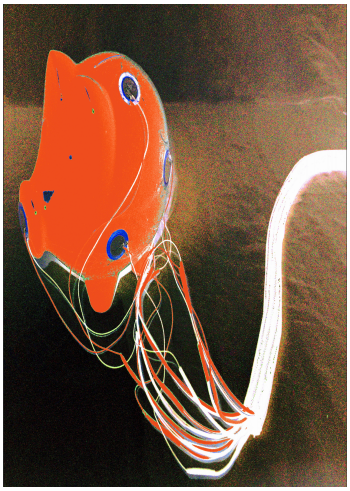
Navigating (personal) finances in a latent way

Pierre-Louis Wolfgang Léon Suckrow
Universitat der Kunste
MA Design & Computation
suckrowpierre@gmail.com

Abstract

This investigation presents an innovative method for personal finance management, by leveraging advancements in Natural Language Processing (NLP) and Large Language Models (LLMs). Aimed at automating the analysis of transaction history with minimal input, the system offers a natural language interface for interacting with banking data, addressing the need for an intuitive overview for financial optimization. The paper outlines the system architecture, including a user-friendly web application for data interaction, and describes the initial implementation phases, focusing on data preparation, aggregation, and augmentation processes. It emphasizes the role of LLMs, in enhancing transaction descriptions and extracting categories, thereby improving data readability and searchability. The database structure is designed to facilitate future developments, such as the incorporation of latent querying capabilities. The paper concludes with potential directions for enhancing the system's functionality, including the optimization of embedding functions, the application of hybrid search methods, and the integration of user-generated feedback to refine categorization and description enhancement processes.



I. Problem

Economic limitations associated with academic studies in a post covid world, characterized by rising inflation and unstable housing markets [9], makes it ever so harder for students to solely dedicate themselves entirely to their educational and research endeavors. According to the German Federal Statistical Office [1], in Germany, where I am currently residing, 37.9% of students face a risk of poverty (this figure surges to 76.1% for those living alone or with fellow students), and 24.2% grapple with excessive housing costs (56.6% for students living independently or with peers). Encountering these challenges firsthand prompted me to seek ways to reduce my variable expenses. This endeavor requires an analysis of one's transaction history, for me a system that always existed in a highly entropic state. Typically, banks provide transaction data in printed or digital spreadsheet formats, which for me are an over stimuli of numbers, cryptic transaction descriptions in different and changing formats. Nothing to gain a fast understanding or overview from. Only in sifting through this numerical footprint of one's past purchase decisions and expenditures could reveal insights into spending patterns, unnecessary expenses, and potential areas for financial optimization, albeit at the expense of considerable time needed for analysis. Drawing on my background in computer science and experimental physics, I contemplated the feasibility of creating a system that could automate this analysis with minimal input and maintenance. The recent advancements in Natural Language Processing (NLP) and Large Language Models (LLMs) and their transformative impact on our workflows inspired me to explore their application in this context. Furthermore, the innovative projects undertaken by my study peers, which applied these technologies to simplify complex German bureaucratic language [1] and to preserve an endangered language and culture by developing educational systems [4], encouraged me to pursue a solution within this technological domain.

For this I propose a system to enable personal finance management by making the core interaction method to one's transaction history natural language. The system takes in one's banking data and puts together one single coherent transaction history on which analysis can happen through hybrid latent representation, exploring the given data in a novel way.

I. State of the art

At the time of this publication, no comparable tools that leverage latent embeddings and Natural Language Processing (NLP) for personal financial analysis were identified. Finanzguru[2] offers a promise of personal finance management services; however, their software is not publicly accessible, leaving their implementation strategy undisclosed. The platform's offerings extend beyond multi-banking, contract Optical Character Recognition (OCR), and financial analysis to include features such as long-term analysis, predictions, and the capability to export data to CSV format. However, these advanced features appear to be restricted behind a subscription model. As a commercial entity, Finanzguru has the advantage of accessing APIs from various banks to retrieve users' balances and transaction histories. This unavailable API access to one's personal data to individual developers and users renders my implementation strategy more challenging and potentially diminishes user-friendliness.

