Detecting and Cleaning Errors in Personal Contact Information with Large Language Models

Anna-Christina Glock
Software Competence Center Hagenberg GmbH
Hagenberg, Austria
anna-christina.glock@scch.at

Christine Dominka-Kiss
Austrian Post
Vienna, Austria
christine.dominka-kiss@post.at

Philipp Korom
Austrian Post
Vienna, Austria
philipp.korom@post.at

Lisa Ehrlinger
Hasso Plattner Institute, University of Potsdam
Potsdam, Germany
lisa.ehrlinger@hpi.de

ABSTRACT

Error detection and cleaning of customer and employee data, including names, addresses, and phone numbers, is a critical task in many organizations. Errors in personal contact information, such as misspellings or format inconsistencies, can lead to failed deliveries or delayed tax document distribution. Most enterprise data quality tools offer error detection based on pre-defined rules. These tools often fall short in detecting unexpected and contextual data errors, such as valid but mismatched postal codes and cities.

In this paper, we investigate and benchmark the performance of four large language models (Llama-3, Llama-4, DeepSeek-R1, ChatGPT-4.1), the error detection and data cleaning tools Raha and Baran, as well an Autoencoder to (1) *detect unexpected and contextual errors*, (2) suggest *cleaning steps*, and (3) *explain* error detection in personal contact information. On average, we demonstrate that LLMs outperform Raha and Baran as well as the Autoencoder for error detection and correction. All prompts are provided to repeat and extend our experiments. We further contribute with a synthetic benchmark dataset as well as a data polluter that introduces error types specific to personal contact information. Both components were developed together with domain experts from Austrian Post to replicate key characteristics of real-world data. We conclude that large language models can *detect unexpected and contextual data errors*, which are often overlooked by traditional data quality tools.

VLDB Workshop Reference Format:
Anna-Christina Glock, Christine Dominka-Kiss, Philipp Korom, and Lisa Ehrlinger. Detecting and Cleaning Errors in Personal Contact Information with Large Language Models. VLDB 2025 Workshop: The 2nd International Workshop on Data-driven AI (DATAI 2025).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Anna-Christina-Glock/pci-llm-toolkit.git.

Table 1: Personal contact data with different error types.

Name	Email	City	PostalCode
Jan Beck	jan.beck@ait.at	Salzburg, Altstadt	5020
Kai Eckl	jan.beck@ait.at	Salzburg	–
Ina A[er	ina.auer cas.org	Linz	40b0
Ina Auer	ina.auer@cas.org	Lienz	4020

Type of error: 🟩 Contradictions
🟧 Syntactic formatting error types 🟦 Additional information

1 INTRODUCTION

Most organizations manage personal contact information as part of their master data, for example, data about customers, business partners, applicants, or employees [21]. Personal contact information can change due to relocations or life events and is typically protected by privacy regulations such as the GDPR (General Data Protection Regulation) [10] in Europe or the HIPAA (Health Insurance Portability and Accountability Act) [42] in US healthcare.

Beyond legal requirements to keep personal data up-to-date and correct, organizations themselves have strong incentives to ensure high-quality personal and address data, e.g., to reduce unnecessary mailing costs and CO_2 emissions while strengthening customer satisfaction [30]. Organizations established data governance offices [34] as specialized units to ensure personal data accuracy and security. In a 2025 study, the German Post found that every 8th customer address is incorrect and that the biggest source of errors is outdated customer addresses [30]. In countries with less strict privacy regulations like China [41, 43] or India [39], addresses can contain as many as 200-300 words, possibly including directions to reach a place, mobile numbers, or delivery instructions [39].

Problem statement. Common errors in personal and address data include misspelled words from data entry mistakes (e.g., “Ina A[er”, “Lienz” instead of “Linz”), incomplete records (e.g., “Kai Eckl” missing a PostalCode), or fields containing irrelevant information (e.g., “Salzburg, Altstadt”) [30] as exemplary shown in Table 1. State of the art data quality (DQ) tools (cf. [7, 9, 40]) typically offer lookup tables to validate addresses and also support the detection of expected errors with pre-defined rules. Developing these rules requires significant effort by domain experts – according to Tamm and Nikiforova [40], only ten out of 151 commercial DQ tools can automatically detect and propose DQ rules. Also, rule-based solutions fall short in identifying *unexpected and contextual* errors in personal and address data that appear correct. For example, email addresses can only be validated for syntax (presence of "@"), with semantic correctness requiring contacting the person directly.

Objective. Our research aims to (1) *detect* unexpected and contextual errors in personal and address data, e.g., "Kai Eckl" having the email "jan.beck@ait.at" in Table 1, (2) suggest *cleaning steps* to a user, and (3) *explain* the cause of the identified error, e.g., that an error was detected since the first and last names do not match.

Contribution. To achieve this goal, we investigate the performance of large language models (LLMs) to *automatically detect and clean* personal contact information. In total, we benchmark three local LLMs (Llama-3, Llama-4, DeepSeek-R1), ChatGPT-4.1, the error detection and cleaning tools Raha [25] & Baran [23], as well as Autoencoder for their suitability to perform these tasks. We analyze model performance per data type, pollution level, as well as error types to provide guidelines on where to use LLMs and where not to. We further contribute with a public (synthetic) benchmark dataset and all LLM prompts to repeat and extend our research on error detection and cleaning of personal and address data.

Results. Our experiments show that Llama-3 outperforms the other models in error detection while ChatGPT-4.1 excels in data cleaning. Our results show that

(1) the error type significantly affects performance,
(2) attribute cardinality has a greater influence on error detection and cleaning performance than the data type, and
(3) pollution levels between 0-40% have negligible impact.

Outline. We discuss related work in Section 2 and describe the the evaluation setup, which we developed to conduct our experiments in Section 3. The results of the evaluation are discussed in Section 4 and Section 5 provides an outlook on future work.

2 RELATED WORK

Error detection and cleaning are essential components of any data science pipeline. Therefore, many data quality tools have been developed to detect and clean data errors [7, 9, 29], the majority supporting a user in writing and executing data quality rules. In addition to classic rule-based or similarity-based approaches (as for example implemented in Great Expectations[1] or Deequ [36]), several open-source research tools have been developed that aim at automatically learning data validation and cleaning rules with machine learning (ML), such as, [24, 32, 38]. For example, Raha [25] & Baran [23] are semi-supervised open-source tools to detect and clean data errors with ML methods, such as Gradient Boosting. Another ML-based approach to error detection and cleaning are transformers, which are deep neural networks that learn the context of data and generate the most plausible value for a given context [15].

With the rise of large language models (LLMs), error detection and cleaning can exploit contextual knowledge to detect subtle (unexpected) errors and inconsistencies, which were nearly impossible to detect previously. Zhang et al. [44] demonstrate the high potential of LLMs in identifying previously unseen data errors and suggesting cleaning steps. It is noteworthy to mention that enterprise DQ tools are so far relatively conservative in employing ML or LLMs for error detection and cleaning, as indicated by Gartner's "Magic Quadrant for Augmented Data Quality Solutions" [7].

While fully automated data cleaning is still a too complex task for current LLMs [44], we aim to show their potential specifically for the domain of personal and address data cleaning. We therefore benchmark different LLMs on the one hand to a transformer-based model, and on the other hand to Raha & Baran as representatives for machine-learning based error detection since they clearly outperform all other comparable tools [27, 44].

