Generic Entity Resolution Models

Jiawei Tang[*]
American School of Doha
23jtang@asd.edu.qa

Yifei Zuo
University of Science and Technology of China
dune@mail.ustc.edu.cn

Lei Cao
MIT CSAIL/Univeristy of Arizona
lcao@csail.mit.edu

Samuel Madden
MIT CSAIL
madden@csail.mit.edu

Abstract

Entity resolution (ER) – which decides whether two data records refer to the same real-world object – is a long-standing data integration problem. The state-of-the-art results on ER are achieved by deep learning based methods, which typically convert each pair of records into a distributed representation, followed by using a binary classifier to decide whether these two records are a match or a non-match. However, these methods are dataset specific; that is, one deep learning based model needs to be trained or fine-tuned for each new dataset, which is not generalizable and thus we call them *specific ER models*. In this paper, we investigate *generic ER models*, which use a single model to serve multiple ER datasets over different datasets from various domains. In particular, we study two types of generic ER models: Employs foundation models (*e.g.,* GPT-3) or trains a generic ER model. Our results show that although GPT-3 can perform ER with zero-shot or few-shot learning, the performance is worse than specific ER models. Our trained generic ER model can achieve comparable performance with specific ER models, but with much less train data and much smaller storage overhead.

1 Introduction

Entity resolution (ER) (*a.k.a.* record linkage), a fundamental problem of data integration [5] and cleaning [1], has been extensively studied for several decades [7] from different aspects, including: declarative rules [22, 29]; machine learning based methods (or probabilistic methods) [2, 12], deep learning based methods [13, 18, 8]; and crowdsourcing based methods [28]. ER has a wide spectrum of critical applications such as healthcare [7], e-commerce [10], data warehouses [30], etc.

Deep learning for entity resolution. The state-of-the-art results on ER are achieved by deep learning based methods [13, 18, 8]. These methods typically consist of two steps: using a feature extractor (*i.e.,* an encoder) to convert an entity pair into a representation, and then employing a binary classifier to map this representation to a Boolean output as either a match (1) or a non-match (0). These solutions are dubbed as ***specific ER models***, because each model serves only one ER dataset.

Limitations of specific ER models. There are three main limitations.

1. *Need a lot of train data for each new ER dataset.* Existing deep learning based ER solutions, even with pre-trained language models as the encoder (*e.g.,* Ditto [13] uses BERT [6] and RoBERTa [14]), require a lot of labeled train data for each new ER dataset, while labeled train data for ER is expensive to obtain. Although transfer learning [26, 25, 16] and active

[*] Work done while interning at MIT CSAIL.

Table Representation Learning Workshop at NeurIPS 2022.

learning [15] have been studied for ER, sufficient train data is still needed for each ER dataset.

2. *Lack of generalizability.* The ER solution that is specifically trained for one dataset cannot be easily generalized to other datasets.
3. *Large sizes.* It requires to have one specific deep learning model for each dataset, but there can have tens or hundreds of ER datasets, which cause a large storage overhead.

Generic ER models. In this paper, we seek to answer a different question: Whether we can develop ***generic ER models*** that can be used for many datasets from different domains. Recent advances from the NLP community shows that unified models, such as GPT-3 [3] and T5 [21], can well serve many different downstream datasets and tasks, which suggests that it might be feasible for developing generic ER models as well.

Contributions. Our notable contributions for generic ER models are summarized as follows.

1. *Foundation Models.* We explore the power of foundation models, which are essentially giant frozen language models, such as GPT-3, for answering ER questions.
2. *A Trained Generic ER Model.* We train a generic ER model using multiple ER datasets from different domains, based on pre-trained language models (*e.g.,* BERT or RoBERTa).
3. *Empirical Study.* We compare the above methods with specific ER models. Our main empirical findings are: (a) specific ER models outperform GPT-3 on ER, and (b) the trained generic ER model achieves comparable performance with specific models but does not need any train data on a new dataset and has a much smaller size. These results show that generic models shine some light on ER, and will be an interesting direction for both academia and industry.