The exploration of Language Models within the finance sector has received relatively modest attention in scholarly research. Lee et al. (2024) [5] provides a comprehensive overview and historical context of current Financial Large Language Models (FinLLMs), along with an assessment of their capabilities. Their research highlights models that have emerged from an increasing interest in applying Natural Language Processing (NLP) techniques for sen-

[1] https://www.destatis.de/DE/Presse/Pressemitteilungen/2022/11/PD22_N066_63.html

[2] https://finanzguru.de/

timent analysis, question-answering, and stock market prediction. Among the most recent advancements in this area are models such as *FinGPT* [11], *InvestLM* [12], and *BloombergGPT* [10]

II. Aim

This work aims to provide a possible outline for a novel system architecture to perform personal finance analysis using state-of-the-art advances in Natural Language Processing and Languages Models. It introduces an initial implementation in the form of a foundational prototype. This prototype serves as a base for subsequent experimentation and research, facilitating the iterative incorporation of new features.

The envisioned system aims to offer users an accessible method for analyzing their transaction data, potentially introducing novel approaches to navigate through this information via a latent space. Simultaneously, it should strive to ensure that the output remains accurate, precise, and factual.

III. Problem Space

Navigating through possible implementation strategies and choosing the right frameworks and tools while taking into account evolving constrained imposed by user behavior, hardware, software, development timelines, and developer skillset is a multi-objective optimization problem in itself.

IV. Application Design

Prior to delving into the data science aspects of this project, it is meaningful to first introduce the application through which users will interact with their data. Inspired by the user interfaces of *Stable Diffusion Web UI* [3], *Text Generation Web UI* [4], and *ComfyUI* [5], I determined that a web application would serve as an effective interface. This choice allows for the utilization of the extensive array of tools and libraries developed by the JavaScript community for the frontend while offering the flexibility to select appropriate technology stacks and tools for the backend to meet the system's requirements. Aiming to diverge from the *Gradio* [6] aesthetic and functionality, and to avoid the use of over-bloated JavaScript frameworks like *React* [7], I opted for a minimal, lightweight application that employs server-side rendering with *HTMX* [8]. This approach eliminates the need to separate the project into distinct front and back ends and obviates the necessity for building APIs for communication between these components. Given the data science orientation of the project, *Python* [9] emerged as the most suitable server-side language. The server is constructed using *FastAPI* [10], which offers multiple endpoints delivering HTML encapsulating the data to be displayed. *HTMX* enables the integration of dynamic and responsive elements within the HTML, allowing for the update of specific HTML components from the server in response to user input. The creation of pages and components on the server leverages the *jinja2* [11] templating library.

The application provides dedicated pages for configuring necessary settings, managing various bank accounts, reviewing the entire transaction history, and implementing personalized graphs and statistics. Each page is crafted as an encapsulated component, composed of nested elements, thereby adopting a modular design philosophy.

[3] https://github.com/AUTOMATIC1111/stable-diffusion-webui
[4] https://github.com/oobabooga/text-generation-webui
[5] https://github.com/comfyanonymous/ComfyUI
[6] https://www.gradio.app/
[7] https://react.dev/
[8] https://htmx.org/
[9] https://www.python.org/
[10] https://fastapi.tiangolo.com/
[11] https://palletsprojects.com/p/jinja/

V. Data

I. Data Preparation

Due to the inability to access one's personal banking data through the APIs of various banking institutions directly, users must manually supply their data in the form of CSV files. Although most banking institutions offer the option to export data as a CSV file, the challenge emerges from the varying data formats provided by different banks. Furthermore, a single bank may present different account types in diverse formats. Developing a system capable of automatically identifying these formats and converting them to the application's required format would be a complex task and could lead to the generation of inaccurate data, which is undesirable for precise financial analysis. To address this challenge, I suggest a graph-based workflow (see figure 1) that enables users to create the necessary mappings intuitively and engagingly.

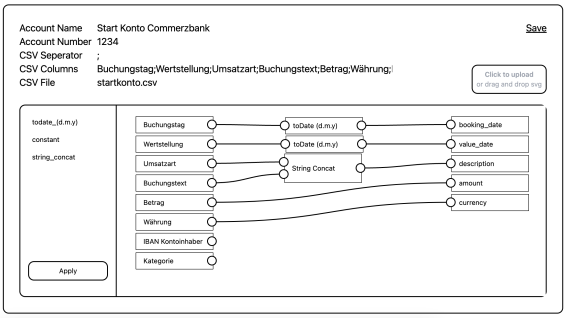


Figure 1: *Account settings element with interactive flowchart to set mapping of provided CSV columns*

