edition	NULL	17.1
new-sharp shr-el1192bl two-color printing calculator 12-digit lcd black red	NULL	56.0

Fig. 1. Entity matching for two tables: determine the pairs of the matching entries from the two datasets. In general, the column names, the order of columns, and the number of columns need not be identical. Ditto arrives at the correct matching decisions on these pairs of records. This example is from [35].

to the same real-world entity, for example, whether the two records (Name:*Jane Smith*, Affiliation: *IBM*, Title: -) and (Company: *International Business Machines Corp.*, Name: *J. M. Smith*, Occupation: *Staff Manager*) refer to the same person *Jane Smith*. To exemplify further, Figure 1 shows two tables with three records each where we would like to understand which pairs of records (one from each table) are matching records. EM is a long-standing problem in data integration and data cleaning, going as far back as [16]. It is used for identifying and removing duplicates [10, 25, 49], and it is a necessary step towards creating holistic views of entities—by identifying disparate information about the same entity and then unifying them into a single view—which are needed for applications such as master data management [66, 67] and knowledge base construction [27, 74]. The term "entity matching" also loosely refers to the broader problem of determining whether two possibly heterogeneous representations of *different entities* should be associated together. This problem has an even wider scope of applications: from matching subsidiaries to their parent companies where both sets of entities are relational to matching job descriptions with job seekers where the former may reside in a semi-structured format and the latter may reside in a pdf file or a relation. Until now, solutions to the broader EM problem were developed independently and were different from those developed for the traditional EM problem. As we shall describe, recent advances in deep learning techniques hold strong promise for providing a uniform treatment to the EM problem, whether in the traditional or in the broader setting. However, solutions based on deep learning techniques also lead a number of other considerations, including the need to find ways to reduce the amount of training data required and to explain matching outcomes, which is especially important for matching decisions that will impact lives.

Returning to Figure 1, out of the nine candidate matching pairs, only the first and last pairs are true matching pairs even though their title, manf./modelno, and price are quite different. More interestingly, the second pair is not a matching pair even though the title and price are quite similar.

The decision on whether two records should match or not is predominantly based on measuring the *similarity* between them [15, 31, 33]. Earlier methods for determining the degree of similarity between two records compare the *syntactic* similarities of values with the corresponding attributes of the two records [33, 34, 70]. This typically means that, to obtain better accuracy, it is important to understand a priori which are the corresponding attributes before determining the similarities of the values of those attributes and then aggregating the degrees of similarities to arrive at an overall match/unmatch decision. With recent advances in natural language processing (NLP), it has become possible to adopt similarity measures based on the *semantics* of tokens in text that occur in attribute values. In fact, the Transformer architecture [69, 76] generates token embeddings based on all tokens in the input sequence. Hence, the generated embeddings are highly contextualized and thus able to capture both the semantics and contextual understanding of the words far better than conventional word embedding techniques such as word2vec [45], GloVe [53], or fastText [5]. It is for this reason that pre-trained language models, which use Transformer architectures, can help make better matching decisions through their ability to understand the semantics of tokens and discern homonyms and synonyms. That is, the same word may have different meanings in different phrases, and, respectively, different words may have the same meaning in different phrases. For example, the word "*Sharp*" has different meanings in "*Sharp resolution*" versus "*Sharp TV.*" On the other hand, the words "*second*" and "*2nd*" have the same meaning in the phrases "*second edition spanish*" and "*Spanish 2nd edition.*"

As the above examples illustrate, pre-trained language models are robust to semantic heterogeneity in language. Like other deep learning models, they are also robust to structural heterogeneity because they learn to pay attention to the "right" components of two records when making matching decisions. A particularly convenient feature of deep learning models is that one need not specify which are the corresponding attributes of two records a priori to make matching decisions. In fact, the two records may have attributes that appear in a different order under different attribute names, and such models are still able to learn to pay attention to the right components despite the structural differences. For the same reason, such methods are also robust to records that may contain corrupt or missing values and attribute names or differ in the type of structure (e.g., one is a flat record, while the other is semi-structured), making them promising candidates for the broader EM task.

Contributions We present the following in this article:

- We report our recent work Ditto [35], which is an example of a modern EM system based on pre-trained language models (LMs). Ditto features a versatile architecture for serializing the entity records and further boosts its performance with three optimizations: domain knowledge, data augmentation, and summarization. (See Section 2.)
- We provide an overview of recent trends in applying deep learning techniques to EM. In particular, we summarize EM solutions based on non-Transformer techniques such as recurrent neural networks (RNNs) and word embeddings. We also discuss EM systems that are based on pre-trained LMs. (See Section 3.)
- We discuss what we believe are exciting directions beyond EM that are largely fueled by the use of deep learning methods for EM, including the need to reduce the training data requirements of deep-learning-based EM solutions and extend deep-learning-based solutions for the broader EM problem, which leads to an even more pressing need to explain matching outcomes. (See Section 4.)

2 DITTO

Ditto [35] relies on pre-trained Transformer-based LMs (or pre-trained LMs in short) to support EM. In Ditto, we reduce the EM problem to a sequence-pair classification problem to leverage such models. Pre-trained LMs have been shown to generate highly contextualized embeddings that capture better language understanding compared to traditional word embeddings, which can help with matching entities with significant text descriptions. Such models are also more robust to inconsistencies that may occur in data, such as missing or corrupt values. Ditto further improves its matching capability through three optimizations:

(1) Ditto allows experts to explicitly inject domain knowledge into the training examples by highlighting important pieces of the input that are worthy of attention in making matching decisions (e.g., model number, brand name in long text descriptions).
(2) Ditto augments training data with (difficult) examples, which reduces the amount of labeled training data required to train Ditto.
(3) Ditto can summarize long strings so that it pays attention to only the most essential information that is retained for EM. The summarization feature is also essential as most pre-trained LMs allow at most 512 input tokens.