Error detection and cleaning of personal and address data features specific challenges, such as the parsing and standardizing of addresses that are available as concatenated strings [1, 19], correcting spelling and semantic mistakes [5], matching geodata information to an address [18], or extracting addresses from images of package labels [2, 6]. In our research, we focus on machine-readable personal and address data (e.g., stored in database or CSV file) that has already been split into separate columns but still requires checks for spelling mistakes and contextual consistency.

Transformer-based models [8, 14] and LLMs [43] have already been explored in previous work for personal and address data cleaning. Those works show that these models have an advantage over classic distance-based and rule-based approaches, especially with respect to detecting and correcting contextual data errors. Yang et al. [43] show that LLMS are better than transformer-based models like BART [20], which aligns very well with our findings in this paper. In contrast to Yang et al. [43], who use a fine-tuned RAG (Retrieval Augmented Generation) model for address cleaning, we evaluate the suitability of out-of-the-box (not fine-tuned) LLMs with a zero-shot approach since we do not assume training data to be available. However, to the best of our knowledge, none of these works investigated the performance of LLMs and transformer-based models with respect to different data types as well as error types. We deem this as an important finding especially for application in practice, where it is important to decide where to use language-based models and where rule-based systems.

3 EXPERIMENTAL SETUP

Figure 1 provides an overview of the experimental setup used to conduct our study. We discuss the methods used in Section 3.1, details of the implementation in Section 3.2, both the company and the synthetic datasets in Section 3.3, and the error pollution to introduce errors with ground truth in Section 3.4.

The process of error detection and correction within our experiments can be shown with an example from Table 1. Assume that City contains an incorrect character, which turns the correct city name "Linz" to a still valid Austrian city name "Lienz", which is however incorrect in the context of the record considering postal code and street name (omitted for brevity). During the experiments, all attributes of a record are provided as input data to the different methods. If a method detects the respective error (i.e., that City is incorrect), it also provides a cleaning suggestion (e.g., to replace "Lienz" with "Linz" based on knowledge about the PostalCode).

3.1 Error Detection and Cleaning Methods

Overall, we evaluate the ability of three different methods to detect and clean errors in personal contact information: (1) four LLMs,

[1]https://greatexpectations.io

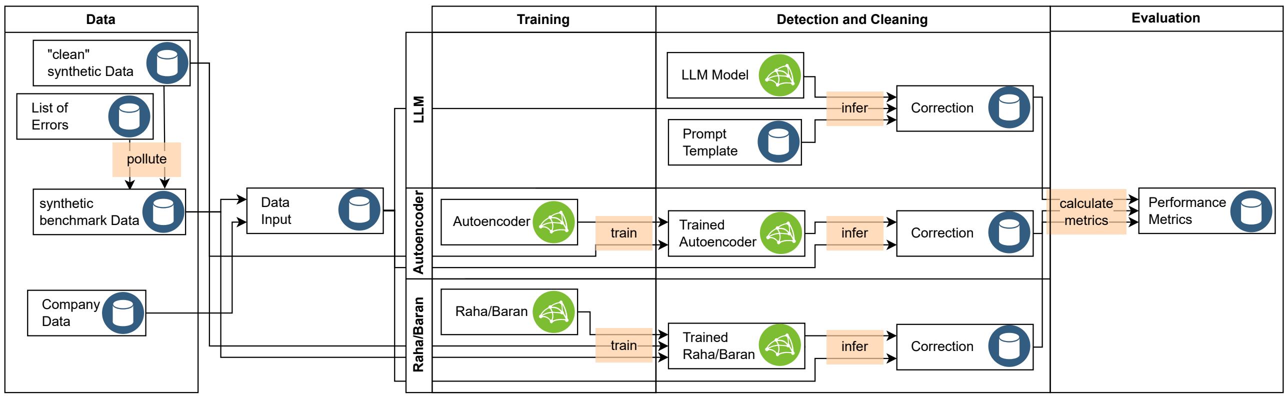

Figure 1: Experimental setup of the evaluation of the data detection and cleaning methods.

(2) Raha & Baran as representatives of state-of-the-art data cleaning tools, and an (3) Autoencoder. Given a set of input data (see Section 3.3) each method computes the results, which are subsequently compared to the ground truth. We use the F1 score to indicate the performance of each method.

3.1.1 LLMs. We use pre-trained models for all LLMs and provided the data via prompting to perform the error detection and correction task. The prompt was created and refined beforehand, then applied consistently to each record of the dataset and can be reused across different types of personal contact data. We refined the prompt in four steps – see GitHub[2] to access all prompts – and used the last one (p4) for our experiments. The LLM response is subsequently parsed to extract the error detection and correction results.

3.1.2 Raha and Baran. Raha and Baran require a clean and a dirty dataset for training. For evaluating the benchmark dataset (see Section 3.3) we simply used a clean dataset (before pollution) as well as a dirty dataset (after pollution) for training. We used the same benchmark dataset (5,000 records) for training Raha and Baran to perform the experiments on the company dataset, since we aimed to model a real-world scenario without the availability of clean data.

3.1.3 Autoencoder. The architecture of the Autoencoder model is a bidirectional LSTM (long short-term memory) layer, an LSTM layer with attention, and a dense output layer. To train the Autoencoder model, we generated a clean dataset (50,000 records) that differs from the evaluation dataset. Some samples may be found in both.

3.2 Implementation

The experimental evaluation was conducted in Python. The Autoencoder was built and trained on a local server (CPU: Intel Xeon E5-2698 v4, GPU: 4x NVIDIA Tesla V100 DGXS 64GB, RAM: 256GB) using the KERAS[3] framework. For Raha [25] and Baran [23], we use the latest commit [22] of the original Python implementations as proposed in the respective publications. Both were run on the same server as the Autoencoder. The experiments with local LLMs are performed using the Python OpenAI[4] package for API calls to each of the following three LLMs:

- **Llama-3**: casperhansen/llama-3.3-70b-instruct-awq,
- **Llama-4**: RedHatAI/Llama-4-Scout-17B-16E-Instruct-FP8-dynamic, and
- **DeepSeek-R1**: Valdemardi/DeepSeek-R1-Distill-Qwen-32B-AWQ.

Those models are chosen based on their performance on the Chatbot Arena LLM Leaderboard[5] and their compatibility with our computing infrastructure. Our local high-performance computing platform has a single AMD EPYC 7643 with 48-cores, 2.3 GHz CPU and four NVIDIA A100 GPUs, each with 160GB of high-bandwidth memory. The system is equipped with 1TB of RAM. ChatGPT-4.1 is evaluated with the batch processing feature from OpenAI.

3.3 Personal and Address Datasets

For our experiments, we use a proprietary company dataset provided by the Austrian Post (postal service) about Austrian personal and address data, which is a combination of synthetic and real data. Since this dataset cannot be made publicly available, we developed a synthetic benchmark dataset that replicates its key characteristics and which we publish to enable reproducible research[6].