Users can designate the name and account number for reference purposes. They are required to specify the CSV separator and the CSV column headings, from which the inputs for the flowchart diagram are dynamically generated. The user's task then involves linking these input fields to the corresponding output fields defined by the system. Furthermore, the interface allows for the addition of elements to modify column values, such as date formatting and string concatenation, through a drag-and-drop mechanism. This design ensures that new elements can be incorporated into the application based on user needs. The flowchart is visually rendered on the client side using the JavaScript *Drawflow* [12] library. The diagram's state is preserved in the database and reloaded as needed, adapting to any changes in the CSV column specifications. Once a mapping is established and a CSV file uploaded, this configuration can be applied to import and convert the CSV data into the system. To not halter the development of the prototype, it was decided to limit the upload functionality to one CSV file per bank account initially. Future updates will extend this feature to support the upload and storage of multiple CSV files, enabling continuous updates to the transaction history with the latest data.

By applying the flowchart diagram, the system generates a function mapping for each output node by traversing the graph backward until reaching every needed input node. This mapping is then employed to load the CSV data into a *Pandas* [13] dataframe, facilitating efficient row and column operations.

II. Data Aggregation

After converting each CSV file for the respective bank accounts, the subsequent step involves aggregating them into a unified transaction history. To ensure clarity for both LLM and human analysis in later stages, transactions between the provided bank accounts are omitted. This exclusion process involves comparing the booking and value dates of transactions and determining whether the amounts offset each other. Consequently, the refined dataset exclusively contains transactions that originate from or are directed outside the ecosystem formed by the given bank accounts.

[12] `https://github.com/jerosoler/Drawflow`

[13] `https://pandas.pydata.org/`

III. Data Augmentation

Transaction descriptions provided may often be cryptic, posing interpretative challenges for both machines and humans alike (refer to table 1). To address this, I propose a system designed to enhance these descriptions and to further extract potential categories from a predetermined list, tailored to each transaction. These categories are derived from personal use cases and are also influenced by those typically utilized in tax reporting.

The process of description augmentation and text classification leverages Large Language Models (LLMs) equipped with *function calling* [14] and *prompt engineering* [15] techniques. The LLM operates within a defined role (see Listings 1 & 3) and responds to specific prompts (see Listings 2 & 4). Prompt engineering alone is sufficient to enhance transaction descriptions effectively. However, for categorizing transactions, the LLM is required to produce a list of appropriate categories, achieved through the use of *function calling*. Typically, function calls enable the LLM to execute actions or retrieve data by invoking predefined functions. Function calling operates by defining a set of input parameters within the context of a prompt, where the LLM is tasked with generating values for these parameters based on the input text or query. This approach not only facilitates dynamic interactions but also enables the LLM to produce outputs tailored to specific requirements, such as generating a list of valid categories from transaction descriptions. Essentially, the LLM assesses the input text, executes a function call by populating the function's parameters with generated values, and returns an output in a predefined format. In the context of this project, function calling is applied to constrain the LLM's output to a structured list of categories (see Listing 5), enhancing the system's ability to categorize transactions accurately.

For fast prototyping the choice of LLM fell on the latest *OpenAI GPT-4* [6] API model *gpt-4-0125-preview* [16].

IV. Database

The database architecture currently in place and envisioned for future development is structured around three primary functions: storing user-specific settings, managing bank accounts along with their associated graph and CSV file references, and recording the transaction history. The implementation utilizes *SQLite* [17]. This approach represents a departure from the initial concept, which considered the use of Vector Databases such as *ChromaDB* [18] or *Marqo* [19]. The original idea was to load the transaction history into a Vector Database to facilitate latent search and querying capabilities. The deviation from the initial exploration of using Vector Databases for querying the transaction history is grounded in an understanding of how these databases operate and their applicability to the project's requirements.