Organization. Section 2 discusses related work. Section 3 introduces the background of deep learning for ER. Section 4 describes the two generic models, using foundation models or a trained model, for ER. Section 5 presents our empirical findings. Section 6 concludes this paper and discusses future work.

2 Related work

The paper [9] surveys traditional, non-deep learning based, ER methods. In this section, we will focus on recent advances of applying deep learning for ER.

2.1 Word embedding based methods

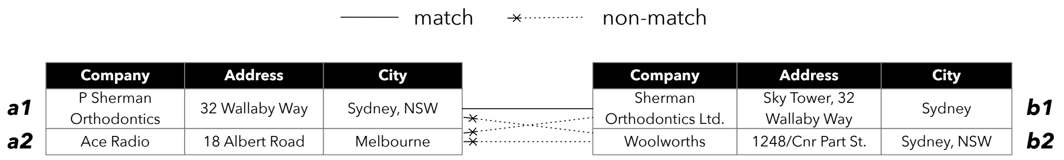
DeepER [8] and DeepMatcher [18] are earlier work of applying deep learning for ER. Both are based on word embeddings, such as GloVe [20] or fastText [17]. Technically, DeepER [8] designs two deep neural networks to extract features of entity pairs, and models ER as a binary classification task, DeepMatcher [18] systematically defines the architecture and design space of DL solutions for ER.

2.2 Pre-trained language models

Ditto [13] first applies pre-trained language models (*e.g.,* BERT or RoBERTa) to ER, which can reduce the number of training data needed. Thanks to the knowledge learned by these pre-trained language models, Ditto outperforms DeepER and DeepMatcher. RPT [23] pre-trains an encoder-decoder model using tabular table and can serve multiple data preparation tasks, including entity resolution, data cleaning, and information extraction.

2.3 Transfer learning and domain adaptation

Domain adaptation, which is a case of transfer learning, is an effective way to reuse labeled source data to different target data, and has been studied for ER. The work [24] introduces a method to re-weight source data and make them adaptable for the target. [11] applies domain adaptation to ER with gradient reverse. A recent work [27] systematically studies domain adaptation for ER, especially comparing various design choices.



	Company	Address	City
a1	P Sherman Orthodontics	32 Wallaby Way	Sydney, NSW
a2	Ace Radio	18 Albert Road	Melbourne

Company	Address	City	
Sherman Orthodontics Ltd.	Sky Tower, 32 Wallaby Way	Sydney	b1
Woolworths	1248/Cnr Part St.	Sydney, NSW	b2

Figure 1: Sample entity resolution problem: match/non-match pairs.

The methods discussed above are specific ER models. That is, they all focus on how to improve the model performance or reduce the size of train data for a specific ER dataset. Different from them, our goal is generic ER models; that is, a single ER model that can be directly used on a new dataset without any train data or with only a few train data.

3 Background

3.1 Entity resolution

Entity resolution. Let A and B be two relational tables with multiple attributes. Each record $a \in A$ (or $b \in B$) is referred to as an entity. The problem of *entity resolution* (ER) is to find all the entity pairs $(a, b) \in A \times B$ that refer to the same real-world objects.

Match/non-match. An entity pair (a, b) is said to be a match (resp. non-match) if they refer to the same (resp. different) real-world objects.

Example 1 *Figure 1 shows two tables, $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$. There is only one match pair (a_1, b_1), while the other three non-match pairs: (a_1, b_2), (a_2, b_1), and (a_2, b_2).* □

Train data for ER. Each train data point is an entity pair (a, b) associated with a 0/1 label. Note that the train data is generally needed for any ER solutions, for discovering ER rules [22, 29], training machine learning based [2, 12] or deep learning based ER models [13, 18, 8].

3.2 Deep learning for entity resolution

Existing deep learning methods for ER typically use a framework that consists of an Encoder and a Matcher, as shown in Figure 2.

