Automated deal amount validation in CRM systems using large language models*

Mykhaylo Petryk[1,†], Oleksiy Tsebriy[1,†], Yurii Stoianov[1,*,†] and Oksana Petryk[1,†]

[1] Ternopil Ivan Puluj National Technical University, Ruska 56 46000 Ternopil, Urkaine

Abstract

In modern CRM-driven sales environments, the accuracy of deal data is critical for effective decision-making and operational efficiency. This paper presents an approach to automate the validation of deal amounts entered by sales managers using large language models (LLMs). Manual verification processes are time-consuming and prone to human error, while LLMs, when guided by clear instructions, can perform intelligent validation based on predefined business rules. The proposed solution demonstrates how LLMs can enhance data integrity, reduce manual workload, and support scalable CRM workflows in IT companies focused on ready-made CRM solutions.

Keywords

CRM systems, Large language models, Data validation, Sales automation

1. Introduction

An IT company focused on selling off-the-shelf solutions to facilitate work with CRM systems that require verification of a salesperson's estimate of the transaction value. This meets a business need to ensure the accuracy and reliability of data, as manual verification is time-consuming. It was decided to develop an automated service to check the accuracy of the transaction value entered by the manager. Large Language Models (LLMs) are best suited for this purpose. LLMs allow you to set the decision logic using instructions.

2. Algorithm

1. Identification of the fact of discussion of the amount.
2. Extracting the email in which the amount was discussed and 2 neighboring ones.
3. Analyze emails for information about the product (name, quantity, cost).
   a) If there is information about the product name, quantity, and cost, LLM calculates the transaction value independently;
   b) If there is information about the name and quantity, then the most similar product in the price list is found using a semantic search and its cost is extracted. After that, the cost of the product is multiplied by the quantity;
   c) In other cases, there is not enough information for LLM to calculate the cost.
4. The last step is to compare the calculated cost with the one specified by the manager for classification.

* CITI'2025: 3rd International Workshop on Computer Information Technologies in Industry 4.0, June 11–12, 2025, Ternopil, Ukraine

[1,*] Corresponding author.

[†] These authors contributed equally.

mykhaylo_petryk@tntu.edu.ua (M. Petryk); oleksii_tsebrii@tntu.edu.ua (O. Tsebrii); yuriy556s@gmail.com (Y. Stoianov); oopp3@ukr.net (O. Petryk)

0000-0001-6612-7213 (M. Petryk); 0000-0003-1848-2258 (Y. Stoianov); 0000-0001-8622-4344 (O. Petryk)

