Automatic Error Correction Using the Wikipedia Page Revision History

Md Kamrul Hasan
Technische Universität Berlin
Berlin, Germany
m.hasan@tu-berlin.de

Mohammad Mahdavi
GISMA University of Applied Sciences
Potsdam, Germany
mohammad.mahdavi@gisma.com

ABSTRACT

Error correction is one of the most crucial and time-consuming steps of data preprocessing. State-of-the-art error correction systems leverage various signals, such as predefined data constraints or user-provided correction examples, to fix erroneous values in a semi-supervised manner. While these approaches reduce human involvement to a few labeled tuples, they still need supervision to fix data errors. In this paper, we propose a novel error correction approach to automatically fix data errors of dirty datasets. Our approach pretrains a set of error corrector models on correction examples extracted from the Wikipedia page revision history. It then fine-tunes these models on the dirty dataset at hand without any required user labels. Finally, our approach aggregates the fine-tuned error corrector models to find the actual correction of each data error. As our experiments show, our approach automatically fixes a large portion of data errors of various dirty datasets with high precision.

CCS CONCEPTS

• **Information systems** → **Data cleaning**; • **Theory of computation** → *Data integration.*

KEYWORDS

data cleaning, error correction, edit distance, fastText

ACM Reference Format:
Md Kamrul Hasan and Mohammad Mahdavi. 2021. Automatic Error Correction Using the Wikipedia Page Revision History. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482062

1 INTRODUCTION

Data cleaning is a crucial task in any data science pipeline. According to CrowdFlower's Data Science Report 2016, data scientists spend 60% of their time in data analytics to clean and organize datasets [1]. Data cleaning aims to detect and correct erroneous values in datasets [16]. Due to the challenges and the time-consuming nature of data cleaning, data scientists usually consider it as a two-step task: (1) error detection, which identifies erroneous values in the dirty dataset, and (2) error correction, which replaces the erroneous values with correct values [16].

State-of-the-art error detection and correction systems are semi-supervised [16, 17]. They require a few labeled tuples from the user to train error detectors/correctors. In fact, they take a few example data errors and their correction from the user to mark and fix the rest of the data errors by generalizing error detection/correction operations [9, 16, 17]. These approaches are promising as they achieve high performance using only a few labeled tuples. However, at the same time, they admit the fact that limited human involvement in data cleaning is inevitable [16]. That is, fully automated data cleaning is still hard to achieve.

Recently, there has been a new line of research to automatically detect a portion of data errors of dirty datasets with high precision [10, 22]. These unsupervised approaches learn typical patterns of clean values from a huge set of web tables to identify erroneous values of a given dirty dataset that do not comply with those typical learned patterns. Of course, without any human supervision, these unsupervised approaches can only detect a portion of data errors of a given dataset. However, as long as the error detection approach is accurate and automated, it is still highly valuable as "any recall for free is better than no recall" [22].

While automated error detection approaches have been already studied [10, 22], studying automatic error correction has been mainly neglected. Error correction is more challenging than error detection due to its search space [16]. While error detection is a binary classification task, where each data value is classified as clean or dirty, the search space of error correction is theoretically infinite. That is, potentially any possible string could be the correction of an erroneous value. Therefore, without any human supervision, this challenging task seems even more complex.

In this paper, we propose a new unsupervised error correction approach to automatically fix as many data errors of dirty datasets as possible. The intuition is that we can learn prevalent syntactic and semantic error correction operations from historical revision logs and then apply them to new dirty datasets. To extract error correction examples, we leverage the Wikipedia page revision history as it has been shown to be a rich resource for training error corrector models [16]. We pretrain various error corrector models on the Wikipedia page revision history and then fine-tune them on the given dirty dataset. The fine-tuned error corrector models propose various correction candidates for each data error of the dirty dataset. In contrast to the semi-supervised error correction systems [16], our unsupervised approach automatically aggregates the error correction models to replace erroneous values of dirty

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482062

datasets with the most likely correction candidates. While, without any human supervision, our approach naturally can replace only a portion of erroneous values, these automatic corrections are highly accurate. Therefore, we make the following contributions:

- We propose a new unsupervised error correction approach to automatically fix syntactic and semantic data errors.
- We show how to turn the edit distance [15] and fastText [11] models into general-purpose error corrector models. We propose a method to effectively pretrain these models on the Wikipedia page revision history and then fine-tune and aggregate them on the dirty datasets.
- We thoroughly evaluate our unsupervised error correction approach via extensive experiments on real-world and synthetic datasets.