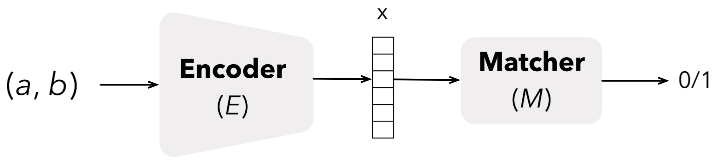


Figure 2: Deep learning for entity resolution.

Specifically, given an entity pair (a, b), an Encoder E will convert this pair into a d-dimensional vector-based representation, denoted by $\mathbf{x}$, *i.e.,* $\mathbf{x} = E(a, b)$. Then, the representation $\mathbf{x}$ will be fed into a Matcher M, which is a deep learning based binary classification model. The Matcher M takes $\mathbf{x}$ as input and outputs a 0/1 label.

Note that, in ER, there are natural language terms, such as company names, addresses, and so on, as shown in Figure 1. Hence, a common strategy is to fine-tune a pre-trained language model (*e.g.,* BERT or RoBERTa) as the Encoder.

Entity serialization. Each entity a with a list of attribute-value pairs $(att_i, val_i)_{1 \leq i \leq k}$ will be serialized into a token sequence as:

$$\text{serialize}(a) = [\text{ATT}]\ \text{att}_1\ [\text{VAL}]\ \text{val}_1\ \ldots\ [\text{ATT}]\ \text{att}_k\ [\text{VAL}]\ \text{val}_k,$$

where [ATT] and [VAL] are two special tokens for the starting of attributes and values respectively.

Example 2 *Entity $a1$ in Figure 1 will be serialized to:*

[ATT] *Company* [VAL] *P Sherman Orthodontics* [ATT] *Address* [VAL] *32 Wallaby Way* [ATT] *City* [VAL] *Sydney, NSW* □

Entity pair serialization. Each entity pair (a, b) will be serialized into a token sequence as:

$$\mathsf{serialize}(a, b) = [\mathsf{CLS}]\ \mathsf{serialize}(a)\ [\mathsf{SEP}]\ \mathsf{serialize}(b)\ [\mathsf{SEP}],$$

where [SEP] is a special token separating the two entities and [CLS] is a special token in BERT to encode the entire sequence.

Example 3 *Entity pair $(a1, b1)$ in Figure 1 will be serialized to:*

[CLS] [ATT] *Company* [VAL] *P Sherman Orthodontics* [ATT] *Address* [VAL] *32 Wallaby Way* [ATT] *City* [VAL] *Sydney, NSW* [SEP] [ATT] *Company* [VAL] *Sherman Orthodontics Ltd.* [ATT] *Address* [VAL] *Sky Tower, 32 Wallaby Way* [ATT] *City* [VAL] *Sydney* [SEP] □

3.3 Specific entity resolution models

We say that an ER model is a *specific model*, if it fine-tunes a pre-trained language model using only one dataset.

Therefore, a specific ER model is typically trained using one dataset and thus can server only one ER dataset.

3.4 Generic entity resolution models

We say that an ER model is a *generic model*, if it can serve many ER datasets from different domains.

Intuitively, a generic ER dataset must be trained using datasets from multiple domains, such that it can learn cross-domain knowledge and then serve multiple ER datasets.

4 Methods

We study two types of generic ER models: Employs foundation models (*e.g.,* GPT-3) or trains a generic ER model using the model architecture as shown in Figure 2.

4.1 GPT-3 for entity resolution

GPT-3 [3] is an autoregressive language model that uses deep learning to produce human-like text, which has shown remarkable results on many tasks. Here, we investigate whether GPT-3 can decide a pair of entities is a match.

To this purpose, given a pair of entities (a, b), we need to turn this pair into a text prompt and then ask GPT-3. Let a be a list of attribute-value pairs $(att_i^a, val_i^a)_{1 \leq i \leq m}$, and b be a list of attribute-value pairs $(att_j^b, val_j^b)_{1 \leq j \leq n}$, the entity pair will be serialized into a prompt as:

$$\begin{aligned} \mathsf{prompt} = &\textit{A is } att_1^a : val_1^a, \ldots, att_m^a : val_m^a \\ &\textit{B is } att_1^b : val_1^b, \ldots, att_n^b : val_n^b \\ &\textit{Are A and B the same?} \end{aligned}$$

For each entity pair, we will feed its corresponding prompt to ask GPT-3. Under the zero-shot setting, GPT-3 can answer questions like “A and B are not the same”, “No, A and B are not the same”, or simply “No”. Note that a recent work [19] uses similar prompts for ER when interacting with foundation models.