Vector Databases store data as high-dimensional vectors, mathematical representations of features or attributes of the data. These vectors are generated by transforming or embedding the raw data—text, images, audio, video, and more—using machine learning models, word embeddings, or feature extraction algorithms. One of the key advantages of Vector Databases is their capability for fast and accurate similarity search and retrieval, contrasting traditional databases that rely on exact matches or predefined criteria. Vector Databases excel in applications requiring natural language processing, computer vision, and recommendation systems by identifying the most similar or relevant data points based on vector distance or similarity. [2]

[14] https://python.langchain.com/docs/modules/model_io/chat/function_calling

[15] https://platform.openai.com/docs/guides/prompt-engineering

[16] https://openai.com/blog/new-embedding-models-and-api-updates

[17] https://www.sqlite.org/

[18] https://github.com/chroma-core/chroma

[19] https://github.com/marqo-ai/marqo

However, the project's specific needs highlight a limitation in the k-nearest neighbor search methodology inherent in Vector Databases. While effective for many scenarios, the approach of only returning the k-nearest embeddings might not capture all relevant data points within the context of a given query. The desired query mechanism for this project should aim to identify **all** data points matching a provided string. This requires comparing the distances of all embeddings to the embedding of the query. A feasible task with the relatively low number of entries in the table of transaction history, compared to the typical rather large number of entries in typical use cases of vector databases. Calculating the distance of a data point embedding and the query embedding transforms the measurement of fitness into a one-dimensional metric with a lower bound of 0. Thus the extraction of meaningful data points is reduced into a binary classification task on one axis between fitting and non-fitting entries. For this conventional binary classification methods such as *K-Means Clustering* with $k = 2$, *DBSCAN*, or percentile-based strategies can be applied. Not needing to add an extra vector database to the architecture also reduces the complexity and overhead of the project.

Owing to time limitations, adding the embeddings and the latent querying of the database table of the transaction history, and was not completed by the time of this publication. These components are scheduled for inclusion in the near future.

VI. Distribution & Publishing

The source code for this project is published under `https://github.com/suckrowPierre/LatentAccounting` and is publicly available as an open-source project. The application can be run by cloning the repository and installing the Python dependencies defined in *requirements.txt* and executing *uvicorn app.main:app*. The database data is stored under *db/* while the CSV files of the accounts and the complete transaction history are under *csv_files/*.

VII. Current State

The initial prototype introduced provides a fundamental step towards intuitively navigating personal financial data using NLP and latent embeddings. To utilize this system, users must first specify a currency in the settings, selecting from the denominations listed in the *foreign exchange reference rates* by the *European Central Bank* [20]. In addition to setting the system currency, it is necessary to input an OpenAI key and select a model capable of function calling within the settings. Following these configurations, users can add an arbitrary number of bank accounts and upload a CSV file for each. To initiate the conversion of the CSV file, users must apply the mapping by clicking on the corresponding option in the flowchart diagram. Subsequently, the transaction history can be compiled from its designated page.

It is important to note that the current system design regenerates the entire transaction history from the beginning, erasing any previously generated history each time the process is initiated. This operation incurs calls to the OpenAI API, which may result in charges for the user. The user should carefully consider this when interacting with the system to manage potential costs effectively.

Once generated the complete transaction history is displayed. This history is currently searchable by checking if the search string is contained in the enhanced descriptions or the categories. Additionally, the history can be filtered by setting a time interval. From all displayed transactions, the system calculates and shows the total expenditure, revenue, and aggregate sum.

The process of augmenting descriptions with the *gpt-4-0125-preview* model generally yields

[20] `https://www.ecb.europa.eu/stats/policy_and_exchange_rates/euro_reference_exchange_rates/html/index.en.html`

satisfactory results. However, there are occasional minor inaccuracies. The categorization of transactions correctly aligns most of the time, though there may be instances where one or two categories do not seem appropriate. Moreover, in cases where descriptions lack sufficient data or clues regarding the applicable category, the model tends to assign all possible categories, which is not an ideal outcome.

Despite the limitations inherent in its preliminary implementation and the absence of latent querying capabilities, the tool is already capable of providing meaningful insights into one's financial data. The textual enhancement of descriptions from banking institutions improves readability, while the extraction of categories facilitates basic search functionality. These features can empower users to gain a clearer understanding of their expenditures and income sources, encouraging efforts to optimize them. This outcome aligns with the original objective set for this project.

VIII. Privacy

Given the sensitive nature of the data involved in this project, it is crucial to minimize the risk of it falling into unauthorized hands. While *OpenAI* assures that data processed through its API is not retained, the lack of transparency concerning the handling of data within this "black box" raises concerns for individuals. Consequently, processing embeddings and employing further LLMs locally on the user's device could offer a more secure alternative. However, the significant computational demands and hardware requirements might render this solution impractical for some users. Should the system be hosted on a remote server in the future, ensuring security and trustworthiness will be paramount. Additionally, the imperative of safeguarding data privacy limits the potential for aggregating data across users to optimize the system universally, likely restricting it to individual optimizations.

IX. Where do You go from here ?

As highlighted, the present state of implementation captures merely a segment of the application's full potential, opening a broad "attack vector" for further optimization.

The addition of the embedding should be the first thing followed up. Selecting an appropriate embedding function is crucial for converting both the enhanced descriptions and categories, as well as incoming queries, into a unified embedding format. An ideal embedding function would be one that can be executed efficiently on a local machine. To identify the most suitable option, it is advisable to evaluate several candidates, choosing the one that creates the most separable distances given a set of expected queries.

With a focus on keywords in the proposed system, incorporating a *hybrid* search/retrieval method through the introduction of *sparse vectors* could enhance search capabilities by integrating it with the use of "normal" *dense vectors*. Vector databases such as *Pinecone* [21], *qdrant* [22], and *Weaviate* [23] employ this technique to yield more accurate search results. While *dense vectors* are adept at capturing semantic features, *sparse vectors* are more directly aligned with specific keyword values. The hybrid search approach then allows for a parameter to be set, dictating whether the search results should lean more towards the insights provided by *sparse* or *dense* embeddings, facilitating a more tailored search experience.

The concept of *prototypes*, as introduced by Eleanore Rosch [8] and further explored by Daniel Hromada [3], could also provide a valuable direction for enhancing the processes of querying or category extraction within the system. This theoretical foundation offers a method for conceptualizing and organizing

[21] https://www.pinecone.io/learn/hybrid-search-intro/

[22] https://qdrant.tech/articles/sparse-vectors/

[23] https://weaviate.io/blog/hybrid-search-explained

data that may facilitate more efficient and meaningful interpretations and interactions with the system's functionalities.

Given the current category extraction process is prone to errors, the application should enable users to alter and validate both the categories and the enhanced prompts. This capability would facilitate the creation of a validated dataset, laying the groundwork for further system optimization. Moreover, if incorrect mappings are recorded, employing *Direct Preference Optimization* [7] could refine the Large Language Model's (LLM) performance with respect to categorization and description enhancement. This approach involves providing examples of both correct and incorrect outputs for the specific tasks, thereby fine-tuning the LLM to better align with the objectives of the application.

The selection of prompts significantly influences the results of categorization and augmentation. Identifying the optimal prompt can be approached as an optimization problem, wherein the prompt is considered a trainable parameter. This parameter can be optimized through gradient descent, leveraging data validated by the user regarding categories. Lillian Weng (2023)[24] provides a comprehensive overview of potential methodologies for this process. This approach emphasizes the dynamic nature of prompt engineering, underscoring the importance of iterative refinement based on user feedback to enhance the application's performance.

Should the system accumulate sufficient validated data for accurate classification, it may become viable to reconsider the use of Large Language Models (LLMs) for categorization. In this scenario, evaluating the effectiveness of a self-trained, NLP-driven multi-label classification system [25] becomes appropriate. This approach would assess whether such a model could surpass LLMs in terms of inference speed and accuracy, potentially offering a more efficient and precise mechanism for categorization within the application.

For a more detailed insight into one's spending habits and financial situation, incorporating receipts from purchases into the transaction history could enrich the data set with information on individual items purchased. This could be achieved by utilizing Optical Character Recognition (OCR) technology on provided receipts, followed by the extraction of enhanced descriptions and categories as previously outlined. Additionally, compiling receipts within this system could prove advantageous for tax reporting purposes, offering a comprehensive and organized collection of financial transactions. Not only limited to traditional bank entities the integration of crypto wallets could also be explored.

The system's utility could extend beyond merely optimizing financial objectives. With the detailed transaction history, it becomes possible to derive other meaningful insights and evaluate various goals, or even to integrate these insights with other systems. For instance, it could indirectly measure achievements related to nutritional goals or expenditures on family activities, entertainment, cultural engagement, and more, thus broadening its applicability and enhancing its value to users.

Incorporating dedicated Large Language Models (LLMs) for querying the data represents a vital step toward making data analysis more accessible and interaction more engaging. It's essential to evaluate various models and strategies, such as employing one of the Lee et al. (2024) [5] proposed FinLLMs or creating an SQL agent to enable an LLM to create SQL queries [26], or even use function calling to query the database from the LLM with the above-described embedding search.