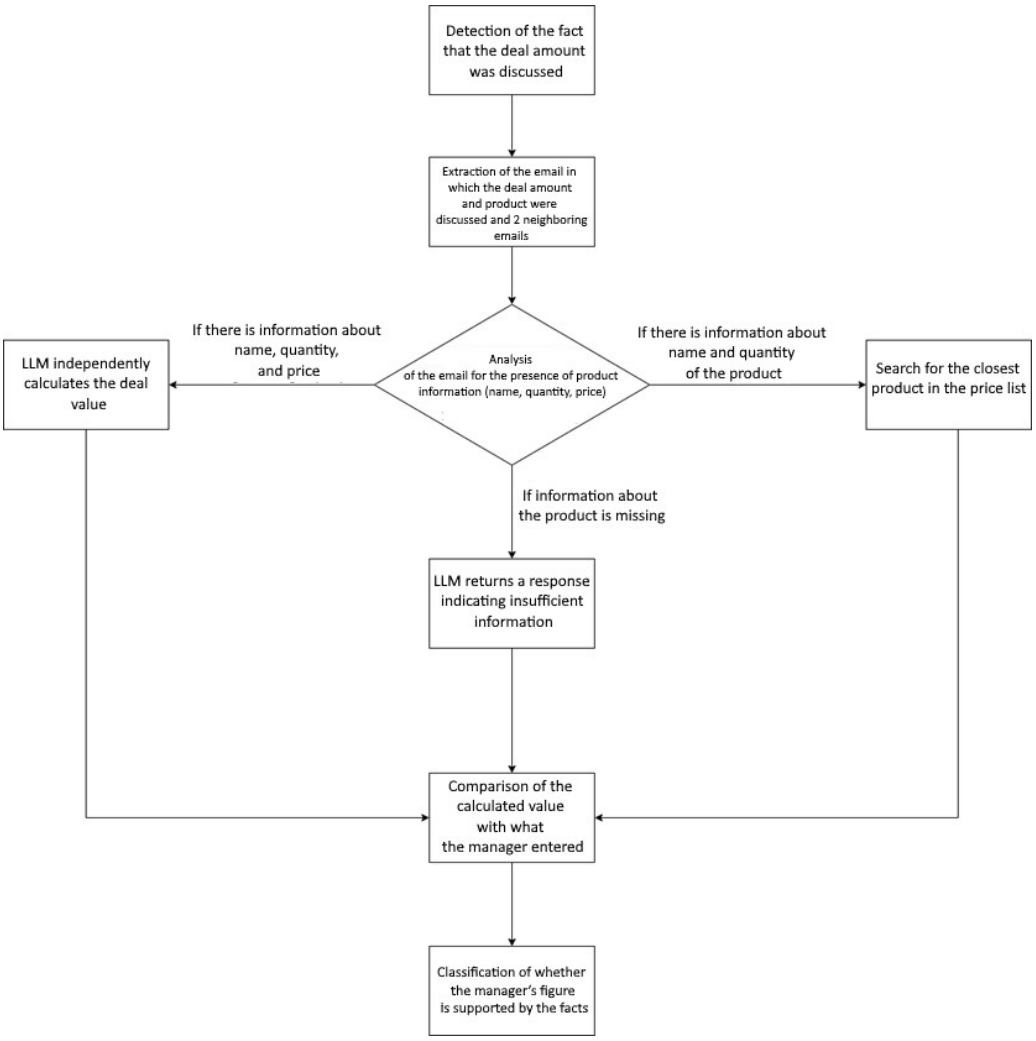

Figure 1: Algorithm steps.

3. Libraries used

Langchain is an open-source framework designed to simplify the development of applications based on large language models (LLMs) [1]. Langchain is widely used to extend the customization capabilities of large language models, ensuring the relevance of the generated information.

Sentence-transformers is a library used to calculate text and image embeddings. Sentence-transformers also include the ability to calculate the similarity between two texts, search, classify and cluster texts.

4. Development

4.1. Design

An LLM agent was used to develop an automated service for checking the correctness of the transaction amount entered by the manager [2]. An LLM agent is a specialized program or system that uses the capabilities of large language models to perform various tasks related to natural language processing. They can perform such functions as text generation, answering questions, translation, text analysis, and other tasks that require understanding and generation of human speech. An important difference between LLM and LLM agents is the application aspect [3]. LLM is the underlying technology that provides language capabilities, while LLM agents use this technology to perform specific tasks and functions. LLM agents can be customized for specific

contexts and purposes, making them more effective in solving specific problems. Also, depending on the context, LLM agents are able to use different utilities for better decision-making.