Example 4 *Consider entity pair* $(a1, b1)$ *in Figure 1, we will change it into a prompt as below to ask GPT-3:*

prompt = *A is Company: P Sherman Orthodontics, Address: 32 Wallaby Way, City: Sydney, NSW*
B is Company: Sherman Orthodontics Ltd., Address: Sky Tower, 32 Wallaby Way, City: Sydney
Are A and B the same?

Given the above prompt, in the zero-shot setting, GPT-3 will return "Yes", indicating that $(a1, b1)$ *is a match.* □

4.2 A trained generic entity resolution model

We employ the same deep learning architecture as shown in Figure 2. We use a lot of ER benchmarks from different domains to train the model (see Section 5.2.1 for more details). What we want to test is whether the trained model can be directly applied to new ER datasets with zero-shot or few-shot learning.

5 Experiments

5.1 GPT-3 for entity resolution

5.1.1 Experimental setting

Datasets. We randomly used four ER benchmarks, Restaurant1, Bikes, Movies1, and Books, from the Magellan Data Repository [4]. Table 1 provides statistics of the used datasets.

Table 1: Statistics of ER datasets used for GPT-3.

Dataset	#-Categorical	#-Numerical	#-Temporal	Missing Values	#-labels (#-positive/#-negative)
Restaurant1	3	0	0	None	450 (124/326)
Bikes	3	2	0	None	450 (130/320)
Movies1	2	0	1	None	600 (190/410)
Books	7	1	1	Many	397 (92/305)

Zero-shot and few-shot. We used the Davinci model of GPT-3[2]. For zero-shot learning, we use the method discussed in Section 4.1 to generate the prompt for each entity pair and ask GPT-3. For few-shot learning, we randomly sample two positive examples (*i.e.,* matched entity pairs) and two negative examples (*i.e.,* non-matches entity pairs), serialize each example as discussed in Section 4.1 and adds the true label as one-shot, and combine the above serialized examples and the question prompt to ask GPT-3 (*i.e.,* 4-shot learning).

Along with the prompt, another important setting for GPT-3 is Temperature, which ranges within $[0, 1]$ and controls how much randomness is in the output. Temperature $= 0$ eliminates randomness and GPT-3 will always produce the same output for a given prompt. Setting Temperature $= 1$ will deliver very inconsistent and sometimes interesting results. Which Temperature value to use is application dependent.

Evaluation metrics. Let TP be true positives, FP be false positive, and FN be false negatives. We use Precision, Recall, and F1-score to measure the result of GPT-3, which are standard to measure ER solutions [12, 18] and are defined as:

$$\begin{aligned} \mathsf{Precision} &= \mathsf{TP}/(\mathsf{TP} + \mathsf{FP}) \\ \mathsf{Recall} &= \mathsf{TP}/(\mathsf{TP} + \mathsf{FN}) \\ \mathsf{F1\text{-}score} &= (2 \times \mathsf{Precision} \times \mathsf{Recall})/(\mathsf{Precision} + \mathsf{Recall}) \end{aligned}$$

[2] `https://beta.openai.com/docs/models/gpt-3`

5.1.2 Experimental result

Experiment 1: zero-shot by varying temperature. In the first group of experiments, we tested the performance of GPT-3 for ER by varying the Temperature value in 0, 0.3, 0.6, 0.8 and 1. The best results are highlighted in bold.

Table 2: Zero-shot of GPT-3 by varying temperature (P: precision; R: recall; F1: F1-score).