Additionally, the system would benefit from a feature to keep the transaction history up to date, by providing the latest CSV files. It

[24] https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/

[25] https://huggingface.co/blog/Valerii-Knowledgator/multi-label-classification

[26] https://python.langchain.com/docs/use_cases/sql/agents

should verify which CSV files have been previously processed by comparing the HASH value of the file's content and identify the entries already added to the database by creating a hash index from the account ID, original description, transaction value, and booking date. This approach would eliminate redundant utilization of LLMs, enhancing the system's efficiency and responsiveness.

Users should also have the capability to modify and define categories themselves, fostering a more customized experience. This flexibility allows users to tailor the categorization process to better reflect their personal or business financial activities, thereby increasing the system's utility and relevance to their specific needs.

X. Acknowledgments

I extend my gratitude to Prof. Dr. Daniel D. Hromada and Christian Schmidts for their guidance, expertise, and encouragement, which were pivotal in bringing this project to fruition. Additionally, my appreciation goes to my fellow students from Design & Computation [27] for their collaborative spirit, exchange of ideas, and mutual support throughout this journey into exploring "Wicked Solutions." This collective effort has significantly enriched the project's development and outcome.

References

[1] Franz Christopher Xaver Hagen. "Overcoming Bureaucratic Exclusion: Addressing the Challenges Faced by Foreigners in Germany". In: *Wicked Solutions* 0 (2024).

[2] Yikun Han, Chunjiang Liu, and Pengfei Wang. *A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge*. 2023. arXiv: 2310.11703 [cs.DB].

[3] Daniel Hromada. "Genetic localization of semantic prototypes for multiclass retrieval of documents". In: *17th Conference of Doctoral Students ELITECH '15*. 2015, pp. 1–7. DOI: 10.25624/kuenste-1379.

[4] Lilli-Chiara Kurth. "Rěc :: A Language Learning System". In: *Wicked Solutions* 0 (2024).

[5] Jean Lee et al. *A Survey of Large Language Models in Finance (FinLLMs)*. 2024. arXiv: 2402.02315 [cs.CL].

[6] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL].

[7] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2023. arXiv: 2305.18290 [cs.LG].

[8] Eleanor Rosch. "Principles of categorization". In: *Concepts: core readings*. Ed. by Eric Margolis and Stephen Laurence. The MIT Press, 1999, pp. 189–206.

[9] Manuel Sinn and Sena Doğan. "The Housing Problem". In: *Wicked Solutions* 0 (2024).

[10] Shijie Wu et al. *BloombergGPT: A Large Language Model for Finance*. 2023. arXiv: 2303.17564 [cs.LG].

[11] Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. *FinGPT: Open-Source Financial Large Language Models*. 2023. arXiv: 2306.06031 [q-fin.ST].

[12] Yi Yang, Yixuan Tang, and Kar Yan Tam. *InvestLM: A Large Language Model for Investment using Financial Domain Instruction Tuning*. 2023. arXiv: 2309.13064 [q-fin.GN].