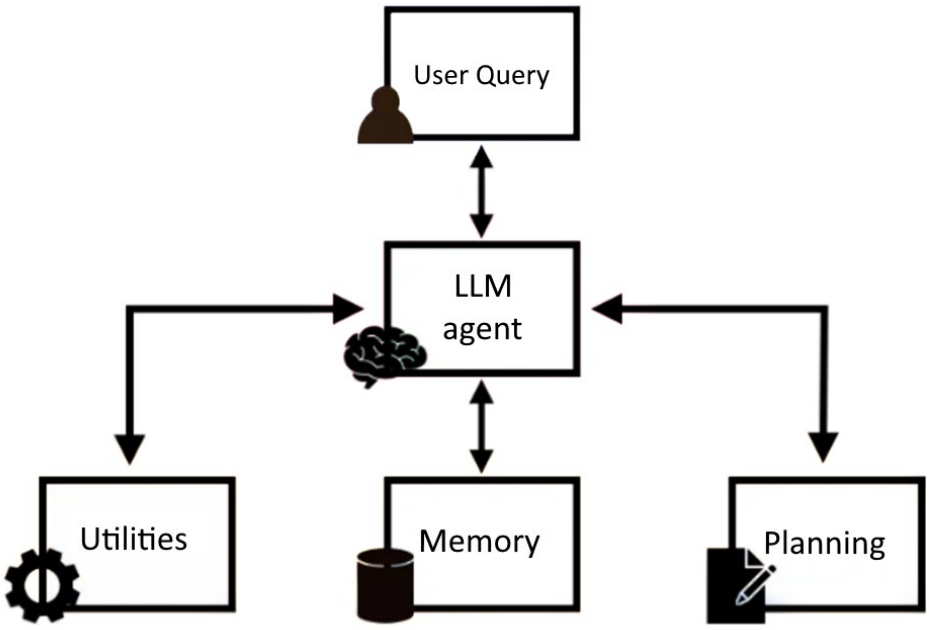

Figure 2: Usage of the LLM agent.

- In general, an LLM agent can consist of the following main components:
- User query - the user's question or request;
- LLM agent/brain - the core of the agent that acts as a coordinator;
- Planning - assists the agent in planning future actions;
- Memory - manages the agent's past behavior.

4.2. Implementation

During the experiment, two models from OpenAI were used to develop LLM agents: GPT-3.5 Turbo and GPT-4 [4]. The difference between the GPT-3.5 Turbo and GPT-4 models is as follows: GPT-4 generates higher quality and more complex text than GPT-3.5 Turbo. GPT-4 has a better understanding of context and can store more information over a longer dialog. GPT-4 is a larger model with more parameters, which allows it to learn better and generate more relevant answers. GPT-3.5 Turbo is optimized for faster performance and lower computing resource requirements. GPT-4 is more commonly used for complex tasks that require high accuracy and deep understanding, while GPT-3.5 Turbo is suitable for more general tasks and when speed is important. For this task, GPT-4 is better suited because it has the ability to understand the context of the correspondence to extract the necessary information and generate better text that is used to evaluate and improve the algorithm.

To improve the efficiency of LLM agents, hint engineering is used to interact with other inputs [5]. These hints describe the tasks that the LLM agent needs to perform and provide the necessary context to generate relevant and high-quality results. Each hint should contain three main elements:

- Role - indicate who the LLM is supposed to answer for (assistant, expert, etc.).
- Objective - clearly define the expected result (generating text, answering questions, etc.);
- Additional information - any additional information that will help the LLM to perform the task better (examples, context, limitations, etc.).

In addition to the basic elements, the prompts can also be improved using zero-shot, one-shot, and multiple-shot learning methods.

Zero-shot - the LLM uses only its own knowledge to make decisions and generates an answer based on that;

- one-shot - the LLM uses one example of how a task is performed to make decisions. The example describes the sequence of steps and the logic of solving the problem;
- multiple-shot - LLM uses multiple examples of how a task is performed to make decisions. The examples describe the sequence of steps and the logic of solving problems in different contexts;

Following these rules will help you to use LLM agents effectively and get the desired results [6]. These guidelines were applied during the development of the LLM agent:

"Use the following format:
Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
Final Answer: the final answer to the original input question."

5. Example

We will now demonstrate an example of how the LLM agent works on several transactions. To begin with, the algorithm accepts only data about the transaction and later uses utilities to extract correspondence that is likely to contain the fact of price negotiation:

ID - 4325A897E2,
Name - Blast Co,
Manager_Amount - 27920,
Email: Oliver,
Our CFO has made his final decision. He will approve the agreement for 13 HG and 3 BK licenses, but the budget doesn't cover the implementation fee. His initial budget was $26k, but he can stretch it to $29,680 for the licenses.
If you can waive the implementation fee, he'll sign the agreement immediately once you send the DocuSign. His name is Liam Brown.
Thanks, Olivia.

After analyzing this agreement, you can see that the client reports the name and quantity of the desired products that have been approved by the CFO.