Temperature	0			0.3			0.6			0.8			1		
Dataset	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Restaurant1	.99	**.92**	**.95**	**1**	.89	.94	.97	.82	.89	.96	.76	.85	.91	.82	.86
Bikes	**1**	.1	.18	**1**	.12	.22	.82	.18	.29	.56	.19	.29	.54	**.25**	**.34**
Movies1	**.71**	.37	.49	.63	.48	.54	.47	.58	.52	.46	.65	.54	.46	**.70**	**.56**
Books	**.63**	**.60**	**.62**	.56	.49	.52	.54	.49	.51	.47	.51	.49	.37	.57	.45

Table 2 shows that the relationship between Temperature and F1-score is not consistent, similar to Precision and Recall. For example, The F1-score will decrease when increasing the Temperature values for Restaurant1 and Books, but the F1-score will increase along with increasing the Temperature values for Bikes and Movies1.

Hence, the next question is whether ensembling the results from multiple Temperature values can achieve a better result. Because ER is a Boolean question, the result is either match or non-match. We use majority voting on the five Temperature values, as the ensembled result from GPT-3. The result is shown in Table 3.

Table 3: Zero-shot of GPT-3 by ensembling results from multiple temperature values.

Temperature	Ensembled Result		
Dataset	P	R	F1
Restaurant1	.99	.88	.93
Bikes	**1**	.09	.17
Movies1	.65	.6	**.63**
Books	.58	.53	.58

Table 3 shows that the effect of ensemble is not obvious. Only for Movies1, the ensembled F1-score outperforms the best F1-score in Table 2, which suggests that ensemble should be used in practice.

Summary 1. A safe choice is to set Temperature = 0, *i.e.,* always picking the answer with the highest probability, which will be used by default in the rest of the paper.

Experiment 2: few-shot learning. Next we test few-shot learning of GPT-3 for ER, as discussed in Section 5.1.1. The goal is to test the hypothesis that *if few-shot learning is always better than zero-shot learning for the ER problem.* We show sample few-shot learning results of GPT-3 in Table 4 for Restaurant1 and in Table 5 for Movies1.

Table 4: Sample few-shot results of Restaurant1.

GPT-3 Result	**True Label**
0	1
0	0
0	0
0	0
0	0
0	1
0	1
0	0
0	1
0	0

Table 5: Sample few-shot results of Movies1.

GPT-3 Result	**True Label**
1	1
1	1
1	1
1	0
1	0
1	1
1	0
1	1
1	0
1	0

The results show that GPT-3 can be easily biased with few-shot learning, which is dataset relevant. For example, in Table 4, GPT-3 predicts all ER pairs as non-matches (*i.e.,* 0), while in Table 5, GPT-3 predicts all ER pairs as matches (*i.e.,* 1). One possible reason is that few-shot learning is mainly for better task reasoning, not for "similar" data understanding. The ER problem is a Boolean problem, which is very easy to understand. Providing more examples cannot help GPT-3 to better understand the task; instead, these examples may even mislead GPT-3.

Summary 2. Few-shot learning is not suitable for using GPT-3 for ER.

Experiment 3: GPT-3 vs. specific models. Next, we compare GPT-3 with specific models (see Section 3.3) trained with a small number of labels. For the four datasets used in Table 1, we split the labeled data as train/validate/test with the ratio 3/1/1. The performance comparison of specific models with GPT-3 are shown in Table 6. Note that for a fair comparison, the GPT-3 results in Table 6 are tested using the same test data as specific models.

Table 6: GPT-3 (Temperature = 0, zero-shot) vs. specific models.

Temperature	GPT-3			Specific Models		
Dataset	P	R	F1	P	R	F1
Restaurant1	**.99**	.91	**.95**	0.94	**.94**	.94
Bikes	**1**	.1	.18	.77	**.55**	**.64**
Movies1	.71	.37	.49	**.98**	**.96**	**.97**
Books	.63	.60	.62	**1**	**1**	**1**

Summary 3. Specific ER models, even only being trained with a small number of train data (*e.g.,* a few hundred), can clearly outperform GPT-3 for the problem of ER.