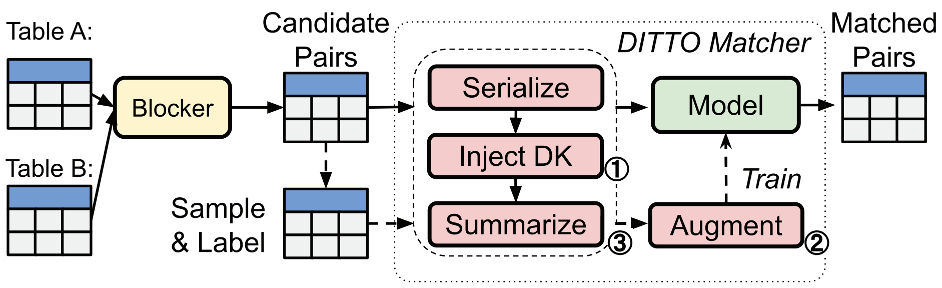


Fig. 2. An entity matching pipeline with a blocking phase followed by entity matching using Ditto [35]. The diagram shows that Ditto can be optimized by (1) injecting domain knowledge (DK), (2) leveraging data augmentation (DA) operators to boost the training set, and (3) using summarization techniques for keeping essential information for matching.

Figure 2 depicts the overall architecture of Ditto, including the blocking phase, which is a step taken to reduce the number of candidate matching pairs for the matching phase, especially when the input tables are large. We discuss more on blocking in Section 4.1.

2.1 Leveraging Pre-trained Language Models

The word embeddings of pre-trained LMs are highly contextual. The embedding of a word token depends on its surrounding tokens. For example, the embedding of the token "*Sharp*" in the phrase "*Sharp resolution*" is different from that of "*Sharp TV*" because their surrounding tokens are different. Contextual word embeddings are therefore more aligned with the semantics of word tokens compared to existing deep learning models that use context-independent embeddings such as word2vec, GloVe, or fastText. As mentioned earlier, contextual word embeddings can distinguish the same token that is used in different contexts, and also the converse, different tokens under similar context that have the same meaning.

In Ditto, we reduce the EM problem to a binary sequence-pair classification problem to leverage such models. To feed training examples into such models, we need to serialize the two corresponding entities of the two tables into a sequence pair. An entity e (or a record) is a set of attribute-value pairs in the format of $\{(attr_i, val_i)\}$, $1 \leq i \leq k$, where e has k attribute-value pairs. Ditto serializes e into

$$\textit{serialize(e)}\text{: }\texttt{[COL]}\ \text{attr}_1\ \texttt{[VAL]}\ \text{val}_1\ \ldots\ \texttt{[COL]}\ \text{attr}_k\ \texttt{[VAL]}\ \text{val}_k,$$

where [COL] and [VAL] are special tokens to indicate the start of an attribute and a value, respectively. It then passes the pair of entities e and e' as the sequence pair

$$\texttt{[CLS]}\ \textit{serialize(e)}\ \texttt{[SEP]}\ \textit{serialize(e')}$$

to the pre-trained LM. The number of attributes in e' may be different from that of e. We make no assumption that they must have the same number of attributes. Nor do we assume that the matching attributes are in order or must possess the same name.

Ditto serializes and feeds the training examples to the pre-trained LM to train their inner neural networks. This process is known as *fine-tuning*, in which the model will learn when to classify a sequence pair as matching or not. We experimented with Ditto with three representative pre-trained LMs, BERT [13], RoBERTa [36], and DistillBERT [62], for fine-tuning.[2] For each model, we

[2]DistillBERT is a smaller version of BERT that is faster to train but less "powerful" in general when compared with BERT. RoBERTa is a more optimized version of BERT that is shown to outperform BERT on several benchmarks.

append a fully connected layer and a softmax output layer to the model for classification. During training, we initialize a model with its pre-trained parameters (e.g., word embeddings) and wait for the model to converge on the training examples.

Although the serialization step seems uninteresting at best, Ditto gains significant flexibility through serializing the entity records the way it is proposed. In particular, the serialization scheme is agnostic to the actual schema of the tables, and hence, Ditto does not require two records to have the same schema nor to perform schema matching prior to serializing the entities. The built-in Transformer architecture also allows Ditto to automatically learn what are the important fields/attributes for matching automatically without manual feature engineering.

As described, the way Ditto leverages pre-trained models is extremely simple. Surprisingly, this simple architecture works extremely well. Ditto along with any of the pre-trained LMs, BERT, RoBERTa, and DistillBERT, can already achieve more accurate predictions than the previous deep-learning-based methods.

2.2 Optimizing Model Training

Even though Ditto was already outperforming the previous EM solutions, we observe three optimizations that can further improve the training of the fine-tuned model for better performance. The three optimizations are injecting domain knowledge, augmenting training data, and summarizing long strings, which we will describe next. The first and last technique helps the model pay attention to more important information, while the second technique helps the model learn better by providing more (difficult) training examples. These optimizations are general and can be applied to various pre-trained LMs (BERT, RoBERTa, and DistillBERT) to improve their prediction accuracy.

Injecting domain knowledge. We observe that human workers perform EM by paying extra attention to informative token spans in the records, and the informative token spans may occur at different places for different records. For example, when matching two products, workers tend to make critical judgments based on the model numbers, if they are available, and the model numbers may occur under different attributes for different records. Ditto allows developers to inject such domain knowledge by highlighting the spans of tokens of importance through *span typing* and *span normalization*.

Span typing highlights informative spans through adding a tag delimiter to the start and the end of a span. For example, after span typing, a sequence pair may contain tags to highlight model numbers in each entity:

“.. `[MODELNO] sharp el1192bl [/MODELNO]... [MODELNO] shr-el1192bl [/MODENO]..`”

This helps the Transformers model attend to the informative spans of two entities to make better predictions. Span normalization canonicalizes two spans of the same type into the same format. For example, phone numbers will be transformed into the same format (e.g., XXX-YYY-ZZZZ), or percentages (e.g., 5% and 5.00%) will be rewritten into a common format (e.g., 5.00%). Canonicalizing such entries makes it easier for Ditto to match the two numeric tokens/spans. Ditto provides developers with a friendly interface to inject their domain knowledge.

Augmenting training data. Obtaining sufficient training data for training deep learning models has always been a challenge. The same problem applies to Ditto. To alleviate this problem, Ditto leverages data augmentation to automatically generate more training examples from existing ones.