Company dataset. While the addresses in the company dataset are real-world data, the personal data have been synthetically generated and polluted to ensure compliance with privacy regulations. The company dataset contains 14 attributes, consisting of:

- 7 personal data attributes: FirstName, LastName, Email, Country, DialingCode, PhoneNumber, and DateOfBirth
- 7 address data attributes: Street, HouseNumber, Stairway, Door, PostalCode, City, and CountryCode

The following 7 attributes contained errors: Email, Street, City, Country, DateOfBirth, HouseNumber, and PostalCode. Incorrect values were marked with one out of six different labels, referring to a previously defined rule that was violated. As these rules were applied to one or multiple attributes, multiple rules can match a

[2]https://github.com/Anna-Christina-Glock/pci-llm-toolkit/blob/main/error-detection-cleaning-llm/data/experiments_5000/prompt.json
[3]https://keras.io

[4]https://github.com/openai/openai-python
[5]https://lmarena.ai/?leaderboard
[6]https://github.com/Anna-Christina-Glock/pci-llm-toolkit.git

single error type. Five labels can be mapped to the error types (ER1a–c) from Table 2 and the sixth label can be mapped to (ER4).

Benchmark dataset. The benchmark data has the same schema as the company data and replicates its key characteristics. Since the benchmark dataset does not contain any personally identifiable information, we publish it to validate and further extend our experiments on personal contact data error detection and cleaning. The benchmark data also allows us to systematically introduce commonly observed real-world data errors (both the type and frequency) with known ground truth. This controlled environment allowed for a sound evaluation of different models' performance in error detection and data cleaning tasks, facilitating the identification of strengths and weaknesses of each approach.

For the generation of the benchmark dataset, we used real Austrian addresses, combined with fictional person data to make it more realistic. Hence, country code was always set to Austria and the DialingCode was selected from a list of Austrian DialingCodes. An Austrian telephone number consists of 3 to 7 numbers, which were selected randomly. The FirstName and LastNames were randomly selected from official lists to simulate real persons. For the DateOfBirth, a date within the allowed bounds was generated randomly. The real Austrian addresses were randomly picked from the official Austrian address register [11].

3.4 Data Pollution

We polluted the person and address benchmark data with different error types to enable systematic evaluation of the error detection and cleaning methods. Although there are a number of open-source data pollution tools like BART [4], GouDa [33], or Jenga [35], we decided to develop our own pollution strategy together with domain experts from Austrian Post to support common data error types (ER) specific to person and address data.

In alignment with related work on data error classification [13, 16, 17, 28, 31], we distinguish on a high-level between two *syntactic* (ER1–2) and four *semantic* error types (ER3–6). For the syntactic errors types, we investigated the impact of inserted/updated/deleted (ER1a) letters, (ER1b) numbers, or (ER1c) special characters that violate a specific data format, as well as (ER2) words that start with lowercase although uppercase is required. Such formatting errors (discussed as "heterogeneous formatting" in [16, 28]) encompass violations of the structure and composition of the data, such as the presence of incorrect characters [16]. These errors can be detected independently of a specific application or data semantics by just investigating the format.

For semantic data errors, we consider the following four types:

- **Additional information (ER3)**: Additional, irrelevant information is present in a field (also mentioned in [13] and as "embedded values" in [31]), such as City="Salzburg, Altstadt" instead of City="Salzburg" as outlined in Table 1.
- **Contradictions (ER4)**: A value that is individually valid, but contradicts with other fields of the same record, e.g., PostalCode=1220 where City="Salzburg" for the same record (note that 1220 is a postal code for Vienna). This error type is also discussed in [26] as well as in [16] under the umbrella term "incorrect value".
- **Valid-value typo (ER5)**: Typographical errors (also: typos) are misspellings caused by keystroke mistakes during data entry [37]. We use the term valid-value typo to describe typos that result in a valid but incorrect value. In other words, the value is inside the domain (generally valid), but does not represent the intended value of its real-world representation. An example is City="Lienz" instead of City="Linz".
- **Value misplacement (ER6)**: The value from one attribute is included in another attribute, also discussed as "embedded values" in [31], which are mainly caused by problems during the data entry process, i.e., the data were entered in the wrong attribute. An example is FirstName="Jan Beck" which includes already the last name of the person.

For our experiments, we generated batches of 5,000 records for different pollution levels (2.5%, 10%, 15%, 20%, 25%, 30%, 40%) and polluted each batch with all 8 error types evenly distributed.

4 RESULTS AND DISCUSSION

In this section, we experimentally evaluate the ability of the models described in Section 3.1 to detect and clean data errors: per data type in Section 4.1, per error type in Section 4.2, per pollution level in Section 4.3, and the runtime performance in Section 4.4. We use the F1 score as a metric to compare the results of the different models, having a range between 1 to 0, where 1 is the best result and 0 means that no error has been found. Since the F1 scores only show minimal variation by pollution level (SD: 0.12), the results are aggregates across all levels. The F1 score is the harmonic mean of recall and precision, where both values are comparable for Autoencoder, Raha, and Baran. For the LLMs, recall is generally higher than precision, which means that LLMs are good in detecting and correcting errors, but also introduce many false positives. For example, for the FirstName column, the average precision of all LLMs is 0.5 and the recall is 0.8 (leading to a F1 score of 0.57), which means that 80% of the errors in FirstName were detected and 50% of all detected errors were also labeled errors.

4.1 Evaluation per Data Type

The first experiment investigates the impact of the datatype on detection and cleaning performance as shown in Table 3. In our data, we have three different data types: `string` (7 attributes), `integer` (6 attributes), and `date` (1 attribute). Results for error detection are discussed in Section 4.1.1 and for data cleaning in Section 4.1.2.

4.1.1 Error detection. Table 3 shows that the LLMs constantly outperform the other methods (Raha and Autoencoder) in error detection, but different LLMs perform best for different attributes.

Benchmark data. For the benchmark data, DeepSeek-R1 demonstrates the lowest error detection performance among all LLMs, except for the DateOfBirth attribute, where it achieves the highest F1 score. Llama-4 shows results comparable to Llama-3 and ChatGPT-4.1, but generally underperforms the other two models except for Street and HouseNumber. Llama-3 outperforms ChatGPT-4.1 across most attributes, except for Email and PhoneNumber.

We further investigated the attributes with the lowest F1 scores, indicating the most difficult to detect error types: Email, PhoneNumber, and DialingCode. For Email, there were several false positives,

Table 2: Mapping of error types to the corresponding attributes in which they were polluted for the benchmark datasets.

		Address data							Personal data						
		Street	HouseNumber	Stairway	Door	PostalCode	City	Country	FirstName	LastName	Email	CountryCode	DialingCode	PhoneNumber	DateOfBirth
Syntactic formatting error types															
(ER1a)	Letters	×				×	×	×				×	×	×	×
(ER1b)	Numbers	×				×	×	×	×	×					×
(ER1c)	Special characters	×	×	×	×	×	×	×	×	×	×	×	×	×	×
(ER2)	Start with lowercase	×					×	×	×	×					
Semantic error types															
(ER3)	Additional information					×	×								
(ER4)	Contradictions	×				×	×	×							×
(ER5)	Valid-value typo	×				×	×	×							
(ER6)	Value misplacement	×	×	×	×	×	×	×	×	×	×	×	×	×	×

Table 3: Error detection and cleaning performance (F1 score) per model and dataset. Best F1 score per attribute is marked bold.

	Benchmark dataset														Company dataset						
	string							date	integer						string					date	integer
	LastName	FirstName	Email	Street	HouseNumber	City	Country	DateOfBirth	CountryCode	DialingCode	PhoneNumber	Stairway	Door	PostalCode	Email	Street	HouseNumber	City	Country	DateOfBirth	PostalCode
Error detection																					
Autoencoder	0.00	0.00	0.01	0.03	0.02	0.05	0.03	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.36	0.05	0.43	0.00	0.06	0.00
Raha	0.03	0.04	0.03	0.04	0.04	0.06	0.04	0.05	0.02	0.03	0.02	0.02	0.02	0.04	0.00	0.52	0.05	0.57	0.52	0.06	0.46
DeepSeek-R1	0.39	0.37	0.07	0.27	0.35	0.17	0.69	**0.33**	0.45	0.06	0.05	0.32	0.37	0.13	0.12	0.92	0.22	0.85	0.98	0.17	**0.83**
Llama-4	0.73	0.51	0.03	**0.38**	**0.59**	0.18	0.83	0.11	0.39	0.03	0.03	0.72	0.69	0.13	0.08	**0.97**	0.34	**0.88**	0.92	**0.63**	0.79
Llama-3	**0.81**	**0.78**	0.06	0.31	0.53	**0.22**	**0.98**	0.30	**0.95**	**0.08**	0.03	**0.81**	**0.75**	**0.21**	**0.18**	0.95	**0.44**	0.86	**0.99**	0.30	0.75
ChatGPT-4.1	0.72	0.59	**0.08**	0.19	0.53	0.12	0.94	0.29	0.30	**0.08**	**0.08**	0.55	0.53	0.16							
Error cleaning																					
Autoencoder	0.00	0.00	0.01	0.03	0.02	0.05	0.03	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Baran	0.02	0.04	0.00	0.03	0.03	0.04	0.00	0.02	0.00	**0.02**	**0.02**	0.00	0.01	0.02	0.00	0.52	**0.05**	0.57	**0.52**	**0.06**	0.46
DeepSeek-R1	0.47	0.32	0.03	0.09	0.14	0.08	0.05	0.11	0.19	0.00	0.01	0.92	0.82	0.03	**0.01**	0.67	0.00	0.69	0.37	0.00	0.60
Llama-4	0.49	0.34	0.01	**0.14**	0.25	0.14	0.02	0.02	0.15	0.00	0.01	0.88	0.83	0.07	0.00	0.80	0.01	0.78	0.26	0.00	**0.73**
Llama-3	0.51	**0.68**	0.02	0.12	0.39	**0.17**	0.08	0.11	0.39	0.00	0.01	**0.93**	**0.90**	0.11	0.00	**0.83**	0.00	**0.80**	0.36	0.00	0.69
ChatGPT-4.1	**0.67**	0.56	**0.06**	**0.14**	**0.46**	0.12	**0.14**	**0.16**	**0.41**	0.00	**0.02**	0.89	0.74	**0.27**							

where an LLM cautiously detected an error, which was not polluted. Examples are email addresses that did not have a typical Austrian domain or mismatches between the name in the Email and the corresponding FirstName and LastName attribute values (e.g., "Kai Eckl" having an email "jan.beck@ait.at"). The high number of false positives for LLMs can be explained by (i) the wide range of valid values, e.g., for email addresses, (ii) limited (domain) knowledge about a specific data type, and can also be attributed to (iii) artifacts inherent to the synthetic benchmark dataset. The LLMs marked several unlikely values as data errors (e.g., an Email whose domain was associated with the French government but was linked to an Austrian address). This artifact was not intentionally introduced during the pollution process, but very unlikely after closer investigation. Although lowering the F1 score, this behavior supports our claim to detect *unexpected and contextual data errors*.

Phone numbers are often classified as incorrect by LLMs due to perceived length issues, e.g., "77712" is incorrectly flagged as erroneous since it is too short. The LLM explanation states that "correct Austrian telephone numbers should have 7-8 digits." However, according to Austrian law there are old telephone numbers that may be shorter than 7 digits, indicating limited knowledge of valid Austrian phone number formats in the evaluated LLMs.

For the DialingCode, LLMs frequently detect inconsistencies between the PhoneNumber and DialingCode as well as between

City and DialingCode, which are causes for false positive error detections. The detected inconsistencies between City and DialingCode are specifically interesting because mobile numbers are not geographically bound and should not depend on the DialingCode.

The LLMs also tend to flag the two attributes City and PostalCode (which are conceptually linked) jointly as incorrect even if only one attribute contains an error. The same holds for the conceptually linked attributes DialingCode, PhoneNumber, and CountryCode. This leads to false positives for the correct attributes. Since CountryCode is an integer, it should not contain non-numeric characters. However, all four LLMs mistakenly consider a missing leading "+" in CountryCode as an error. This limitation stems from the lack of access to attribute metadata for the LLMs, preventing them from recognizing that the "+" is not allowed in integers.

The error detection performance of the LLMs for City and PostalCode is also affected by the limited knowledge of smaller Austrian towns, particularly when multiple cities share one postal code. Street name validation also suffers from the fact that small towns often use the the town name as the street name.

Company data. Table 3 shows that the local LLMs also constantly outperform Raha and the Autoencoder on the company dataset. All models achieved higher F1 scores than for the benchmark dataset, particularly Raha and Autoencoder. An exception is the HouseNumber, likely due to mainly having missing values (instead of syntactic formatting issues in the benchmark data). According to our analysis, reasons for detecting false positive errors by LLMs are comparable to the reasons discussed for the benchmark data.

Summary. We can conclude that there is no single data type that significantly impacts model performance compared to others. The main difference in error detection performance between attributes stems from the cardinality (range of valid values). Attributes like Email or PhoneNumber have a wide range of valid values, leading to many false positives (F1: 0.08 for both attributes). For attributes like Country (F1: 0.98) or CountryCode (F1: 0.95) the cardinality is very restricted in our use case, leading to high accuracy. Also interestingly, the local LLMs outperform ChatGPT-4.1 in all but three attributes. Autoencoder and Raha perform worse than the LLMs, but exhibit similar performance per data types.

4.1.2 Data cleaning. Table 3 shows that the correction (cleaning) of errors is more challenging, with the best F1 scores for each attribute typically being lower than the best F1 score for error detection.

Benchmark data. For the benchmark data, ChatGPT-4.1 generally shows better performance than the local LLMs. Notably, Baran delivered the best results for DialingCode, outperforming all LLMs.

For the LLMs, we observed that error correction for integer attributes is generally easier than for string or date attributes. Interestingly, Baran and Autoencoder perform similarly across all data types. Furthermore, attributes with limited contextual information (e.g., DateOfBirth) were found to perform worse than those with more informative context (e.g., PostalCode). While it is unlikely that an LLM can find information about a person's correct date of birth in its training data, it is expected to have learned the appropriate postal codes for specific City and Street combinations.

Company data. Table 3 shows similar discrepancies between the detection and correction results for the company dataset than with the benchmark data. Again the LLMs perform better than the other methods. However, Baran shows a better performance than the LLMs for DateOfBirth, CountryCode, and HouseNumber, likely because it generates correction values for all detected errors, which increases the chance to correct DateOfBirth and HouseNumber successfully. The LLMs do not automatically suggest a corrected value, but may only inform the user about the correct format and that further investigation is required due to missing knowledge.

In the case CountryCode, the LLM's corrections ("AT" or "Austria") are often incorrect, as only "AUT' is valid in our use case. Here, Baran has learned the correct code from the data. The high correction accuracy suggests that this information would also be useful to be handed over as input to an LLM (e.g., via the prompt) to further improve performance.

Summary. The results of the data cleaning experiments again show that the data type has less impact on model performance than the attribute cardinality. Overall, the LLMs outperform the Autoencoder and Baran in data cleaning. Some attributes (e.g., DialingCode, Street) show significantly better performance on the company dataset. Here we assume that similar error types that occur in these attributes have a higher impact, as discussed in the following subsections.

4.2 Evaluation per Error Type

In this subsection, we discuss the results for the 8 different error types, first for the detection and then for the cleaning suggestions. Figure 2 shows the results in four subplots (1a, 1b, 2a, 2b) with the F1 score being at the y-axis and the attribute names on the x-axis. Each subplot is further split into one plot per error type. As Raha is only used for error detection and Baran is only for data cleaning, we use the same color and symbol for both tools throughout the plots. This means the results marked by a pink star in plots 1a and 2a are produced by Raha and in 1b and 2b by Baran.

4.2.1 Error detection. For the error detection, the LLMs outperform the Autoencoder and Baran across nearly all attributes and error types, as visualized in Figure 2(1a). Note that not all error types are applicable to every attribute; as summarized in Table 2. The evaluation is based on the respective subset of the dataset that contains instances of the error type under investigation.

The results show that LLMs perform better in detecting (ER1) formatting errors, (ER3) additional information, and (ER6) value misplacement when compared to the other error types. In contrast, Autoencoder and Baran demonstrate a better performance in identifying (ER4) contradictions and (ER5) valid-value typos.

Syntactic error types. Similar to the result per data type, LLMs performed worse for Email, PhoneNumber, and DialingCode in comparison to other attributes. For the Email this is because email addresses are very versatile and a rule is often only that a "@" and a domain have to exist at the end for a correct email. CountryCode, DialingCode, and PhoneNumber are often not only marked as incorrect if they are actually incorrect (true positive), but also due to other

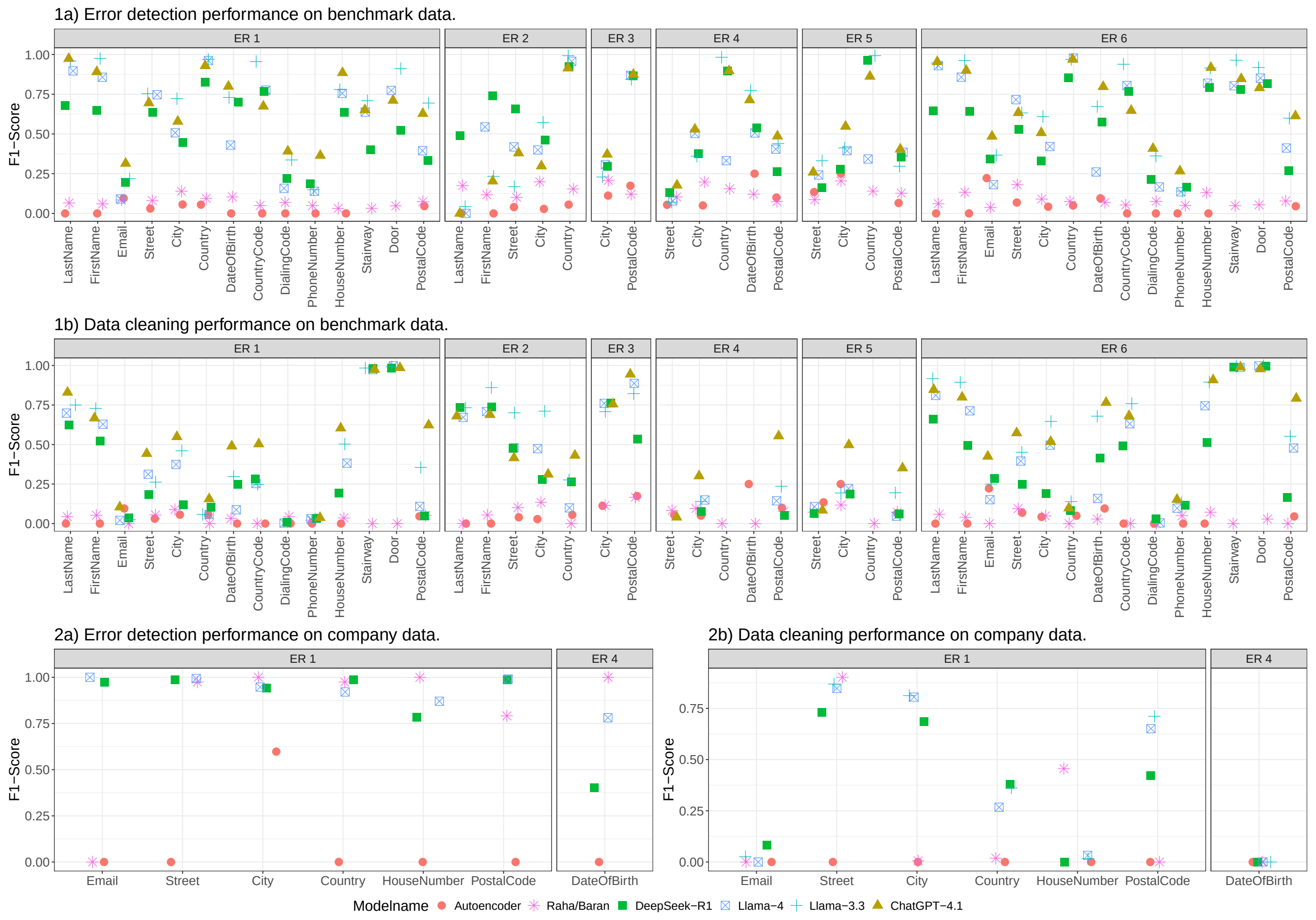

Figure 2: Error detection and cleaning performance (F1 score) per data type. Plots 1a and 1b stem from the benchmark dataset and plots 2a and 2b from the company dataset.

attributes being incorrect. For example, for the incorrect DialingCode “68=6”, a correct CountryCode “43” and a correct PhoneNumber “750530”, all three attributes are marked as incorrect because the combination of CountryCode-DialingCode-PhoneNumber would lead to an incorrect phone number. This results in one true positive and two false positives. Sometimes, the LLMs were also unsure if those attributes were correct due to their inability to check them and still marked them as incorrect.

Note that (ER2) start with lowercase does not apply to all string-based attributes, such as, Email. We did not specifically prompt the LLMs to detect and correct inconsistent cases. Interestingly, Raha performs poorly in detecting (ER2), which might indicate that (ER2) was not sufficiently present in the dataset used to train Raha. In addition, there are notable differences in the results for FirstName and LastName. The LLM-provided explanations suggest that it is more likely to identify FirstName as requiring the first letter uppercase than LastName.

Semantic error types. For the semantic error types, (ER3) additional information is particularly challenging to detect because it requires the classification of valid information as incorrect due to too much information. While the different performance for City and PostalCode might seem surprising, it turned out that the identification of additional characters (especially letters and special characters) is much easier in numeric attributes than in textual attributes due to the restricted domain.

For the detection of (ER4) contradictions, contextual knowledge (from other attributes) is required, which is why the LLMs also clearly outperformed Raha and Autoencoder. Interestingly, (ER4) error detection in the Street attribute is worse than for other error types. From the explanations, we can conclude that the LLMs often do not validate the existence of a street within a city, despite the clear instruction through the prompt to consider all attributes jointly. In contrast, information in City and PostalCode are typically considered in combination. Interestingly, Llama-4 has problems detecting errors in CountryCode compared to the other LLMs.

(ER5) valid-value typo yields similar results to (ER4), but with a better performance on error detection in the Street attribute. Here, we speculate that this higher error detection rate is due to the closeness of (ER4) to regular typos. In contrast to contradictions where any valid value can appear, for (ER5) only valid and similar values are used, e.g., “Seeackerweg” instead of “Seeauerweg”. Error detection performance on PostalCode was worse, possibly due to the LLM’s inconsistent consideration of the Street when checking for errors. This is particularly relevant in larger cities, where multiple PostalCode codes correspond to the same City. Here, the Street information, combined with the City, can help identifying the correct PostalCode in these cases.

For (ER6) value misplacement, the results are similar to those of the syntactic formatting error type. Here LLMs have the advantage of using the context, which makes it easier to detect if further values are included that do not fit the actual value.

Company data. As outlined in Section 3.4, the company dataset contains two error types (ER1) syntactic error types and (ER4) contradictions. The results for the company dataset (see Figure 2 2a) show that the LLM constantly outperforms Raha and the Autoencoder. Note that Raha tends to mark nearly every record as incorrect for some attributes, which leads to very good results when only considering the attributes that are labeled as incorrect. The observations from the benchmark dataset on (ER1) and (ER4) can be transferred to the company partner dataset.

4.2.2 *Data cleaning.* The cleaning capabilities of the methods reveal that some error types, like value misplacement are easier to correct than others, such as contradictions. In alignment with the detection results, the LLMs also outperform the other methods in error cleaning. However, even though LLMs are effective in detecting CountryCode errors, they are not as effective in correcting them, as they often correct to “Austria” or “AT” instead of the correct code for this data set “AUT”.

Syntactic error types. Correcting (ER1) formatting errors like inserted/updated/deleted letters, numbers, or special characters is challenging because replacing a character makes it hard to infer the original value. For some attributes like City or FirstName, LLMs can easily guess the correct value using the knowledge gained during training. However, attributes like PhoneNumber or Email are more difficult to correct because an LLM is unlikely to have trained knowledge about these specific values. For (ER2) start with lowercase, error correction seems very straightforward due to the high performance across the models.

Semantic error types. Once (ER3) additional information is detected, the correction is very easy, as it simply requires removing the irrelevant information. (ER4) contradictions and (ER5) valid-value typos are the hardest error types to correct. The challenge lies in determining which attribute contains the correct value and which one is incorrect since obviously both values are generally valid (inside the domain). For example, if the combination of PostalCode and a City is incorrect, but each value itself is considered correct, it is unclear which value should be corrected. In some cases, further attributes provide additional insights such as the Street, which however does not always hold for very common street names. Correcting (ER6) value misplacements is straightforward with respect to their detection (cf. Figure 2).

Company data. The results from the company dataset in Figure 2 show that for attributes where illegal characters are added, the cleaning works well. For attributes where the entire value is missing (e.g., HouseNumber in ER1) or characters are removed (e.g., Email in ER1), the correct value can only be assumed since the information provided to the methods is not enough to robustly provide the correct value. This is also true for the DateOfBirth (ER4). Baran performs surprisingly well for cleaning HouseNumber, probably because it uses the clean data set to identify common patterns and provide cleaned values for incorrect ones.

4.3 Evaluation per Pollution Level

Table 4 shows that the standard deviation (SD) in the F1 score (averaged over all attributes) per model is rather small for both error detection and cleaning (< 0.1). We can see that Autoencoder, Raha, and Baran have more stable results at different levels of pollution than the LLMs. The lower SD values for data cleaning across all pollution levels indicate more stable results for this task compared to error detection.

Table 4: Standard deviation of the F1 score across all pollution levels and models for error detection and cleaning.

Model	SD Detect	SD Clean
Autoencoder	0.006	0.006
Raha& Baran	0.012	0.005
DeepSeek-R1	0.098	0.058
Llama-4	0.07	0.067
Llama-3	0.071	0.068
ChatGPT-4.1	0.092	0.079

4.4 Runtime Performance

In this subsection, we compare the execution runtime between the models. Runtime performance was no priority for our application use case, but an aspect to consider for future applications.

While the Autoencoder, Raha, and Baran were run on the same local server (see Section 3.2 for details), LLMs required a more powerful computing environment. ChatGPT-4.1 was executed via the batch processing API[7], which reduces the cost per token but introduces variability in execution time, making direct comparison challenging. Depending on the workload, it can take up to 24 hours to receive the results. In our experience with a batch size of 5,000 prompts, one batch took about 22 to 48 minutes.

The Autoencoder had a mean execution time of 26 seconds per record on our hardware, leading to an execution time of 36.1 hours for one dataset. Running 50 records in parallel reduced the execution time to 2.32 hours for the whole dataset and 1,68 seconds per record.

For Raha and Baran, we calculated their execution times from the overall result, because the implementation utilizes parallel processing which we could not turn off easily. Raha needed an average of 2.5 minutes per dataset, which translates to 0.03 seconds per record. Baran requires 1.1 hours per dataset and 0.79 seconds per record, making it significantly slower than Raha. The combined time for cleaning and correcting a record is 0.82 seconds.

For the local LLMs, one prompt takes an average of 14 seconds, thus 19.44 hours for the entire dataset. Like with Autoencoder, we

[7]https://platform.openai.com/docs/guides/batch

Table 5: Execution time per model, the fastest time per record and dataset is marked in bold.

	Serial processing		Parallel processing	
Model	Record	Dataset	Record	Dataset
Autoencoder	26 s	36.111 h	1.675 s	2.326 h
Raha	NA	NA	**0.030 s**	**2.516 m**
Baran	NA	NA	0.794 s	1.103 h
Local LLMs[†]	13.866 s	19,258 h	1,585 s	2,201 h
ChatGPT-4.1[*]	NA	NA	0,007 s	36.111 m

[s] seconds; [m] minutes; [h] hours; [NA] no measurement

[*] can take up to 24 hours; [†] run on high performance cluster

ran 50 prompt requests in parallel to increase the processing speed for the entire dataset. The time was reduced to 2.2 hours for the whole dataset and 1.59 seconds per record.

Our experiments show that Raha and Baran are the fastest models. While the Autoencoder has similar runtime performance as the local LLMs, it is run on a local machine and therefore considered to be faster. The experiments also highlight the importance of parallelization, especially when detecting errors in larger datasets.

4.5 Summary of Findings

We found that overall ChatGPT-4.1 and Llama-3 performed best for the detection and cleaning of personal contact information. The good performance of Llama-3 is particularly relevant, since personally identifiable information often requires processing by local LLMs due to privacy concerns.

Our experiments further show that the *error type* as well as the *cardinality* (range of valid values within a data attribute) have the greatest impact on error detection and cleaning performance across all models. For the *error type*, syntactic formatting errors are much easier to detect and clean than semantic error types, as the latter tend to require consideration of multiple attributes. In contrast, the *data type* (string, integer, or date) as well as the *pollution level* (0-40%) have a minor impact on the results, as indicated in Table 3.

Risks of using LLMs. Apart from limited domain knowledge for very specific domains (e.g., LLMs have troubles in finding the correct PostalCode to a certain City for small City), a core limitation of using LLMs is the fact that they do not always return the results in the requested format, making parsing difficult. Due to these limitations and the tendency of LLMs to hallucinate, we claim (in alignment with Zhang et al. [44]) that a fully automated error detection and cleaning pipeline is not feasible to the current state, especially for critical domains. However, we believe that using LLMs within such pipelines is extremely valuable, especially paired with a human in the loop to verify the results, as detailed below.

Potential of using LLMs. The LLMs can detect and correct errors without explicit fine-tuning or prompting for specific tasks, for example, they detected the (ER2) start with lowercase error type without explicit prompting. This behavior opens up the possibility that LLMs can identify previously unknown and unexpected error types. A concrete example that appeared during our experiments were unexpectedly exchanged values. For example, during our experiments, we found that tuple {Street: “Bregenz”, City: “6900”, PostalCode: “Schloßgasse”} in the company data had exchanged values for Street, City and PostalCode. The LLM detected all three columns as incorrect, provided the reason that the values are in the wrong attributes, and suggested the correct exchange to: {Street: “Schloßgasse”, City: “Bregenz”, PostalCode: “6900”}.

The capability of LLMs to provide explanations for detected error types allows domain experts to understand and learn from the LLMs, for example, to turn the detection results into rules.

In alignment with Ankireddi [3], we believe that the future will lead to a next-generation data stewardship, merging LLMs and human expertise. In such a framework, AI-based automation is not restricted to rule-based data quality checks, but LLMs that actively scan datasets to identify inconsistencies and suggest error detection and correction rules. These error detection and correction steps are overseen by data stewards who are free to reject what was suggested by an LLM (human in the loop approach). This specific combination of LLMs and human intelligence is likely to improve not only efficiency but also accuracy in data governance initiatives.

5 CONCLUSION

In this paper, we analyzed four LLMs (Llama-3, Llama-4, DeepSeek-R1, ChatGPT-4.1) and compared it to an Autoencoder and the error detection and cleaning tools Raha and Baran. The aim of our research was to find a model to (1) detect *unexpected and contextual errors* in personal and address data, (2) *suggest cleaning* steps, and to (3) *explain* the cause of the error.

We found that ChatGPT-4.1 and Llama-3 performed best for these tasks, which is especially relevant for personal contact information due to privacy concerns. Regarding the influence of different factors on performance, we found that error types and attribute cardinality have a significant impact. In contrast, the impact of pollution level and data type is minimal. A summary of the detailed findings with respect to error detection and cleaning by data type, pollution level, and error type is provided in Section 4.5.

For future work, we will explore the tighter integration of LLMs to error detection and cleaning for personal contact data in organizations from two sides. On the one hand, we would like to transfer knowledge about formally unexpected and unknown errors from the LLM to the domain experts, for example, by using explanations to automatically generate validation rules. Here, we plan to investigate the quality of the explanations provided and how to reuse them. On the other hand, we want to integrate domain experts' knowledge about data quality rules to LLMs. This integration could be done by enriching the prompts, or alternatively, by using a RAG (Retrieval Augmented Generation) [12]. RAGs offers a more flexible alternative to fine-tuning the LLM by providing supplementary information, for example about small towns in Austria.

ACKNOWLEDGMENTS

This research has been funded by BMIMI, BMWET, and the State of Upper Austria in the frame of the SCCH competence center INTEGRATE (FFG grant no. 892418) part of the FFG COMET Competence Centers for Excellent Technologies Programme and by the “ICT of the Future” project QuanTD (no. 898626).

REFERENCES

[1] Nosheen Abid, Adnan ul Hasan, and Faisal Shafait. 2018. DeepParse: A Trainable Postal Address Parser. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, Canberra, Australia, 1–8.

[2] Moa Andersson, Summrina Kanwal, and Faiza Khan. 2022. Cognitive-Inspired Post-Processing of Optical Character Recognition for Swedish Addresses. In *2022 IEEE 21st International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*. IEEE, Toronto, ON, Canada, 248–257. https://doi.org/10.1109/ICCICC57084.2022.10101672

[3] Viswakanth Ankireddi. 2025. Next-Generation Data Stewardship: Merging AI and Human Expertise. *International Journal of Information Technology and Management Information Systems* 16, 2 (March 2025), 1243–1251.

[4] 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. *Proceedings of the VLDB Endowment* 9, 2 (2015), 36–47. https://doi.org/10.14778/2850578.2850579

[5] Divya Bhadauria, Alejandro Sierra Múnera, and Ralf Krestel. 2024. The Effects of Data Quality on Named Entity Recognition. In *Proceedings of the Ninth Workshop on Noisy and User-generated Text (W-NUT 2024)*. Association for Computational Linguistics, San Ġiljan, Malta, 79–88. https://aclanthology.org/2024.wnut-1.8/

[6] Moncef Charfi, Monji Kherallah, Abdelkarim El Baati, and Adel M Alimi. 2012. A New Approach for Arabic Handwritten Postal Addresses Recognition. *International Journal of Advanced Computer Science and Applications* 3 (2012), 1–7.

[7] Melody Chien, Divya Radhakrishnan, and Sue Waite. 2025. *Gartner Magic Quadrant for Augmented Data Quality Solutions.* Technical Report. Gartner, Inc.

[8] André V. Duarte and Arlindo L. Oliveira. 2023. Improving Address Matching Using Siamese Transformer Networks. In *Progress in Artificial Intelligence*. Springer Nature Switzerland, Cham, 413–425.

[9] Lisa Ehrlinger and Wolfram Wöß. 2022. A Survey of Data Quality Measurement and Monitoring Tools. *Frontiers in Big Data* 5 (2022), 850611.

[10] European Union. 2016. *General Data Protection Regulation.* https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:02016R0679-20160504 (Accessed: 2025-07-01).

[11] Federal Office of Metrology and Surveying. 2024. Austrian Adressregister. https://www.bev.gv.at/Services/Produkte/Adressregister/Oesterreichisches-Adressregister.html (Downloaded: 2024-04-01).

[12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] https://arxiv.org/abs/2312.10997

[13] Mouzhi Ge and Markus Helfert. 2007. A Review of Information Quality Research – Develop a Research Agenda. In *Proceedings of the International Conference on Information Quality (ICIQ)*. MIT, Cambridge, MA, USA, 76–91.

[14] Zahra Rezaei Ghahroodi, Hassan Ranji, and Alireza Rezaee. 2024. Address Matching Using Machine Learning Methods: An Application to Register-Based Census. *Statistical Journal of the IAOS* 40, 1 (2024), 25–40.

[15] Yassine Guermazi, Sana Sellami, and Omar Boucelma. 2023. GeoRoBERTa: A Transformer-Based Approach for Semantic Address Matching. In *Joint Conference Proceedings of the Workshops of the EDBT/ICDT 2023 (CEUR Workshop Proceedings)*, Vol. 3379. CEUR-WS.org, Ioannina, Greece, 11.

[16] João Marcelo Borovina Josko, Marcio Katsumi Oikawa, and João Eduardo Ferreira. 2016. A Formal Taxonomy to Improve Data Defect Description. In *Lecture Notes in Computer Science*, Vol. 9645. Springer, Dallas, TX, USA, 307–320. https://doi.org/10.1007/978-3-319-32055-7_25

[17] Won Kim, Byoung-Ju Choi, Eui-Kyeong Hong, Soo-Kyung Kim, and Doheon Lee. 2003. A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery* 7, 1 (Jan. 2003), 81–99. https://doi.org/10.1023/A:1021564703268

[18] Ellen J Kinnee, Sheila Tripathy, Leah Schinasi, Jessie L C Shmool, Perry E Sheffield, Fernando Holguin, and Jane E Clougherty. 2020. Geocoding Error, Spatial Uncertainty, and Implications for Exposure Assessment and Environmental Epidemiology. *Int. J. Environ. Res. Public Health* 17, 16 (2020), 5845.

[19] Ioannis Koumarelas, Axel Kroschk, Clifford Mosley, and Felix Naumann. 2018. Experience: Enhancing Address Matching With Geocoding and Similarity Measure Selection. *Journal of Data and Information Quality (JDIQ)* 10, 2 (2018), 1–16.

[20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[21] David Loshin. 2010. *Master Data Management.* Morgan Kaufmann, Oxford, England.

[22] Mohammad Mahdavi. 2019. Raha. https://github.com/BigDaMa/raha/commit/9fb862677c6338e93f5ef008817bf54586414507 (Accessed: 2025-07-01).

[23] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning. *Proceedings of the VLDB Endowment (PVLDB)* 13, 11 (2020), 1948–1961.

[24] Mohammad Mahdavi and Ziawasch Abedjan. 2021. Semi-Supervised Data Cleaning with Raha and Baran. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*. VLDB, Copenhagen, Denmark, 7.

[25] 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 *Proceedings of the International Conference on Management of Data (SIGMOD)*. ACM, Amsterdam, 865–882.

[26] H. Müller and J.C. Freytag. 2005. *Problems, Methods, and Challenges in Comprehensive Data Cleansing.* Humboldt-Univ. zu Berlin, Berlin, Germany.

[27] Wei Ni, Xiaoye Miao, Xiangyu Zhao, Yangyang Wu, Shuwei Liang, and Jianwei Yin. 2024. Automatic Data Repair: Are We Ready to Deploy? *PVLDB* 17, 10 (2024), 2617–2630.

[28] Paulo Oliveira, Fátima Rodrigues, and Pedro Rangel Henriques. 2005. A Formal Definition of Data Quality Problems. In *Proceedings of the International Conference on Information Quality (ICIQ)*. MIT, Cambridge, MA, USA, 10.

[29] Vasileios Papastergios and Anastasios Gounaris. 2024. A Survey of Open-Source Data Quality Tools: Shedding Light on the Materialization of Data Quality Dimensions in Practice. arXiv:2407.18649 [cs.DB] https://arxiv.org/abs/2407.18649

[30] Deutsche Post. 2025. Address Study 2025 – Study on the Quality of Address Data in German. https://www.deutschepost.com/dam/jcr:8666c092-3b6c-424f-b131-125bf03674d6/dp-ddp-adress-studie-2025-en.pdf (Accessed: 2025-07-01).

[31] Erhard Rahm and Hong Hai Do. 2000. Data Cleaning: Problems and Current Approaches. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 23, 4 (2000), 3–13.

[32] Sergey Redyuk, Zoi Kaoudi, Volker Markl, and Sebastian Schelter. 2021. Automating Data Quality Validation for Dynamic Data Ingestion. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. OpenProceedings.org, Nicosia, Cyprus, 61–72. https://doi.org/10.5441/002/EDBT.2021.07

[33] Valerie Restat, Gerrit Boerner, André Conrad, and Uta Störl. 2022. GouDa - Generation of Universal Data Sets: Improving Analysis and Evaluation of Data Preparation Pipelines. In *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning (DEEM '22)*. ACM, Philadelphia Pennsylvania, Article 2, 6 pages. https://doi.org/10.1145/3533028.3533311

[34] Dimitrios Sargiotis. 2024. *Data Governance Policies and Standards: Development and Implementation.* Springer Nature Switzerland, Cham, 247–277.

[35] Sebastian Schelter, Tammo Rukat, and Felix Biessmann. 2021. JENGA: A Framework to Study the Impact of Data Errors on the Predictions of Machine Learning Models. In *Proceedings 24th International Conference on Extending Database Technology (EDBT 2021) Industrial and Application Track*. EDBT, Cyprus, 6.

[36] Sebastian Schelter, Philipp Schmidt, Tammo Rukat, Mario Kiessling, Andrey Taptunov, Felix Biessmann, and Dustin Lange. 2018. DEEQU – Data Quality Validation for Machine Learning Pipelines. In *32nd Conference on Neural Information Processing Systems (NIPS 2018)*. NeurIPS, Montréal Canada, 3.

[37] Kshitij Shah and Gerard de Melo. 2020. Correcting the Autocorrect: Context-Aware Typographical Error Correction via Training Data Augmentation. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC, Marseille, France.* European Language Resources Association, Marseille, France, 6930–6936. https://aclanthology.org/2020.lrec-1.856/

[38] Shreya Shankar, Labib Fawaz, Karl Gyllstrom, and Aditya Parameswaran. 2023. Automatic and Precise Data Validation for Machine Learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*. ACM, NY, USA, 2198–2207. https://doi.org/10.1145/3583780.3614786

[39] Ravindra Babu Tallamraju. 2022. Geographical Address Models in the Indian e-Commerce. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. ACM, New York, NY, USA, 5096–5097. https://doi.org/10.1145/3511808.3557515

[40] Heidi Carolina Tamm and Anastasija Nikiforova. 2024. Towards AI-Augmented Data Quality Management: From Data Quality for AI to AI for Data Quality Management. https://doi.org/10.48550/ARXIV.2406.10940

[41] Qin Tian, Fu Ren, Tao Hu, Jiangtao Liu, Ruichang Li, and Qingyun Du. 2016. Using an Optimized Chinese Address Matching Method To Develop a Geocoding Service: A Case Study of Shenzhen, China. *ISPRS International Journal of Geo-Information* 5, 5 (2016), 65.

[42] U.S. Department of Health and Human Services 2024. *HIPAA Privacy Rule To Support Reproductive Health Care Privacy.* https://www.federalregister.gov/documents/2024/04/26/2024-08503/hipaa-privacy-rule-to-support-reproductive-health-care-privacy (Accessed: 2025-07-01).

[43] Qinchen Yang, Zhiqing Hong, Dongjiang Cao, Haotian Wang, Zejun Xie, Tian He, Yunhuai Liu, Yu Yang, and Desheng Zhang. 2025. AddrLLM: Address Rewriting via Large Language Model on Nationwide Logistics Data. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1* (Toronto ON, Canada) *(KDD '25)*. ACM, New York, NY, USA, 2756–2767.

[44] Shuo Zhang, Zezhou Huang, and Eugene Wu. 2024. Data Cleaning Using Large Language Models. arXiv:2410.15547 [cs.DB] https://arxiv.org/abs/2410.15547