5.2 A trained generic entity resolution model

5.2.1 Experimental setting

Datasets. For testing, we used the same four ER datasets as shown in Table 1. For training, we used 16 datasets that are different from the test datasets from the Magellan Data Repository [4]. We mix the labeled train data from these 16 datasets and train the model as described in Section 4.2.

Evaluation metrics. We also use Precision, Recall, and F1-measure.

Zero-shot, fine-tuning and delta-tuning. We consider different variants of the trained generic ER model.

- Zero-shot means that we directly apply the trained ER model on a new ER dataset.
- Fine-tuning is to use a small set of train data in the tested dataset for fine-tuning the model.
- Delta-tuning is to add a small delta network using a MLP and freezes the Encoder and Matcher, as shown in Figure 3 .

For both Fine-tuning and Delta-tuning, we used 1/3 of labeled data for training, and the rest for testing. Different from Fine-tuning, Delta-tuning is designed to avoid the catastrophic forgetting phenomena when a trained model learns new knowledge but performs badly on the tasks it was trained before.

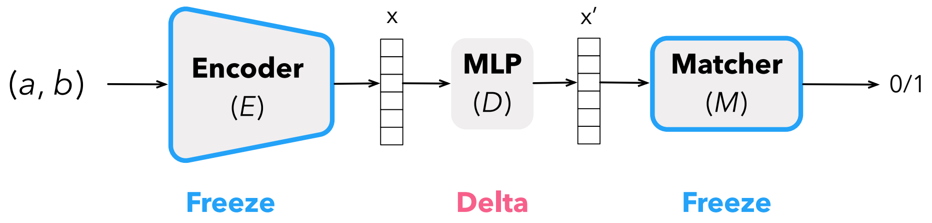


Figure 3: Generic entity resolution with a delta network for fine-tuning.

5.2.2 Experimental result

Experiment 4: generic models with zero-shot vs. specific models. The results of generic models and specific models are shown in Table 7. In this table, we only highlight the best F1-score.

Table 7: Generic Models with zero-shot vs. specific models.

Temperature	Generic Models			Specific Models		
Dataset	P	R	F1	P	R	F1
Restaurant1	.98	.94	**.96**	0.94	.94	.94
Bikes	.41	.81	.54	.77	.55	**.64**
Movies1	.87	.98	.92	.98	.96	**.97**
Books	.83	1	.91	1	1	**1**

Summary 4. Generic ER is slightly worse than, but comparable with, specific models. Note, however, that if there are 10 ER datasets, specific models need 10x storage of a generic model. Hence, a generic model is a good trade-off between effectiveness and efficiency.

Experiment 5: generic models with fine-tuning and delta-tuning vs. specific models. The results of generic models with Fine-tuning and Delta-tuning, and those of specific models, are shown in Table 8. In this table, we only highlight the best F1-score, for simplicity.

Table 8: Generic models with fine-tuning and delta-tuning vs. specific models.

Temperature	Generic Models (Fine-tuning)			Generic Models (Delta-tuning)			Specific Models		
Dataset	P	R	F1	P	R	F1	P	R	F1
Restaurant1	.98	.96	**.97**	.94	.98	.96	.94	.94	.94
Bikes	.55	.86	.67	.56	.88	**.69**	.77	.55	.64
Movies1	1	.98	**.99**	1	.98	**.99**	.98	.96	.97
Books	.97	1	.99	.94	1	.97	1	1	**1**

Summary 5. Both Fine-tuning and Delta-tuning can achieve comparable or even better performance than specific models with a small number of train data. However, as discussed above, Fine-tuning may cause catastrophic forgetting for the generic model. Hence, we would recommend to use Delta-tuning to quickly adapt the generic model to a specific task.

6 Conclusion

In this paper, we have proposed methods for training generic ER models, where one model can be used for many ER datasets from different domains. We have first shown that GPT-3 can work for ER, but only to some extent, which is worse than deep learning models trained for ER tasks. We have further explored the opportunity of training one generic model and shown that it can achieve comparable performance with specific models, but with a much smaller size. In addition, we have proposed two methods to further tune a trained generic model, namely Fine-tuning and Delta-tuning; both achieve comparable or even better results than specific models but with only a small number of train data. If one still wants the ER model to perform well on previously trained ER datasets, then Delta-tuning should be used.

References

[1] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endow.*, 9(12):993–1004, 2016.

[2] M. Bilenko, R. J. Mooney, W. W. Cohen, P. Ravikumar, and S. E. Fienberg. Adaptive name matching in information integration. *IEEE Intell. Syst.*, 18(5):16–23, 2003.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child,

A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[4] S. Das, A. Doan, P. S. G. C., C. Gokhale, P. Konda, Y. Govind, and D. Paulsen. The magellan data repository. `https://sites.google.com/site/anhaidgroup/projects/data`.

[5] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org, 2017.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] H. L. Dunn. Record linkage. *American Journal of Public Health*, 36(12):1412–1416, 1946.

[8] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[9] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[10] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu. Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 601–612, 2014.

[11] J. Kasai, K. Qian, S. Gurajada, Y. Li, and L. Popa. Low-resource deep entity resolution with transfer and active learning. In A. Korhonen, D. R. Traum, and L. Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5851–5861. Association for Computational Linguistics, 2019.

[12] P. V. Konda. *Magellan: Toward building entity matching management systems*. The University of Wisconsin-Madison, 2018.

[13] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60, 2020.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[15] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017.

[16] M. Loster, I. K. Koumarelas, and F. Naumann. Knowledge transfer for entity resolution with siamese neural networks. *ACM J. Data Inf. Qual.*, 13(1):2:1–2:25, 2021.

[17] L. Mouselimis. *fastText: Efficient Learning of Word Representations and Sentence Classification using R*, 2022. R package version 1.0.2.

[18] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34, 2018.

[19] A. Narayan, I. Chami, L. Orr, and C. Ré. Can foundation models wrangle your data?, 2022.

[20] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[21] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[22] R. Singh, V. V. Meduri, A. Elmagarmid, S. Madden, P. Papotti, J.-A. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. Synthesizing entity matching rules by examples. *Proceedings of the VLDB Endowment*, 11(2):189–202, 2017.

[23] N. Tang, J. Fan, F. Li, J. Tu, X. Du, G. Li, S. Madden, and M. Ouzzani. RPT: relational pre-trained transformer is almost all you need towards democratizing data preparation. *Proc. VLDB Endow.*, 14(8):1254–1261, 2021.

[24] S. Thirumuruganathan, S. A. P. Parambath, M. Ouzzani, N. Tang, and S. Joty. Reuse and adaptation for entity resolution through transfer learning. *arXiv preprint arXiv:1809.11084*, 2018.

[25] S. Thirumuruganathan, S. A. P. Parambath, M. Ouzzani, N. Tang, and S. R. Joty. Reuse and adaptation for entity resolution through transfer learning. *CoRR*, abs/1809.11084, 2018.

[26] J. Tu, J. Fan, N. Tang, P. Wang, C. Chai, G. Li, R. Fan, and X. Du. Domain adaptation for deep entity resolution. In Z. Ives, A. Bonifati, and A. E. Abbadi, editors, *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 443–457. ACM, 2022.

[27] J. Tu, J. Fan, N. Tang, P. Wang, C. Chai, G. Li, R. Fan, and X. Du. Domain adaptation for deep entity resolution. In Z. Ives, A. Bonifati, and A. E. Abbadi, editors, *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 443–457. ACM, 2022.

[28] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *arXiv preprint arXiv:1208.1927*, 2012.

[29] J. Wang, G. Li, J. X. Yu, and J. Feng. Entity matching: How similar is similar. *Proc. VLDB Endow.*, 4(10):622–633, 2011.

[30] W. E. Winkler. Data quality in data warehouses. In J. Wang, editor, *Encyclopedia of Data Warehousing and Mining, Second Edition (4 Volumes)*, pages 550–555. IGI Global, 2009.