Ditto derives new training examples from existing examples through span-level, attribute-level, and entity-level augmentation operators. The two span-level operators, span_del and span_shuffle, randomly sample a span of tokens from the input sequence and remove/shuffle the tokens within the span. There are two attribute-level augmentation operators: attr_del and attr_shuffle. Given an existing sequence, attr_shuffle shuffles the order of attributes, and attr_del deletes a randomly chosen attribute along with the corresponding value. There is one entity-level operator that swaps the order of two entities e and e'. We expect the labels to stay unchanged after Ditto applies each transformation.

The goal of these operators is to create more examples so that Ditto will learn invariant properties that are useful for matching two entities. For example, attr_shuffle generates examples that will encourage Ditto to learn that the order of the attributes in an entity does not affect the outcome of matching decisions. Nor does the order of two entities e and e'.

Since some of the generated examples still may contain wrong labels, Ditto attempts to reduce the effect of wrong labels through the MixDA technique [43]. Intuitively, MixDA interpolates the augmented example with the original example to ensure label correctness. Our experiments demonstrate that Ditto benefits from the data augmentation techniques even when the original training set may be small [35].

Summarizing long strings. Long strings that occur in textual attributes of entities are problematic to the pre-trained LMs. First, it is more difficult for pre-trained LMs to understand what to focus on in the long string. Second, existing pre-trained LMs have a hard limit on the maximum sequence length due to limited GPU memory. For example, BERT accepts at most 512 tokens from the input sequence and automatically truncates sequences beyond this limit. Consequently, a simple truncation method can drop important information for matching and thus reduce the model's performance.

Ditto addresses this issue by keeping only the important tokens in long strings. Our current implementation of Ditto uses TF-IDF to measure the importance of a token. This technique automatically eliminates stop words, which typically have low TF-IDF scores, and hence, this method is effective for reducing false-positive predictions.

In general, other summarization techniques can also be used [44, 60]. In our experiments, we find that TF-IDF works sufficiently well in practice. With summarization, Ditto achieves a significant F1 score improvement on a text-heavy dataset, from 41% to over 93%.

2.3 Performance of Ditto

We evaluated Ditto on two benchmark datasets, WDC [55] and the ER-Magellan benchmark [30, 32], which are widely used to benchmark the performance of EM solutions. Here, we report only the results on the ER-Magellan datasets, and we refer the interested reader to [35] for further experimental results on the WDC benchmark. The ER-Magellan benchmark has 13 datasets and is divided into three categories:

- **Structured:** There are seven datasets under this category: Amazon-Google (AG), Beer (BR), DBLP-ACM (DA), DBLP-Google (DG), Fodors-Zagats (FZ), iTunes-Amazon (IA), and Walmart-Amazon (WA). Any two entities e and e' in each dataset must share the same schema.
- **Dirty:** There are four datasets under this category: DBLP-ACM (DA), DBLP-Google (DG), iTunes-Amazon (IA), and Walmart-Amazon (WA). In these datasets, the values of attributes of some records may be missing. These dirty datasets are derived from the structured datasets by removing some attribute values.
- **Textual:** There are two datasets under this category: Abt-Buy (AB) and Company (CM). An entity in AB (or CM) consists of a textual description for the entity.

These datasets form a diverse set of test cases to evaluate the applicability of an EM solution in different settings. We used RoBERTa as the pre-trained LM for Ditto since RoBERTa generally

Table 1. F1 Scores of Ditto on the ER-Magellan Benchmark Datasets Categorized by the Types of Data

	Structured							**Dirty**				**Textual**	
Dataset	AG	BR	DA	DG	FZ	IA	WA	DA	DG	IA	WA	AB	CM
Size	11,460	450	12,363	28,707	946	539	10,242	12,363	28,707	539	10,242	9,575	112,632
DM+	70.7	78.8	98.45	94.7	100	91.2	73.6	98.1	93.8	79.4	53.8	62.8	92.7
Ditto	75.58	94.37	98.99	95.6	100.00	97.06	86.76	99.03	95.75	95.65	85.69	89.33	93.85
	(+4.88)	(+15.57)	(+0.54)	(+0.9)	(+0.0)	(+5.86)	(+13.16)	(+0.93)	(+1.95)	(+16.25)	(+31.89)	(+26.53)	(+1.15)
Ditto(DK)	74.67	90.46	99.10	95.80	100.00	97.80	83.73	99.08	95.57	94.48	80.67	81.69	93.15
Ditto(DA)	75.08	87.21	99.17	95.73	100.00	97.40	85.50	98.94	95.47	95.29	85.49	89.79	93.69
Baseline	74.10	84.59	98.96	95.84	98.14	92.28	85.81	98.92	95.44	92.92	82.56	88.85	41.00

DM+ denotes the best performance among DeepMatcher (DM) [47] and its two follow-up works [19, 28]. Ditto consistently outperforms DM+ except on FZ, where they both achieve a 100% F1.

achieves better performance than BERT or DistillBERT on this benchmark. We compared Ditto's performance with the best-performing deep-learning-based methods for EM, including DeepMatcher [47] and two follow-up works [19, 28]. We denote by DM+ the system among [19, 28, 47] that achieves the highest F1 score. The F1 scores of Ditto and DM+ are tabulated in Table 1. Ditto achieves higher scores than DM+ consistently on all datasets, except FZ, where both Ditto and DM+ performed equally well (i.e., they both achieved the optimal score, 100.00). On all the datasets, Ditto improves the F1 score by up to 31.89. We observe that the F1 margins on the smaller datasets are larger than those of larger datasets. Specifically, on the seven smallest datasets (BR, FZ, IA, WA from Structured, IA, WA from Dirty, and AB from Textual), Ditto achieves an average improvement of 15.6, while the average F1 improvement on the remaining six larger datasets is 1.48. These results indicate that Ditto is a very effective EM solution that achieves the state of the art. Furthermore, it also brings significant performance improvement over existing solutions when the number of labels is small.

We conducted an ablation study to understand the effect of different optimizations of Ditto. We report the F1 scores of three variants of Ditto: Ditto(DK), Ditto(DA), and Baseline. Ditto(DK) uses the optimization with domain knowledge injection and Ditto(DA) uses the optimization with data augmentation. Both Ditto(DK) and Ditto(DA) enable long string summarization to process long strings. Baseline refers to Ditto without any optimizations (domain knowledge, data augmentation, and summarization). That is, it is the pre-trained LM RoBERTa without any optimization. We present the F1 scores of the three variants at the bottom of Table 1. In the absence of optimizations, Baseline already achieves higher F1 scores than DM+ on the majority of the datasets. The average F1 gap is 7.75 (excluding CM dataset), which accounts for 78.5% of the improvement of the full Ditto(9.87). For the CM dataset, the lengths of strings exceed the RoBERTa maximum sequence length. However, this issue can be easily addressed with summarization. When using data augmentation and domain knowledge injection, Ditto gains further performance improvements over Baseline. Ditto(DK) and Ditto(DA) improve the F1 scores of Baseline by up to 5.52 and 5.12, respectively. The results of this ablation study suggest that RoBERTa contributes largely to Ditto's superior performance and the added optimizations further improve Ditto's performance.

Comparing with HierMatcher and Seq2SeqMatcher. In addition to DM+, we also compare Ditto with two recent deep-learning-based methods, HierMatcher [18] and Seq2SeqMatcher [50] (see the next section for a more detailed summary). HierMatcher and Seq2SeqMatcher represent the state-of-the-art solutions for matching heterogeneous data, where two entities do not share the same schema. We summarize the results in Table 2. In this experiment, we adopt the evaluation setting of HierMatcher of matching heterogeneous data. The three datasets, Hetero_WA$_{\{3,4,5\}}$, are derived from the WA dataset by merging different combinations of the WA dataset's attributes.

Table 2. Comparing Ditto with HierMatcher and Seq2SeqMatcher on Matching Heterogeneous Data (Where the Entity Records Do Not Share the Same Schema; the F1 Scores for HierMatcher, Seq2SeqMatcher, and DM Are Taken from [18])

	Size	Baseline (BERT)	Ditto(DK)	Ditto(DA)	Ditto	HierMatcher	Seq2SeqMatcher	DM
Hetero_WA$_3$	10,242	82.20 (+1.50)	81.70 (+1.00)	83.84 (+3.14)	81.06 (+0.36)	80.70	75.60	67.10
Hetero_WA$_4$	10,242	82.74 (+1.34)	82.04 (+0.64)	83.00 (+1.60)	82.11 (+0.71)	81.40	74.70	63.40
Hetero_WA$_5$	10,242	83.22 (+2.22)	82.01 (+1.01)	83.63 (+2.63)	81.40 (+0.40)	81.00	74.40	66.50

The numbers in parentheses are improvements of Ditto against HierMatcher (which is the best among the 3 compared methods). All 4 variants of Ditto (including Baseline (BERT)) consistently outperform HierMatcher.

Our results in Table 2 show that all variants of Ditto consistently outperform HierMatcher, Seq2SeqMatcher, and DeepMatcher in this setting. In particular, Ditto with only the data augmentation optimization gives the overall best results. These results further demonstrate that Ditto's improved language understanding capability leads to the significant performance gain. Note that the experiments comparing Ditto with HierMatcher and Seq2SeqMatcher are new and were not presented in the original Ditto paper [35].

Additional experiments. We also evaluated Ditto on the WDC [55] dataset, where we showed similar results on the different sizes of WDC datasets. That is, Ditto consistently outperforms DeepMatcher in most cases. In particular, we showed that Ditto already outperforms DeepMatcher when given only at most half of the training data.

Apart from the qualitative evaluation, we also evaluate the training and prediction efficiency of Ditto. In short, although Ditto is deeper in its architecture, it can be trained in roughly the same time as DeepMatcher (within tens of minutes on a single GPU). The average prediction time for Ditto is within 10ms for each entity pair to be matched. We refer the interested reader to [35] for further details.

3 DEEP LEARNING MODELS FOR ENTITY MATCHING

Recent EM solutions are based on deep learning methods and they achieve state-of-the-art performance over EM benchmarks. Even more recently, EM solutions started to adopt pre-trained transformer-based methods, which outperform deep learning models largely due to their immensity for language understanding and the ability to learn where to pay attention. In what follows, we describe some of these techniques.

Non-Transformer-based methods. The early attempts and successes in applying deep learning techniques to EM tasks are DeepER [14] and DeepMatcher [47].

DeepER [14] adopts LSTM-based RNN with the Siamese architecture [48], which is a neural network architecture that has a pair of neural networks with exactly the same architecture and shared parameters. DeepER first tokenizes each pair of entities, which are converted into distributed representations respectively using a word embedding model such as GloVe [53] or fastText [5]. The model then aggregates token-level distributed representations into an entity representation for each entity. Next, the two entity representations are fed into a dense layer that calculates the similarity between the entities, followed by an output layer that predicts the matching. Thus, the entity-entity comparison is conducted in the dense layer based on the entity representation.

DeepMatcher [47] explored different architectures of the deep learning models for EM including a similar model as DeepER. In all the different design spaces of EM solutions under the DeepMatcher architecture, the EM task is formulated as the binary classification problem and so the models can be trained in the same way as DeepER. The authors reported that their attention-based model outperformed the non-DL-based model Magellan [30] by a considerable margin, even when only a small number of training data is available. Compared to DeepER, a notable difference is that

DeepMatcher's attention-based model explicitly takes into account attribute similarity, which evaluates the similarity between tokens from the same attribute of an input entity pair. This functionality allows DeepMatcher to capture attribute-level similarity better than DeepER because DeepER uses aggregated and hence coarser-grained entity-level representations for similarity calculation between two entities.

Recently, several deep learning models were developed to address the limitations of DeepER and DeepMatcher. Both DeepER and DeepMatcher consider an attribute as a sequence of tokens and leverage the word embedding model. However, these models are known not to perform numerical comparisons well. Fu et al. [19] developed a multi-perspective matching (MPM) model that takes multiple representations (numeric, string, and text) as input and adaptively switches similarity measures accordingly to address the issue.

To manage heterogeneous data where two entities may not share the same schema, Seq2SeqMatcher [50] extended DeepMatcher's attention-based model by relaxing the soft-alignment calculation (from the same attribute) to any attribute pairs. With the more general method of aligning attributes, Seq2SeqMatcher showed that it outperforms DeepMatcher and DeepER not only on heterogeneous EM tasks but also on other EM tasks on the ER-Magellan dataset. HierMatcher [18] further improved the performance on heterogeneous data by implementing three alignment layers for token-level, attribute-level, and entity-level similarity calculation. Recall, however, as we showed in Table 2, that Ditto still outperforms Seq2SeqMatcher and HierMatcher.

CorDEL [73] implements a symbolic operation that contrasts token sets of each attribute for each entity pair to create three token groups, namely unique tokens for the first/second entity and the shared tokens. CorDEL then converts the token sets into distributed representations using word embeddings so it can calculate similarity representations and difference representations individually. The representations are fed into the dense layers to predict matching in the same manner as other deep learning models.

Pre-trained Transfomer-based methods. Most of the deep learning models discussed above leverage word embedding models for better language understanding. However, word embeddings are known to have limitations on handling ambiguity. For example, a token "apple" can be a fruit name or a company name, depending on the context. However, word embedding models return the same distributed representation for the same token regardless of the context.

Recently, pre-trained LMs [13, 54] that overcome the above limitation by calculating *contextual word embeddings* were introduced by the NLP community. Pre-trained LMs are trained on large-scale textual corpora such as Wikipedia and possess deep language understanding. Furthermore, unlike conventional deep learning models that are specialized for each task, pre-trained LMs also enable researchers to *fine-tune* a powerful pre-trained LM using training data for the target task. Such fine-tuned pre-trained models, which possess both broad language understanding through Wikipedia and deep domain knowledge through fine-tuning, showed groundbreaking improvements on various NLP tasks compared to the previous state-of-the-art methods. This has spurred the trend of developing better and better pre-trained models and fine-tuning them for downstream applications.

Ditto fine-tunes pre-trained LMs for EM tasks. Contemporaneous to Ditto, [6, 52] also applied pre-trained LMs to EM tasks. Brunner and Stockinger [6] have tested four pre-trained LMs on the ER-Magellan benchmark. Their method is similar to the baseline version of Ditto. As described earlier, Ditto further improves the EM solution using domain knowledge, data augmentation, and summarization. Peeters et al. [52] have tested BERT on the WDC benchmark and showed that a pre-trained LM can further boost its performance through intermediate training, which tailors a pre-trained LM before fine-tuning.

Among the pre-trained LMs, BERT [13] or its variants (e.g., DistillBERT [62] and RoBERTa [36]) are among the most popular choices for pre-trained LMs because of their versatility and high performance. BERT adopts the Transformer-based architecture [69], which is a stack of self-attention layers that calculates distributed representations based on the similarity against all tokens. The stack of multiple self-attention layers produces contextual embeddings of each token in the input. Therefore, the application of Transformer-based architectures to EM tasks can be considered as a natural extension and a more generalized version of deep learning models such as Seq2SeqMatcher or HierMatcher, which extended the attribute-wise soft-alignment technique of DeepMatcher. Another benefit of Transformer-based architecture and the stacked self-attention layers is that we can insert special tokens to incorporate prior knowledge. In Ditto's case, we serialize the original entity by inserting `[COL]` and `[VAL]`, and can also insert entity type tags (e.g., `[MODELNO]`) to incorporate the domain knowledge.

Despite significant advances in EM solutions, we believe there is even more to be done. We outline some possible directions in the next section.

4 DISCUSSIONS: BEYOND ENTITY MATCHING

In practice, for an EM system to be successful, there needs to be a good blocking method alongside the EM system. Furthermore, both blocking and EM systems must be easy to deploy. In particular, if the blocking/EM systems are based on deep learning models, care must be taken to ensure that the requirement on training data should not become an impedance for the adoption of such systems. A successful EM system should also be versatile. Specifically, it should be easy to apply the EM system to different domains, whether it is about matching company records or products. It should also be easy to apply the EM system to match entities *across different domains*, such as matching subsidiaries with its parent company, or a job seeker's profile with candidate job descriptions. In the latter case, it is especially crucial to be able to explain EM outcomes, such as why a job seeker is matched with a job description, and, perhaps even more importantly, explain why a job seeker is *not* matched to a job description. We discuss these topics next.

4.1 Blocking and Entity Matching Together

Blocking is an essential step that occurs before EM, especially when there is a large number of records to match. The goal of blocking is to quickly eliminate from further consideration pairs of records that are known not to match with high certainty, thereby reducing the amount of matching work that needs to be done during the subsequent EM step. The blocking step typically has high recall (i.e., it keeps as many true matching pairs as possible), and this is often achieved by applying heuristics. For example, to match people records, a heuristic blocking rule may remove pairs of records where none of the past addresses of both records overlap in state or city. Through this heuristics, pairs of records that are unlikely matches are removed from further consideration.

Blocking and EM are often considered as independent steps. In reality, they can be synergistic when considered together as a trained matching model can in turn be used in blocking to reduce the number of pairs to match. DeepER [14], for example, used the trained deep model to convert every entity to a vector representation. For each entity e in the first table, DeepER's blocking step retrieves from the second table approximate nearest neighbors that have the highest cosine similarity scores, in vector representations, to e. The number of entities to match for e is thus reduced to the small set of approximate nearest neighbors. AutoBlock [80] is another deep learning model for blocking, which takes a similar approach as DeepER. AutoBlock implements multiple alignment layers for token-level, attribute-level, entity-level comparisons to incorporate finer-grained similarity calculation between entities than DeepER. Zardetto et al. [79] performs EM by clustering entity pairs based on the assumption that the similarity values for matched/unmatched entity pairs follow two different distributions. Their formulation allows any similarity function to be used, and since it does not require labeled data, it can be directly applied to perform EM without the blocking step. ZeroER [77] extended the idea by considering multiple similarity functions and fitting a Gaussian Mixture Model in the multi-dimensional entity-pair space.

In Ditto, we explored a similar idea as DeepER for blocking where we first tune a pre-trained LM Sentence-BERT [58], using labeled data. We then obtain vectors of entities from the trained model and retrieve the top-k approximate nearest neighbors using blocked matrix multiplication [1]. With the trained model for blocking, Ditto is able to reduce the number of candidate matching pairs from 10 million to 0.6 million , where the 10 million pairs are obtained from a basic blocking method that uses heuristic rules. As a consequence, Ditto reduces the total time for matching from 6.49 hours to 1.69 hours on a single-GPU machine. Further investigation can be carried out to combine rules and advanced blocking techniques to improve the overall precision and recall of EM.

4.2 Alleviating Training Data Requirements

While deep learning models have demonstrated promising results for EM, a downside of leveraging deep learning models is that it requires a lot of training data to perform well. To alleviate the cost of obtaining training data, recent work has leveraged active learning [28, 41], transfer learning [83], and data augmentation techniques [35].

Active learning is a technique that selects unlabeled examples that could improve the current model, so the human annotator assigns labels to the selected examples for updating the model. The active learning process is usually conducted in an iterative manner. Although it requires human annotators, one could minimize the labeling cost to obtain the target performance and one could even monitor the performance in a realistic scenario. For example, the active learning system developed by Kasai et al. [28] samples *unclear* instances based on the current EM model while it considers high-confidence predictions as true labels in a semi-supervised manner, so the system can focus on challenging examples to label.

Transfer learning [51] is a technique that "transfers" knowledge learned from a source domain to a target domain so the model can leverage what it has learned from the source domain. The key challenge in transfer learning is to estimate the *transferability* of a model to other domains. AutoEM [83] is a transfer learning framework that trains different DL-based matching models for different attribute types and then adaptively switches to a type-specific matching model for each attribute-pair using a type-detection model. AutoEM achieved better performance than DeepMatcher and Magellan on the ER-Magellan benchmark in low-resourced settings where each method has access to 5% to 20% of the training data. Kasai et al. [28] incorporated adversarial transfer learning [20] into their model. With adversarial transfer learning, their model is optimized to "mislead" a task classifier that aims to identify the original dataset while optimizing the EM accuracy. By doing so, the model is trained to be more robust since the framework facilitates the learning of task-independent representations.

Data augmentation is a technique that synthetically generates new training examples based on a given set of training data using prior knowledge. For example, adding new images of, say, an animal by rotating the existing image at various angles is helpful for ensuring that the trained model is robust at identifying that animal. Good data augmentation can add examples for covering corner cases that are not satisfactorily represented by existing training data. Such generated examples help the trained model to be better at generalizing and tend to stabilize better. Therefore, a key challenge of data augmentation is how to find and incorporate domain knowledge to create useful training examples.

We have seen promising progress along this line in Ditto. For example, with data augmentation, Ditto achieves the same performance as or better performance than DeepMatcher when it is trained on only 20% (respectively, 50%) of the labels used by DeepMatcher for the ER-Magellan benchmark (respectively, the WDC benchmark).

All the techniques discussed above, namely active learning, transfer learning, and data augmentation, are shown to be effective for reducing training data requirements when applied individually. However, it is unclear which technique is applicable to which cases or if we can combine multiple techniques to further reduce the label requirement. We believe that a promising research direction is to investigate whether one can achieve a unified framework for combining the different techniques to further minimize the number of required labels.

4.3 Matching Different Entities

The term "entity matching" could also refer to the more general problem settings, that is, matching entities that are related according to some criteria, but not necessarily identical. The more general problem is referred to as entity linking [7], although in the NLP community, entity linking/matching refers to the problem of determining the identity of entities in text. A specific instance of this problem is to identify mentions in text by linking them to concepts in knowledge bases [64]. Based on this observation, we can broaden the scope of EM and apply existing techniques to more related tasks, for example, whether a job description matches a job seeker's profile, whether a travel package matches a user's search criteria, or whether a company is a subsidiary of another when they do not necessarily share the same name.

Studies on EM and linking are predominantly carried out for structured data where the goal is to determine whether two records in relational tables are identical or not. As the above examples suggest, in addition to matching entities that are not necessarily identical or entities of different types, the real world also requires one to match entities that are described in different data formats: relational tables, semi-structured, or unstructured text documents. Since deep learning models are robust to the format and "cleanliness" of records, they are good candidates for establishing solutions to match entities that reside in different formats, such as semi-structured and unstructured data. At Megagon Labs, we are working towards an extension of Ditto that can be applied to match or link structured entities for different downstream applications. Existing studies primarily focus on similarity joins over semi-structured [21] or unstructured data [71] using manually designed similarity functions and similarity thresholds. Since these approaches rely on heuristics to identify the right similarity functions, they are hard to generalize to other domains or applications. To enable a more general EM solution, several challenges need to be addressed:

- Construct benchmark datasets for evaluating the general EM tasks. Currently, the Magellan project [30] provides large-scale datasets for evaluating EM over structured data. However, there are no benchmark datasets for matching different entities, or matching entities in different data formats.
- Design and develop an effective EM pipeline to support the general matching conditions. In the job matching example, there can be a lot of irrelevant information such as the candidate's name, job posted date, or a long description of the company. Although one could rely on deep learning models to learn to ignore such attributes, by providing more training examples, an effective pre-processing operation can be used to keep only the essential information for matching. Such operations will accelerate the training and prediction process and can also potentially improve the model's overall performance.
- Develop new models to jointly learn the structure and semantics of the data. As shown in a recent study [8], for the task of EM over structured data, deep learning models could

benefit from incorporating the structural information into the token embeddings. Since semi-structured data is inherently more complex in its format, it is even more crucial to incorporate structural information of semi-structured data when performing EM, which will provide the model with knowledge of how elements may be associated through nesting or by association. Ditto's current approach of flattening a record might miss important structure information, such as the correspondence between attributes in a nested structure.

4.4 Explaining Entity Matching Outcomes

As responsible data management takes center stage [3, 46, 65], it has become more and more important to develop *responsible technologies.* One aspect of responsible technologies is to ensure that algorithms are explainable. For the case of EM, this means adding the capability to explain matching outcomes, the reasons two entities are matched, and, even more importantly, the reasons two entities are *not* matched. This is especially crucial when we start to match different entities where the matching decisions may greatly impact the lives of people. Examples of such impactful matching decisions are matching mortgage packages to individuals or jobs to job seekers. In fact, [65] describes an example on the hiring process at length. It states that while hiring technologies are "rarely used to make affirmative hiring decisions, they often automate rejections." Hiring practices (and other impactful decisions such as mortgage applications) are subject to anti-discriminatory laws in many countries, and therefore, it is crucial to ensure that any automation in the hiring process must at least be explainable to establish trust and fairness in the process.