[27] https://www.newpractice.net/study

XI. Appendix

Description	Augmented Description	Categories
REWE Markt GmbH-Zw Berlin	Payment made to REWE Markt GmbH, a supermarket chain, located in Berlin.	['groceries']
OBI Bau- und HeimwerkermaBerlin	Payment made at OBI, a home improvement and DIY store in Berlin.	['equipment', 'hobbies', 'home improvement']
Lastschrift Telefonica Germany GmbH + Co. OHG *** Kd-Nr.: ***, Rg-Nr.: ***, Ihre Tarifrechnung End-to-End-Ref.: ** Rückgabegrund: Rückgabe mangels Deckung	Received a direct debit from Telefonica Germany GmbH + Co. OHG for your tariff invoice, returned due to insufficient funds.	['phone']
UZR*Klosterapotheke beim Munchen	Payment made to UZR Klosterapotheke, a pharmacy located near Munich.	['health', 'groceries']
Lastschrift PayPal Europe S.a.r.l. et Cie S.C.A ***/. Bolt Operations OU, Ihr Einkauf bei Bolt Operations OU End-to-End-Ref.: *** Mandatsref: *** Gläubiger-ID:*** SEPA-BASISLASTSCHRIFT wiederholend	Payment made via PayPal to Bolt Operations OU for a purchase.	['taxi', 'transport']
Überweisung ZenJob GmbH Lohn / Gehalt 10/2023 End-to-End-Ref.: *** Kundenreferenz: ***	Salary payment for October 2023 from ZenJob GmbH.	['business income']
Überweisung *** Starthilfe Berlin End-to-End-Ref.: *** Kundenreferenz: ***	Received a transfer from *** for "Starthilfe Berlin."	['financial aid']
Lastschrift PayPal Europe S.a.r.l. et Cie S.C.A ***/. Thomann GmbH, Ihr E inkauf bei Thomann GmbH End-to-End-Ref.: *** Mandatsref: *** Gläubiger-ID: *** SEPA-BASISLASTSCHRIFT wiederholend	Payment via PayPal for a purchase from Thomann GmbH, a recurring SEPA direct debit transaction.	['equipment', 'subscriptions and membership', 'music', 'hobbies']
Lastschrift PayPal Europe S.a.r.l. et Cie S.C.A ***. , Ihr Ei nkauf bei End-to-End-Ref.: *** Mandatsref: *** Gläubiger-ID:*** SEPA-BASISLASTSCHRIFT wiederholend	Recurring SEPA direct debit payment to PayPal Europe for a purchase, with reference number ***	['subscriptions and membership', 'clothing', 'entertainment', 'groceries', 'restaurant', 'health', 'personal care', 'sports', 'utilities', 'internet and cable', 'phone', 'electricity', 'gas', 'travel', 'equipment', 'hobbies']

Table 1: *Selected rows of example transaction data with descriptions and the resulting augmented descriptions and captions after data augmentation*

- INVESTMENT
- RENT
- INSURANCE
- BUSINESS INCOME
- PASSIVE INCOME
- TAX
- FINANCIAL AID
- FOOD
- GROCERIES
- RESTAURANT
- CLOTHING
- GIFTS AND DONATIONS
- ENTERTAINMENT
- SUBSCRIPTIONS MEMBERSHIP
- EQUIPMENT
- MUSIC
- HOBBIES
- HEALTH
- SPORTS
- PERSONAL CARE
- SAVINGS
- LOAN
- FEES
- UTILITIES
- INTERNET CABLE
- PHONE
- WATER
- ELECTRICITY
- GAS
- TRAVEL
- VACATION
- TRANSPORT
- CAR
- PUBLIC TRANSPORT
- TAXI
- FLIGHT
- FUEL
- EDUCATION
- LITERATURE
- ANIMALS
- FAMILY

Table 2: *Possible keywords that can be extracted from the enhanced description*

Listing 1: *System Role of the LLM to augment the description*

```
Your are an intelligent system that
takes in a transaction description
from my banking history and enhances
that description to make it more
readable and understandable for an
entry in a database. The description
should be on sentence. Don't
incoperate any infromation about the
payment method , date, transaction
number or other unnecessary detail.
Just a short context about the
transaction. Do not hallucinate
or add any information that is not
present in the original description.
If the descriptions is not in
english please translate it into
english.
```

Listing 2: *User Role of the LLM to augment the description*

```
Please enhance the following
transaction description from my
banking history. The description
should be one sentence. Don't
incoperate any infromation about
the payment method, date,
transaction number or other
unnecessary detail. Just a short
context about the transaction. Do
not hallucinate or add any
information that is not present in
the original description. If the
descriptions is in german please
translate it into english.
Description:
```

Listing 3: *System Role of the LLM to extract the categories*

```
Your are an intelligent system that
takes in a transaction description
from my banking history and extracts
the best matching categories for the
transaction from the following list
and gives me back a json array with
the best matching categories. Form
a precise decision. Do not
hallucinate. Don't bload your
answers with non matchign
categories. If you are uncertain
leave the catergory out. The
categories are:
```

Listing 4: *User Role of the LLM to extract the categories*

```
Please take in the following
transaction description from my
banking history and extract the
best matching categories. Give me
back a json array with the best
matching categories. Form a precise
decision. Do not hallucinate. Don't
bload your answers with non matchign
categories. If you are uncertain
leave the catergory out. The
description is:
```

Listing 5: *Functioncall to extract categories based on transaction description*

```
function_schema = {
    "name": "extractCategories",
    "parameters": {
        "type": "object",
        "properties": {
            "categories": {
                "type": "array",
                "items": {
                    "type": "string",
                    "enum": [category.value for category in Categories]
                }
            }
        },
        "required": ["categories"]
    }
}
```