2 AUTOMATIC ERROR CORRECTION

Figure 1 shows the workflow of our approach. The input to our approach is a dirty dataset. In accordance with other error correction systems [16, 20], we assume erroneous cells of this dirty dataset are already marked by an upstream error detection step. Our error correction approach aims at automatically fixing as many of these detected data errors as possible with high precision.

As shown in Figure 1, our approach has three main phases: (1) extracting revision data, (2) pretraining and fine-tuning models, and (3) aggregating models.

2.1 Revision Data Extraction

Wikipedia is a free encyclopedia where anyone can access and edit a wiki page. When a piece of information is edited by users, Wikipedia keeps its edit history alongside its previous incorrect versions. All these revision histories are stored in the Wikipedia dump files [2]. The Wikipedia page revision history is an information-rich resource for the error correction task. The value-based corrections extracted from these revision logs have been shown to be effective for pretraining and aggregating error corrector models in a semi-supervised manner [16]. However, in our unsupervised error correction scenario, learning value-based corrections, such as replacing "US" with "United States", is not enough. Since our approach aims at automatically correcting data errors, it needs to also extract the context of value-based corrections from the Wikipedia revisions to learn, in which contexts, a potential correction would be most likely accurate.

Our approach compares any subsequent revision versions of Wikipedia pages to extract correction examples along with their context from the Wikipedia tables and infoboxes. In particular, our approach extracts the erroneous value itself, its user-revised correct value, and the contextual values in adjacent rows and columns of Wikipedia tables and infoboxes. These extracted revision data are stored as structured JSON files. This way, we can pretrain context-aware error corrector models.

2.2 Pretraining and Fine-Tuning Models

Same as state-of-the-art error correction systems, our approach aims at fixing both syntactic data errors, such as typos, and semantic data errors, such as the wrong usage of "Holland" instead of "Netherlands" [16]. Thus, we need to leverage error corrector models that can theoretically capture and fix both of these general data error types.

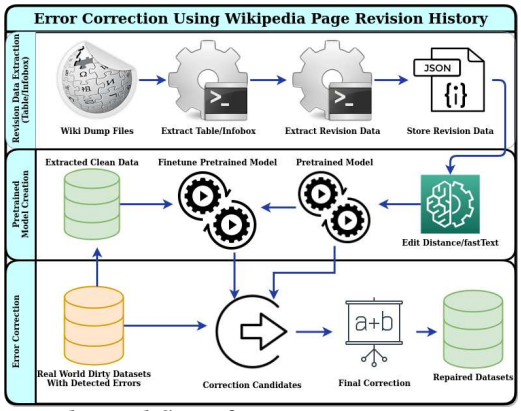


Figure 1: The workflow of our error correction approach.

A couple of error corrector models have been proposed so far for fixing syntactic and semantic data errors [16]. For example, substring adder models can add substring "eet" to an erroneous value "Str" to generate the correct value "Street". Functional dependencies can also be considered as error corrector models that can fix semantic data errors based on their contextual values. However, these error corrector models are not suitable for our *unsupervised* error correction scenario as they need human supervision. The user has to provide some example corrections so that these models can learn which substring adder operations and functional dependencies are valid for data errors of this particular dataset [16].

Therefore, in the lack of human supervision, we have to choose more generic families of error corrector models that can fix only general syntactic and semantic data errors but in an automatic way. We turn the edit distance [15] and fastText [11] models, which can capture the syntactic and semantic similarity of values respectively, into error corrector models. Even in the lack of any human supervision, we can still decide to whether accept their proposed corrections or not based on their correction likelihood.

Edit distance. Edit distance refers to the minimal operations, i.e., insertions, deletions, and substitutions, that are needed to transform one string into another [15]. It is widely used as a measure to fix spelling errors [5]. Edit distance is useful to fix syntactic errors when a dataset preserves redundant values.

We design an error corrector model based on the edit distance measure to fix syntactic errors. For each erroneous value, edit distance proposes correction candidates from a trained pool of candidates. A probability score $P_{\text{ed}}(c|e)$ is assigned to each correction candidate c of a data error e as

$$P_{\text{ed}}(c|e) = \frac{count(c, V)}{|V|}, \tag{1}$$

where $count(c, V)$ is the frequency of the correction candidate c in the pool of correction candidates V and $|V|$ is the total number of correction candidates in the pool.

The correction candidate pool is generated in the pretraining phase and then fine-tuned on the current dataset. In the pretraining phase, we leverage the extracted value-based corrections of the

Wikipedia page revision history to collect the most similar corrections of each erroneous value in terms of edit distance. In the fine-tuning phase, we enrich this default correction candidate pool by adding the most similar clean values in the same data columns of the data error.

fastText. Word embedding approaches transform words into dense vectors of floating-point values [19]. One of the main benefits of these vectors is that they can represent the semantic similarity of textual values. For example, the semantically similar values "cat" and "dog" will have similar vector representations. Word2vec [19] is one of the most popular word embedding models that have been used for error detection as well [12]. fastText extends word2vec by utilizing the substring- and sentence-level information [11]. In addition to words, fastText can represent substrings of words by vectors. Furthermore, supervised fastText can be used for text classification problems, such as spam detection and sentiment analysis.

We use fastText as an error corrector model to remove semantic errors. In particular, we leverage its sentence classification technique to capture the correlation between values of different columns. In the pretraining phase, the fastText model trains on the extracted web tables and infoboxes of the Wikipedia page revision history. In fact, the fastText model learns to predict the corrected value of an erroneous cell using substrings of the contextual values. For example, when an erroneous value "MerCel" has a neighboring data cell "Chancellor, Germany, CDU", fastText can propose the correction candidate "Merkel" with high probability as this value has a similar vector representation to the neighboring substrings "Chancellor", "Germany", and "CDU". In the fine-tuning phase, we update the pretrained supervised fastText model in two ways. If the dirty dataset has predefined data constraints, such as functional dependencies, we update the pretrained fastText classifier based on each predefined functional dependency, where the model learns to predict the dependent target attribute based on the independent feature attributes. In the lack of predefined data constraints, we fine-tune the fastText classifier for each attribute of the dataset. In fact, the fastText model learns to predict each attribute based on other attributes.

fastText represents each contextual value and each potential correction candidate as a vector in a multidimensional space. Therefore, fastText can calculate the probability score for each correction candidate based on the similarity of the context vector and the correction candidate vector. Formally, it assigns the probability score $P_{\text{ft}}(c|e)$ to each correction candidate c of a data error e as

$$P_{\text{ft}}(c|e) = \frac{\exp\big(y(c)\big)}{\sum_{c'} \exp\big(y(c')\big)}, \tag{2}$$

where $y(c)$ is the dot product of the context vector and the correction candidate vector.

2.3 Model Aggregation

The pretrained and fine-tuned edit distance and fastText error corrector models propose various syntactic and semantic correction candidates for each data error. Now, we need to aggregate these correction candidates into one final correction for each data error. The main challenge here is that, since our approach is designed to be unsupervised, we cannot ask the user to provide feedback to learn the best aggregation function [16].

In the lack of user supervision, we have to resort to aggregation formulas to combine our two probabilistic error corrector models in a way that we can control their corresponding weight. For each correction candidate c of a data error e, we compute the aggregated correction probability as

$$P(c|e) = z \times P_{\text{ed}}(c|e) + (1 - z) \times P_{\text{fd}}(c|e), \tag{3}$$

where z is the aggregation coefficient. Since in our automated error correction setting the precision of error correction is important, we also define a cutoff threshold θ to specify the minimum acceptable correction probability. In fact, our approach only selects the most probable correction candidate c^* of a data error e if $P(c^*|e) >= \theta$.

3 EVALUATION

3.1 Experimental Setup

Datasets. We evaluate our approach on three different datasets. Hospital [20] is a small dataset with 1000 rows and 20 columns. It has a lot of duplicate tuples and a relatively low error rate (3%). The main data error types of this dataset are typos and violated functional dependencies. Tax [3] is a synthetic large dataset with 200000 rows and 15 columns. It has data errors of formatting issues, typos, and violated functional dependencies. Flights [16, 20] is a real-world dataset with 2376 rows and 7 columns. Its high data error rate (30%) is mainly due to the violated functional dependencies.

Evaluation measures. We evaluate the correctness and completeness of our unsupervised error correction approach by reporting precision, recall, and F_1 score [16, 20]. It is worth emphasizing that, since our approach is automated, our main goal is to maximize precision, similar to unsupervised error detection approaches [10, 22]. Nevertheless, we also show that recall of our approach is also high when data redundancy is preserved.

Baselines. We compare our system to three baselines:

(1) Baran [16] is a semi-supervised data cleaning approach. We ran this approach with different numbers of user labels.
(2) HoloClean [20] repairs dirty datasets holistically by combining integrity constraints, matching dependencies, and external knowledge bases. We ran this system with and without providing data constraints.
(3) Katara [6] repairs datasets by cross-checking them with external knowledge bases. We connected Katara to the DBpedia [4] knowledge base.