The work of [68] described the need to explain matching outcomes, and the solutions offered by machine learning techniques are generally inadequate for EM. We believe explanations should come at least in two forms: one for end users and the other for computer scientists who can understand the underpinnings of the technology. In both cases, there should be capabilities to provide a global explanation to how a matching decision is made in general and a local explanation tailored for a particular individual (i.e., why a job is matched or not matched to that individual).

Recent work [56, 57] has started towards explainable EM by synthesizing rules from the matching examples to explain the model that is used. The AI community has also considered approximating complex black box models (e.g., neural nets) with simple models (e.g., shallow decision trees) to interpret or explain the outcome of a complex model [17, 29, 39, 59] (see [22] for a detailed survey). Such techniques have also been applied to recommender systems [81] aiming at generating user-friendly textual explanations [40, 72, 78, 82] or identifying related facts (e.g., "customers like X also like Y") to explain the recommendations [24, 63].

Attention weight analysis [11, 26, 75] is the most common way to analyze the behavior of Transformer-based models such as BERT. The visualization of attention weights of the input allows one to understand how the model calculates attention weights (aka relevance scores) between input tokens and how they are aggregated into an attention vector for each token. For NLP tasks, the attention vectors in shallow layers of Transformer-based models are known to capture lexical patterns, while those in deeper layers capture syntactic and semantic meanings [11]. However, attention-based analyses are based on individual examples and cannot be used to explain the general behavior of the model. Furthermore, explanations based on attention analysis can be controversial [26, 75] since interpretations of the attention weights can be arbitrary as the models are likely to have many attention heads and layers.

We believe it is interesting research to understand whether any of the existing techniques described above can be applied to EM solutions based on deep learning or pre-trained LMs, that is, to understand, as a global explanation, how EM decisions are made in general and whether or not explanations can also be achieved at a local level [38].

Bias and fairness in entity matching. Another important aspect to achieve responsible EM is to guarantee *algorithmic fairness* [4, 9, 42, 61]. In the context of EM, one aspect of fairness is the requirement that the matching decisions should be made independent of certain attributes such as race or gender for the job matching scenario. Fairness is difficult to guarantee for deep EM systems because deep learning models implicitly capture unwanted biases (such as gender) from the training data [2, 12, 37]. For the job matching example, a matching model that is trained on data that is heavily biased towards a certain gender or population group is likely to be susceptible to making wrong matching decisions for gender or population groups that are under-represented in the training data. The challenges on training data preparation are thus: (1) how to construct an unbiased labeled training set and (2) how to de-bias the existing labels. In Ditto, this problem can be partially addressed through data augmentation, where data augmentation operators can be applied to randomly perturb gender/race features to ensure that the predictions made are invariant of the two attributes.

5 CONCLUSION

EM has a long history of research. We describe recent work on EM based on deep learning models and pre-trained LMs, which have achieved the state-of-the-art accuracy in EM. We also describe some directions for future work on EM and beyond, including investigating how to combine blocking and EM to further improve accuracy, how to combine existing techniques to further reduce the amount of training data required, generalizing EM solutions to work for a wider range of applications, and explaining matching outcomes.

REFERENCES

[1] Firas Abuzaid, Geet Sethi, Peter Bailis, and Matei Zaharia. 2019. To index or not to index: Optimizing exact maximum inner product search. In *ICDE*. IEEE, 1250–1261.

[2] Alexander Amini, Ava P. Soleimany, Wilko Schwarting, Sangeeta N. Bhatia, and Daniela Rus. 2019. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 289–295.

[3] Ricardo Baeza-Yates. 2018. Bias on the web. *Commun. ACM* 61, 6 (2018), 54–61.

[4] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2017. Fairness in machine learning. *NIPS Tutorial* 1 (2017).

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL* 5 (2017), 135–146.

[6] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures-a step forward in data integration. In *EDBT*. 463–473.

[7] Douglas Burdick, Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang-Chiew Tan. 2016. A declarative framework for linking entities. *ACM Trans. Database Syst.* 41, 3 (2016), 17:1–17:38.

[8] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *SIGMOD*. 1335–1349.

[9] Alexandra Chouldechova and Aaron Roth. 2020. A snapshot of the frontiers of fairness in machine learning. *Commun. ACM* 63, 5 (2020), 82–89.

[10] Peter Christen. 2011. A survey of indexing techniques for scalable record linkage and deduplication. *TKDE* 24, 9 (2011), 1537–1555.

[11] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? An analysis of BERT's attention. In *ACL Workshop BlackboxNLP*. 276–286.

[12] Chris DeBrusk. 2018. The risk of machine-learning bias (and how to prevent it). *MIT Sloan Management Review.*

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*. 4171–4186.

[14] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *PVLDB* 11, 11 (2018), 1454–1467.

[15] Vasilis Efthymiou, George Papadakis, Kostas Stefanidis, and Vassilis Christophides. 2019. MinoanER: Schema-agnostic, non-iterative, massively parallel resolution of web entities. In *EDBT*, Melanie Herschel, Helena Galhardas, Berthold Reinwald, Irini Fundulaki, Carsten Binnig, and Zoi Kaoudi (Eds.). 373–384.

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

[17] Ruth C. Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*. 3429–3437.

[18] Cheng Fu, Xianpei Han, Jiaming He, and Le Sun. 2020. Hierarchical matching network for heterogeneous entity resolution. In *IJCAI*. 3665–3671.

[19] Cheng Fu, Xianpei Han, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. End-to-end multi-perspective matching for entity resolution. In *IJCAI*. AAAI Press, 4961–4967.

[20] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* 17, 59 (2016), 1–35.

[21] Sudipto Guha, H. V. Jagadish, Nick Koudas, Divesh Srivastava, and Ting Yu. 2002. Approximate XML joins. In *SIGMOD*. 287–298.