The next step is to use a utility to extract the cost of the product from the price list to be able to calculate the amount. For this purpose, a semantic search algorithm will be used. The names of the products found in the emails and the names of all available products from the price list are converted to a numeric format. After that, they are compared using cosine similarity, and as a result, the cost of the product with the highest value is obtained and multiplied by the number of subscriptions. If the email contains several names, then we find the sum of the products.

In this example, the names of the products from the correspondence coincide with the one in the price list, the LLM agent returned *{HG: 19890, BK:8910}*. The sum of these values is 28800. The final step is to compare the number calculated by the LLM agent and the number specified by the manager. In this case, these are the numbers 27920 and 28800. We can assume that the manager has specified the price correctly. Despite the fact that these numbers are different, the instructions also state that the numbers can differ by no more than 10%, because a potential discount could be burned. After that, the LLM agent displays information that the price of the transaction has been confirmed and displays the information on the basis of which it made its decision.

Here is the next example:

ID - 43257C392E,
Name - Chango,
Manager_Amount -2128,
Email: "Oliver,
"Dear Chris,

Absolutely, your interpretation is spot on and in harmony with the detailed breakdown in the revised proposal. We're kicking off with the $9.50/user/month rate. The sum of the contract values for Year 1, comprising 144 licenses, is $1368, plus the Extra Plan at $520 and the DS at $240, bringing us to a total of $2128.

We recognize and respect your hesitation to commit to a 3-5 year term at this juncture. In light of this, let's pivot to a middle ground - a 24-month term."

To find the final amount, LLM repeated the steps described in the previous example. In this email, the client specifies the quantity, name, and price of the desired products, so LLM easily recalculated the transaction amount. As a result, the number coincides with the one specified by the manager, which means that the transaction amount is supported by the fact that it was discussed in the email.

In the previous examples, you could determine the value of the deal using emails, and the amount was the same as the one specified by the manager. However, there are cases when the need for a different amount of product could be discussed in an email than during live discussions. Because of this, the LLM agent may make mistakes because he or she does not have information about the outcome of the discussions. However, if we take into account the amount of information received by the LLM agent, the result is correct. This example shows the importance of high-quality information that is passed to LLM agents to obtain the desired results.

6. List of material structure

- *main.py* - the file is the main module of the program that runs the main functionality.
- *agent.py* - the file in which the LLM agent is created.
- *template.py* - the file in which the instructions are described.
- *tool.py* - a file in which functions are created for use in utilities.

The *main.py* script reads the input dataset, creates utilities, launches the LLM agent, and saves the output dataset with the results.

The *agent.py* script is responsible for creating an agent and configuring the parameters necessary for correct operation.

The *tool.py* script creates the functions necessary for the utilities, namely, extracting the necessary information about transactions from the input dataset and finding the price of the product using semantic similarity for further calculation.

7. Validation

To validate the LLM agent's performance, 50 deals were selected. Only deals that had been successfully completed within the last 6 months were selected. For these deals, a dataset was formed with the relevant data, namely: deal ID, name, opening date, closing date, amount provided by the manager, actual amount at the time of closing, and correspondence. We also used a price list containing the name of the service and its price.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Topsakal, Oguzhan, and Tahir Cetin Akinci. "Creating large language model applications utilizing langchain: A primer on developing llm apps fast." International Conference on Applied Engineering and Natural Sciences. Vol. 1. No. 1. 2023.

[2] What are LLM Agents? A practical guide to LLM agents and functions, 2024. URL: https://www.k2view.com/what-are-llm-agents/

[3] TSAO, Wen-Kwang. Multi-agent reasoning with large language models for effective corporate planning. In: 2023 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2023. p. 365-370.

[4] Openai platform. Models, explore all available models and compare their capabilities., 2025. URL: https://platform.openai.com/docs/models

[5] FU, Jinlan, et al. Hint-before-solving prompting: Guiding LLMs to effectively utilize encoded knowledge. *arXiv preprint arXiv:2402.14310*, 2024.

[6] Openai. A practical guide to building agents, 2025. URL: https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf