Automating Enterprise Data Engineering with LLMs

Jan-Micha Bodensohn[*]
DFKI & Technische Universität Darmstadt

Ulf Brackmann[*]
SAP SE & DFKI

Liane Vogel[*]
Technische Universität Darmstadt

Anupam Sanghi
Technische Universität Darmstadt

Carsten Binnig
Technische Universität Darmstadt & DFKI

Abstract

The automation of data engineering tasks is invaluable for enterprises to increase efficiency and reduce the manual effort associated with handling large amounts of data. Large Language Models (LLMs) have recently shown promising results in enabling this automation. However, data engineering tasks in real-world enterprise scenarios are often more complex than their typical formulations in the scientific community. In this paper, we study the challenges that arise when automating real-world enterprise data engineering tasks with LLMs. As part of the paper, we perform a case study on the task of matching incoming payments to open invoices, an instance of the entity matching problem. We also release a hand-crafted dataset based on the actual enterprise scenario to enable the research community to study the complexity of such enterprise tasks.[1]

1 Introduction

LLMs have shown promise for data engineering. Large Language Models (LLMs) have shown great potential to support the automation of a broad spectrum of data engineering tasks, such as column type annotation and entity matching [5, 8, 10]. Since LLMs are easy to use via prompting, they can render data engineering accessible to many practitioners. Therefore, they are a promising avenue for enterprises to automate processes without needing expensive, specialized solutions [17, 12]. However, recent work has shown that LLMs often do not work well out-of-the-box on enterprise data, as its characteristics differ vastly from the public datasets that LLMs are usually trained on [1, 15].

Data engineering in enterprises looks different. Apart from the distinctive characteristics of enterprise *data*, a second overlooked challenge is that the *tasks* in enterprise scenarios also differ from those formulated in the scientific community. Data engineering in enterprises is typically approached with broader business objectives in mind. One challenge, therefore, is that enterprises often work with business entities represented by multiple items in different tables. Moreover, enterprise tasks are often compounds of many individual steps. For example, since business objects are often spread over multiple tables, the first step is usually to define a view before the actual task (e.g., entity matching) can be executed. With this paper, we want to draw attention to the fact that data engineering tasks in real-world enterprise scenarios are, thus, typically much more challenging. To showcase this problem, we provide experiments from an initial case study demonstrating how these difficulties affect the performance of LLMs when used for such tasks.

Case study: Entity matching in enterprises. In this paper, we work on a real-world instance of the entity matching problem based on a scenario of the enterprise software company SAP. As illustrated

[*]Authors with equal contribution, alphabetical ordering.

[1]`https://github.com/DataManagementLab/llmeval-trl24`

Table Representation Learning Workshop at NeurIPS 2024

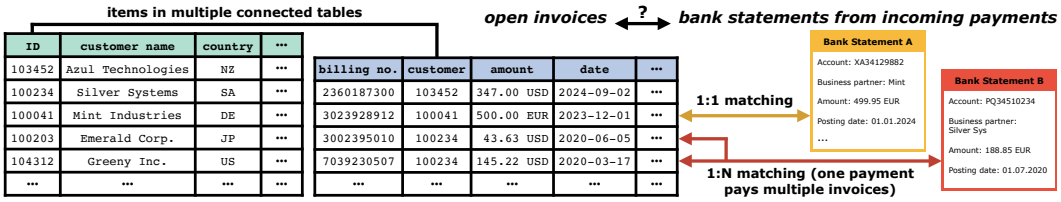

Figure 1: Real-world enterprise entity matching use case: matching incoming bank statements to open invoices. The task is complex due to the data being represented by multiple tables, the occurrence of 1:N matches, as well as human errors and small discrepancies in the transferred amounts.

in Figure 1, we use a scenario where incoming payments must be matched to open invoices, a task many companies face that is still solved with high manual efforts. This task is an example of a challenging enterprise problem where invoices and bank transfers are not of the same type, as is often the case in enterprise entity matching. Moreover, the data also presents challenges. For example, the memo lines of incoming bank statements are not standardized, and thus, human errors occur regularly, making automatic matching difficult.

Contributions. We make the following contributions: (1) We describe common complexities of data engineering tasks in enterprise scenarios. (2) We perform a case study on matching incoming payments to open invoices with the help of LLMs and pinpoint where the difficulties of enterprise tasks are. (3) We release our hand-crafted dataset for the payment-to-invoice matching task, which aims to resemble the data from the actual enterprise software system we have access to and mirrors some of the challenges to foster further research on enterprise data and tasks.

2 Data Engineering Tasks in Enterprise Scenarios

In real-world enterprise scenarios, data engineering tasks are often more complex than their typical formulations in the scientific community. In this section, we highlight general challenges we see in enterprise tasks and point out task-specific challenges specifically for entity matching in enterprises.

Compound tasks. While data engineering tasks in research are usually addressed as isolated problems, such as de-duplication [9] and missing value imputation [7], tasks in enterprise contexts are typically approached on a more holistic level concerning broader business objectives. Instead of focusing on individual tasks, enterprises aim to solve end-to-end workflows. One example beyond matching payments to invoices is *tariff classification*, which determines the correct commodity code when importing and exporting goods to ensure compliance with customs regulations. This process includes multiple separate steps: (1) Collecting and integrating structured and unstructured product information (materials, country of origin, etc.) from various sources (ERP systems, catalogs, product databases), as well as finding previous classifications of similar products. (2) Data cleaning and normalization, such as considering units of measurement. (3) Matching the products to the correct commodity code. While these steps can be executed sequentially, errors often propagate and amplify in later steps. As such, analyzing steps in isolation does not reveal the overall quality of the task.

Task-specific views. Data in enterprise systems often takes the shape of business objects represented by multiple rows stored in different tables. For example, a product might be represented by a table containing basic product information like weight and size. However, the used materials and information on how many products are in storage is stored in separate connected tables. For data engineering tasks, one must either (manually) construct views that extract the fields relevant to the task into a single table, or approaches have to deal directly with the complex 1:N and N:M table structures that form a business entity.

Matching between different types of entities. In public entity matching datasets, the entities to be matched are usually of the same type, like e-commerce products, restaurants, and scholarly articles [4, 11]. Meanwhile, enterprise scenarios often require matching between entities of different types, like payments to invoices or products to commodity codes, which might have overlapping but different sets of attributes, as shown in Figure 1.

Multi-matches. An additional challenge in enterprise scenarios is that the matches are often not 1:1 matches as in the literature, but can also be 1:N, N:1, or even N:M, making the problem much more difficult. In our payment-to-invoice matching scenario, it is quite common for a customer to pay

Table 1: F1 scores when matching payments to invoices. We incrementally increase the difficulty of the task by adding errors, multi-match cases, and representing invoices by multiple connected tables.

model	initial data	+ errors	+ multi-matches	+ multiple tables
GPT-3.5-Turbo-1106	0.97 ± 0.02	**0.96 ± 0.01**	**0.88 ± 0.01**	0.44 ± 0.02
GPT-4o-Mini-2024-07-18	**0.98 ± 0.01**	0.58 ± 0.04	0.52 ± 0.03	0.51 ± 0.02
GPT-4o-2024-08-06	**0.98 ± 0.01**	0.79 ± 0.03	0.66 ± 0.02	**0.58 ± 0.02**

multiple invoices with only one payment (1:N) or for one invoice to be paid by multiple payments (N:1), such as in the case of down payments or by holding back money due to quality issues with the product. Even a combination of both cases is conceivable (N:M).

Data complexity. The characteristics of enterprise data differ vastly from many publicly available datasets from the web [1]. Tables often have many more rows and columns and a higher sparsity. Furthermore, table names, attribute names, and cell values are often not descriptive and require domain-specific background knowledge to understand [3, 13]. Thus, the characteristics of enterprise data again amplify the difficulty of enterprise tasks.

Human errors and discrepancies. Whereas some cases in the payment-to-invoice matching scenario can be trivially solved using regular expressions, the fact that the memo lines of bank statements are not standardized often leads to errors that require more elaborate approaches to solve. For example, our analysis has shown that the way customers fill out bank statements differs considerably between regions. Moreover, human input often contains errors, such as missing zeros in reference numbers like `80000012345`. Besides these genuine mistakes, more sophisticated approaches must also deal with intentional discrepancies. For example, customers occasionally pay less than the invoice requires, which companies sometimes accept due to cost reasons if the amount is minimal. However, for larger differences, the matching must also work but has to emit an additional payment notification, further complicating the automatic comparison of amounts.

Extra context is required. Data engineering tasks in enterprise settings often require additional context information. For example, in the payment-to-invoice matching scenario, customers sometimes send so-called *payment advices* in an unstructured form, such as PDF documents or e-mails, that explain which invoices the customer intends to pay with a single payment. It is particularly challenging to detect if a specific incoming payment requires considering an additional payment advice. Today, many of these cases still require manual work.

3 Case Study: Matching Payments to Invoices

To empirically examine the challenges that arise when automating enterprise data engineering tasks with LLMs, we conduct a small case study on the payment-to-invoice matching task using a hand-crafted dataset. We design a process that generates invoices and payments following the characteristics of the actual data from an enterprise software system and mirroring some of the challenges described above. The full dataset contains $15,521$ invoices and $12,332$ payments, and we experiment on 790 matching pairs and $1,210$ pairs that do not match. To start, we formulate the entity matching task as a binary classification similar to existing literature [8, 10], prompting the LLM to decide if two table rows (one payment and one invoice) match. We additionally include one positive and one negative example in the prompt. Afterward, we make the task incrementally more complex. We compare the performance of three different GPT models from OpenAI [2], as shown in Table 1.

Experiment 1: Matching payments to invoices. In our first experiment, we incrementally increase the difficulty of matching payments to invoices based on scenarios we observe in actual enterprise data. In the following, we explain the different scenarios; the results are shown in Table 1:

1. (***initial data***) First, we run the models on clean 1:1 matches, where the payment memo line includes the correct reference numbers, the payment amount is exactly as stated in the invoice, and the customer name is also identical. All three models reach very high F1 scores in this setup.

2. ***(+ errors)*** Next, we add small errors and discrepancies to the data that typically occur in real-world data, such as missing or additional digits in the reference numbers or small discrepancies in the paid amount, which already lead to a drop in accuracy.

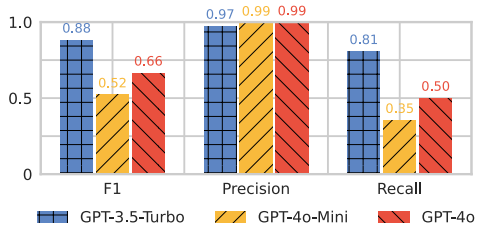

Figure 2: Precision and recall for the ***(+ multi-matches)*** scenario from Table 1. The GPT-4o models show lower recalls, resulting in lower F1 scores than GPT-3.5-Turbo.

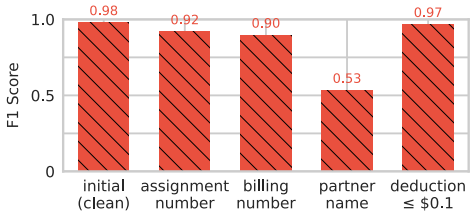

Figure 3: F1 scores for typical error categories in isolation using GPT-4o. Discrepancies in business partner names between invoices and bank statements are especially challenging.

3. ***(+ multi-matches)*** Next, we focus on multi-match cases, where either one payment pays multiple invoices or multiple payments together pay one invoice. As shown in Table 1, this vastly increases the task's difficulty. We further investigate the precision and recall to understand the high deviation in F1 scores between models (see Figure 2). We find that whereas all models achieve a very high precision, the differences in F1 scores are primarily due to differences in recall.

4. ***(+ multiple tables)*** Finally, we represent invoices using multiple connected tables instead of a single flat table. Our goal is to investigate if LLMs can work directly on these complex schemas instead of first creating a view. While metadata about each invoice document is stored in one table, specific information like the amount and due date are stored in a second table, and information about customers is stored in another separate customer table. In this scenario, GPT-4o and GPT-3.5-Turbo see large performance decreases compared to the previous experiments, indicating that the models have difficulties working with the complex data structures often used in enterprises.

Experiment 2: Typical real-world error categories. Our second experiment analyzes the different error types observed during payment-to-invoice matching on real-world data. Figure 3 compares the F1 scores for the initial clean data to each error type in isolation. The results show that each error type causes a decrease in performance, indicating that even minor discrepancies can introduce challenges for the LLM. However, the greatest performance drop is due to discrepancies in business partner names between invoices and bank statements. This may suggest that the model relies more heavily on textual data rather than numerical data and identifiers to perform the matching.

4 Discussion and Road Ahead

Our experiments with LLMs on the task of matching incoming payments to open invoices reveal a noticeable drop of about 40% in F1 score when transitioning from simple data to data that incorporates enterprise-specific challenges. Even when comparing manually created views, the F1 scores remain too low to eliminate the need for manually reviewing the matched instances. Furthermore, given the typical volume of thousands of transactions per day, performing pair-wise matching is not viable from both cost and performance perspectives. Our hand-crafted dataset includes only mild forms of errors and simple table structures with up to three tables. By contrast, the challenges are even more pronounced in real enterprise settings, suggesting that the drop in an actual enterprise setting could be significantly more severe.

To address these challenges, we argue that more robust LLMs tailored for enterprise tasks and data will make a significant impact. For example, we argue that LLMs are required which natively understand structured data more effectively, especially data which is represented as table structures of multiple tables [14]. Additionally, models must improve their handling of numerical data and demonstrate stronger reasoning capabilities to enable tasks that are composed of multiple steps. Future work thus involves trying out a more step-wise approach (chain-of-thought [16], etc.), where the necessary information is automatically extracted to a view before matching. Furthermore, techniques like retrieval-augmented generation (RAG) [6] could help to retrieve necessary context information, for example from internal documentation or e-mails. With these enhancements, we argue that the gap between model accuracy as well as the need for manual verification could be reduced significantly, thereby improving the overall efficiency of enterprise tasks.

Acknowledgments and Disclosure of Funding

This work has been supported by the BMBF and the state of Hesse as part of the NHR Program and the HMWK cluster project 3AI. It was also partially funded by the LOEWE Spitzenprofessur of the state of Hesse. We also thank DFKI Darmstadt and hessian.AI as well as Atreya Biswas from SAP for their support.

References

[1] Jan-Micha Bodensohn, Ulf Brackmann, Liane Vogel, Matthias Urban, Anupam Sanghi, and Carsten Binnig. Llms for data engineering on enterprise data. In *Joint Proceedings of Workshops at the 50th International Conference on Very Large Data Bases (VLDB 2024), Guangzhou, China, August 26 - August 29, 2024*, Tabular Data Analysis (TaDA) Workshop Proceedings, 2024.

[2] OpenAI et al. GPT-4 Technical Report, March 2024. arXiv:2303.08774 [cs].

[3] Jaewoo Kang and Jeffrey F. Naughton. On schema matching with opaque column names and data values. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 205–216, New York, NY, USA, June 2003. Association for Computing Machinery.

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

[5] Keti Korini and Christian Bizer. Column type annotation using chatgpt. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023), Vancouver, Canada, August 28 - September 1, 2023*, volume 3462 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[6] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[7] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. Capturing Semantics for Imputation with Pre-trained Language Models. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 61–72, Chania, Greece, April 2021. IEEE.

[8] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment*, 16(4):738–746, December 2022.

[9] Thorsten Papenbrock, Arvid Heise, and Felix Naumann. Progressive Duplicate Detection. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1316–1329, May 2015.

[10] Ralph Peeters and Christian Bizer. Using chatgpt for entity matching. In *New Trends in Database and Information Systems - ADBIS 2023 Short Papers, Doctoral Consortium and Workshops: AIDMA, DOING, K-Gals, MADEISD, PeRS, Barcelona, Spain, September 4-7, 2023, Proceedings*, volume 1850 of *Communications in Computer and Information Science*, pages 221–230. Springer, 2023.

[11] Ralph Peeters, Reng Chiz Der, and Christian Bizer. WDC products: A multi-dimensional entity matching benchmark. In Letizia Tanca, Qiong Luo, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani, editors, *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, pages 22–33. OpenProceedings.org, 2024.

[12] Yara Rizk, Praveen Venkateswaran, Vatche Isahagian, Austin Narcomey, and Vinod Muthusamy. A case for business process-specific foundation models. In Jochen De Weerdt and Luise Pufahl, editors, *Business Process Management Workshops - BPM 2023 International Workshops, Utrecht, The Netherlands, September 11-15, 2023, Revised Selected Papers*, volume 492 of *Lecture Notes in Business Information Processing*, pages 44–56. Springer, 2023.

[13] Alexandra Savelieva, Andreas Mueller, Avrilia Floratou, Carlo Curino, Hiren Patel, Jordan Henkel, Joyce Cahoon, Markus Weimer, Nellie Gustafsson, Richard Wydrowski, Roman Batoukov, Shaleen Deep, and Venkatesh Emani. The Need for Tabular Representation Learning: An Industry Perspective. *Table Representation Learning Workshop at NeurIPS 2022*, 2022.

[14] Liane Vogel, Benjamin Hilprecht, and Carsten Binnig. Towards Foundation Models for Relational Databases [Vision Paper]. *Table Representation Learning Workshop at NeurIPS 2022*, page 6, 2022.

[15] Adrian Vogelsgesang, Michael Haubenschild, Jan Finis, Alfons Kemper, Viktor Leis, Tobias Muehlbauer, Thomas Neumann, and Manuel Then. Get Real: How Benchmarks Fail to Represent the Real World. In *Proceedings of the Workshop on Testing Database Systems*, pages 1–6, Houston TX USA, June 2018. ACM.

[16] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[17] Michael Wornow, Avanika Narayan, Krista Opsahl-Ong, Quinn McIntyre, Nigam Shah, and Christopher Ré. Automating the enterprise with foundation models. *Proc. VLDB Endow.*, 17(11):2805–2812, 2024.