[22] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM Comput. Surv. (CSUR)* 51, 5 (2018), 1–42.

[23] Sairam Gurajada, Lucian Popa, Kun Qian, and Prithviraj Sen. 2019. Learning-based methods with human-in-the-loop for entity resolution. In *CIKM*.

[24] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. 241–250.

[25] Mauricio A. Hernández and Salvatore J. Stolfo. 1995. The merge/purge problem for large databases. In *SIGMOD*. 127–138.

[26] Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *NAACL-HLT'19*. 3543–3556.

[27] Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *ACL: Human Language Technologies*. 1148–1158.

[28] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. In *ACL*. 5851–5861.

[29] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*. 1885–1894.

[30] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward building entity matching management systems. *PVLDB* 9, 12 (2016), 1197–1208.

[31] Hanna Köpcke and Erhard Rahm. 2010. Frameworks for entity matching: A comparison. *Data Knowl. Eng.* 69, 2 (2010), 197–210.

[32] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *PVLDB* 3, 1–2 (2010), 484–493.

[33] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. 2006. Record linkage: Similarity measures and algorithms. In *SIGMOD*. 802–803.

[34] Chen Li, Jiaheng Lu, and Yiming Lu. 2008. Efficient merging and filtering algorithms for approximate string searches. In *ICDE*. IEEE, 257–266.

[35] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2021. Deep entity matching with pre-trained language models. *PVLDB* 14, 1 (2021). The full version is available at https://arxiv.org/abs/2004.00584.

[36] 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. *arXiv* (2019).

[37] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. 2016. The variational fair autoencoder. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.).

[38] Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* 2 (2020), 56–67.

[39] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *NeurIPS*. 4765–4774.

[40] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 165–172.

[41] Venkata Vamsikrishna Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 2020. A comprehensive benchmark framework for active learning methods in entity matching. In *SIGMOD*. 1133–1147.

[42] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635* (2019).

[43] Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. Snippext: Semi-supervised opinion mining with augmented data. In *WWW*. 617–628.

[44] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *EMNLP*. 404–411.

[45] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.

[46] Don Monroe. 2018. AI, explain yourself. *Commun. ACM* 61, 11 (2018), 11–13.

[47] 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*. 19–34.

[48] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*. 2786–2792.

[49] Felix Naumann and Melanie Herschel. 2010. An introduction to duplicate detection. *Synthesis Lectures Data Manage.* 2, 1 (2010), 1–87.

[50] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In *CIKM*. 629–638.

[51] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *TKDE* 22, 10 (2010), 1345–1359.

[52] Ralph Peeters, Christian Bizer, and Goran Glavaš. [n.d.]. Intermediate training of BERT for product matching. In *DI2KG Workshop at VLDB'20*.

[53] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*. 1532–1543.

[54] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*. 2227–2237.

[55] Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC training dataset and gold standard for large-scale product matching. In *Companion Proc. WWW'19*. 381–386.

[56] Kun Qian, Douglas Burdick, Sairam Gurajada, and Lucian Popa. 2019. Learning explainable entity resolution algorithms for small business data using SystemER. In *Proceedings of the 5th Workshop on Data Science for Macro-modeling with Financial and Economic Datasets*. 1–6.

[57] Kun Qian, Lucian Popa, and Prithviraj Sen. 2019. SystemER: A human-in-the-loop system for explainable entity resolution. *PVLDB* 12, 12 (2019), 1794–1797.

[58] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*. 3982–3992.

[59] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *KDD*. 1135–1144.

[60] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

[61] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional fairness: Causal database repair for algorithmic fairness. In *SIGMOD*. 793–810.

[62] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In *EMC*^{2}.

[63] Amit Sharma and Dan Cosley. 2013. Do social explanations work? Studying and modeling the effects of social explanations in recommender systems. In *WWW*. 1133–1144.

[64] Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.* 27, 2 (2015), 443–460.

[65] Julia Stoyanovich, Bill Howe, and H. V. Jagadish. 2020. Responsible data management. *PVLDB* 13, 12 (2020), 3474–3488.

[66] John R. Talburt and Yinle Zhou. 2013. A practical guide to entity resolution with OYSTER. In *Handbook of Data Quality*. Springer, 235–270.

[67] John R. Talburt and Yinle Zhou. 2015. *Entity Information Life Cycle for Big Data: Master Data Management and Information Integration*. Morgan Kaufmann.

[68] Saravanan Thirumuruganathan, Mourad Ouzzani, and Nan Tang. 2019. Explaining entity resolution predictions: Where are we and what needs to be done? In *HILDA@SIGMOD*. ACM, 10:1–10:6.

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*. 5998–6008.

[70] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity matching: How similar is similar. *PVLDB* 4, 10 (2011), 622–633.

[71] Jin Wang, Chunbin Lin, and Carlo Zaniolo. 2019. MF-join: Efficient fuzzy string similarity join with multi-level filtering. In *ICDE*. 386–397.

[72] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In *ICDM*. IEEE, 587–596.

[73] Zhengyang Wang, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Shuiwang Ji. 2020. CorDEL: A contrastive deep learning approach for entity linkage. In *ICDM*.

[74] Michael J. Welch, Aamod Sane, and Chris Drome. 2012. Fast and accurate incremental entity resolution relative to an entity knowledge base. In *CIKM*. 2667–2670.

[75] Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *EMNLP/IJCNLP*. 11–20.

[76] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's transformers: State-of-the-art natural language processing. *arXiv* (2019).

[77] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity resolution using zero labeled examples. In *SIGMOD*. 1149–1164.

[78] Yao Wu and Martin Ester. 2015. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *WSDM*. 199–208.

[79] Diego Zardetto, Monica Scannapieco, and Tiziana Catarci. 2010. Effective automated object matching. In *ICDE*. 757–768.

[80] Wei Zhang, Hao Wei, Bunyamin Sisman, Xin Luna Dong, Christos Faloutsos, and Davd Page. 2020. AutoBlock: A hands-off blocking framework for entity matching. In *WSDM*. 744–752.

[81] Yongfeng Zhang and Xu Chen. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.

[82] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*. 83–92.

[83] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *WWW*. 2413–2424.

Received October 2020; revised October 2020; accepted October 2020