Our default setting. By default, our approach leverages both edit distance and fastText error corrector models. They have been pretrained on 100000 web tables and infoboxes from the Wikipedia page revision history. To aggregate these models, we set the aggregation coefficient $z = 0.5$ and cutoff threshold $\theta = 0.75$. We assume there are no predefined user-provided data constraints for dirty datasets. Our prototype, including further experiments and results, is available online[1].

[1] https://github.com/hasantuberlin/AutoECUWRH

Table 1: Comparison with baselines.

Approach	Tax			Flights			Hospital		
	P	R	F_1	P	R	F_1	P	R	F_1
Baran (With 0 Labels)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HoloClean (Without Denial Constraints)	N/A[+]	N/A	N/A	0.79	0.38	0.51	0.97	0.90	0.93
Katara (With DBpedia Knowledge Base)	0.59	0.01	0.02	0.00	0.00	0.00	0.98	0.24	0.29
Baran (With 20 Labels)	0.84	0.78	0.81	1.00	1.00	1.00	0.88	0.86	0.87
HoloClean (With Denial Constraints)	0.11	0.11	0.11	0.81	0.39	0.52	0.97	0.90	0.93
Our Unsupervised Approach	**1.00**	0.02	0.01	**0.97**	0.05	0.10	**1.00**	0.45	0.62
Our Correctly Fixed Data Errors		1200			265			229	

[+] HoloClean did not terminate after running for three hours.

Table 2: The effect of pretraining/fine-tuning of the models.

Model	Tax			Flights			Hospital		
	P	R	F_1	P	R	F_1	P	R	F_1
Edit Distance Pretrained	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Edit Distance Fine-tuned	0.53	0.02	0.03	0.05	0.02	0.02	0.78	0.59	0.67
fastText Pretrained	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fastText Fine-Tuned with Data Constraints	1.00	0.01	0.01	1.00	0.83	0.90	1.00	0.28	0.43
fastText Fine-Tuned without Data Constraints	0.20	0.01	0.01	0.87	0.14	0.24	0.84	0.39	0.53
Both Models Pretrained and Fine-Tuned	**1.00**	0.02	0.01	**0.97**	0.05	0.10	**1.00**	0.45	0.62

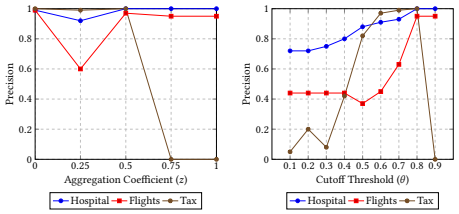
3.2 Experimental Results

Comparison with baselines. Table 1 shows that, in the lack of human supervision, our approach outperforms all the baselines in terms of precision. Although our automated error correction approach maximizes precision, we also report the number of correctly fixed data errors for each dataset to highlight the benefits of our approach. Since our approach is unsupervised but some of our baselines need human supervision, we ran those semi-supervised approaches in two modes: with their default requirements and also without any user supervision. This way, we can have a fair comparison between the semi-supervised approaches and our unsupervised error correction approach.

The baselines cannot achieve high precision in the unsupervised setting as they need some sort of human supervision. Without user-provided correction examples, Baran cannot fix any data error. Without user-provided data constraints, HoloClean cannot always achieve its original precision. With a general knowledge base like DBpedia, Katara cannot fix data errors accurately as the user did not create a domain-related knowledge base for each particular dataset. Of course, human supervision, i.e., a few labeled tuples for Baran and user-provided data constraints for HoloClean, improves the performance of our baselines. Nevertheless, even when our baselines leverage human supervision, our unsupervised approach outperforms them in terms of precision on the tax and hospital datasets and achieves a competitive precision on the flights dataset.

Pretraining and fine-tuning. Table 2 shows the effectiveness of our approach when it uses different pretrained or fine-tuned error corrector models. Our approach achieves high precision when the error corrector models are both pretrained and fine-tuned. We get a precision score of 1.00 for all datasets by using just the fine-tuned supervised fastText model with data constraints. However, data constraints might not always be provided for dirty datasets. It is promising that our unsupervised approach can achieve almost the same precision without using any data constraint when it aggregates pretrained and fine-tuned edit distance and fastText models.

Model aggregation. Figure 2 shows the impact of the aggregation parameters on the overall performance of our approach. Naturally, increasing θ increases precision as we set a higher cutoff threshold for our approach. Our approach achieves relatively the same precision with different z values on all datasets except tax. On this dataset, we get a precision score of 0.0 when $z > 0.5$. Based on Equation 3, when $z > 0.5$, the correction candidates of the edit distance model receive higher weights. However, the edit distance model cannot fix data errors of this dataset because the probability scores of its proposed correction candidates are below the cutoff threshold. Therefore, these correction candidates are discarded.



(a) Aggregation Coefficient (z) (b) Cutoff Threshold (θ)

Figure 2: Precision with different (a) z and (b) θ.

4 RELATED WORK

We aim at correcting erroneous values of dirty datasets automatically. Hence, we explore the recent data cleaning approaches that either addressed the error correction task or are automatic.

Error correction. State-of-the-art error correction systems combine multiple internal signals to learn corrections [16, 20]. Baran [16] is a semi-supervised data cleaning system, which holistically combines different error correction models. The user needs to provide correction examples for this system to train classifiers that predict the correction of each data error. HoloClean [20] also unifies repairing signals, such as integrity constraints and matching dependencies. The user needs to encode data constraints and link the dataset to relevant external sources. Other error correction approaches also rely on user-provided data constraints [7], external sources [6, 14], cleaned data samples [8, 21], or continuous user feedback [13, 23]. These general-purpose error correction approaches are designed to repair datasets with user-specified data quality requirements. However, our unsupervised approach is complementary to these systems as it can be incorporated in data cleaning pipelines with *no effort* to accurately fix a large portion of data errors *for free*.

Automatic data cleaning. Data cleaning approaches usually require some sort of human supervision as the notion of high-quality data is user dependent [16]. Human-involvement-free data cleaning is shown to be possible if we limit the scope of the error correction task to a specific data error type, such as missing values [18]. Unsupervised error detection using a huge repository of web tables is also shown to be feasible [10, 22]. Similarly, our approach learns common correction operations from historical revision data to fix a portion of data errors of the new dirty dataset with high precision. However, our approach is not confined to a specific data error type.

5 CONCLUSION

We proposed a new unsupervised error correction approach that automatically fixes data errors. As our experiments show, in the lack of human supervision, our approach outperforms state-of-the-art error correction systems in terms of precision. We plan to improve the recall of our automatic approach by data enrichment.

REFERENCES

[1] 2016. Data science report. https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf.

[2] 2020. Wiki dumps. https://meta.wikimedia.org/wiki/Data_dumps. Accessed: 2020.

[3] Patricia C Arocena, Boris Glavic, Giansalvatore Mecca, Renée J Miller, Paolo Papotti, and Donatello Santoro. 2015. Messing up with bart: Error generation for evaluating data-cleaning algorithms. *PVLDB* 9, 2 (2015), 36–47.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *ISWC*. 722–735.

[5] Gregory V Bard. 2007. Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric. In *ACSW*. 117–124.

[6] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*. 1247–1261.

[7] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. 2013. Nadeef: A commodity data cleaning system. In *SIGMOD*. 541–552.

[8] Xiaoou Ding, Hongzhi Wang, Jiaxuan Su, Zijue Li, Jianzhong Li, and Hong Gao. 2019. Cleanits: A data cleaning system for industrial time series. *PVLDB* 12, 12 (2019), 1786–1789.

[9] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. Holodetect: Few-shot learning for error detection. In *SIGMOD*. 829–846.

[10] Zhipeng Huang and Yeye He. 2018. Auto-detect: Data-driven error detection in tables. In *SIGMOD*. 1377–1392.

[11] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[12] Sanjay Krishnan, Michael J Franklin, Ken Goldberg, and Eugene Wu. 2017. Boostclean: Automated error detection and repair for machine learning. *arXiv preprint arXiv:1711.01299* (2017).

[13] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. 2016. Activeclean: Interactive data cleaning for statistical modeling. *PVLDB* 9, 12 (2016), 948–959.

[14] Mong Li Lee, Tok Wang Ling, and Wai Lup Low. 2000. Intelliclean: A knowledge-based intelligent data cleaner. In *SIGKDD*. 290–294.

[15] Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[16] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *PVLDB* 13, 11 (2020), 1948–1961.

[17] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A configuration-free error detection system. In *SIGMOD*. 865–882.

[18] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. 2010. Eracer: A database approach for statistical inference and data cleaning. In *SIGMOD*. 75–86.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[20] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB* 10, 11 (2017), 1190–1201.

[21] Jiannan Wang, Sanjay Krishnan, Michael J Franklin, Ken Goldberg, Tim Kraska, and Tova Milo. 2014. A sample-and-clean framework for fast and accurate query processing on dirty data. In *SIGMOD*. 469–480.

[22] Pei Wang and Yeye He. 2019. Uni-detect: A unified approach to automated error detection in tables. In *SIGMOD*. 811–828.

[23] Mohamed Yakout, Ahmed K Elmagarmid, Jennifer Neville, and Mourad Ouzzani. 2010. GDR: A system for guided data repair. *SIGMOD*, 1223–1226.