Degree Project

Computer Engineer 180 credits

Optimizing chemical risk documentation with LLMs

Final Report

Computer Science and Engineering 15 credits

Halmstad June 8, 2025

Faraj Hamedy

Alaa Mouselli

HALMSTAD
UNIVERSITY

Abstract

This research presents an innovative solution based on artificial intelligence, which is used for verifying data in databases. The system aims to detect errors, including duplicate rows, empty fields, or inaccurate chemical product descriptions stored in the database. The project was primarily designed to support industrial safety assessments by automating data verification through comparisons with external sources using the GPT-4 Turbo language model.

The solution incorporates advanced prompt engineering techniques, including retrieval-augmented generation (RAG), where validated feedback examples are dynamically retrieved and inserted into the model prompt. This approach enables the system to adapt to domain-specific knowledge without requiring fine-tuning. Simulated professional testing demonstrated measurable improvements in error detection and risk prioritization, with accuracy rising from 53,3% to 83,3% after iterative prompt refinement and feedback-driven retrieval.

The project addresses challenges such as reducing LLM hallucination, ensuring feedback reliability, and meeting safety-critical application requirements. It offers a scalable foundation for semi-automated chemical documentation validation in industrial environments.

In conclusion, this demonstrates that artificial intelligence can be integrated in order to improve data quality and thus minimize the human burden in environments that require both accuracy and speed.

Keywords: Error detection, automation, database, prompt, feedback loop, retrieval-augmented generation (RAG), Large Language Model (LLM).

Sammanfattning

Denna forskning presenterar en innovativ lösning baserad på artificiell intelligens, som används för att verifiera data i databaser. Systemet syftar till att upptäcka fel, inklusive dubblettrader, tomma fält eller felaktiga beskrivningar av kemiska produkter som lagras i databasen. Projektet har främst utformats för att stödja industriella säkerhetsbedömningar genom att automatisera dataverifiering via jämförelser med externa källor med hjälp av språkmodellen GPT-4 Turbo.

Lösningen innehåller avancerade tekniker för promptteknik, inklusive retrieval-augmented generation (RAG), där validerade feedbackexempel dynamiskt hämtas och infogas i modellens prompt. Detta tillvägagångssätt gör det möjligt för systemet att anpassa sig till domänspecifik kunskap utan behov av finjustering. Simulerade professionella tester visade mätbara förbättringar i felupptäckt och riskprioritering, med en noggrannhet som ökade från 53,3% till 83,3% efter iterativ förfining av prompt och feedbackdriven hämtning.

Projektet adresserar utmaningar såsom att minska hallucinationer från LLM, säkerställa tillförlitligheten hos feedback och uppfylla krav för säkerhetskritiska applikationer. Det erbjuder en skalbar grund för semiautomatiserad validering av kemisk dokumentation i industriella miljöer.

Sammanfattningsvis visar detta att artificiell intelligens kan integreras för att förbättra datakvalitet och därmed minska den mänskliga arbetsbördan i miljöer som kräver både noggrannhet och snabbhet.

Nyckelord: Felupptäckt, automatisering, databas, prompt, feedback-loop, retrieval-augmented generation (RAG), Large Language Model (LLM).

Preface

The project was implemented in close collaboration between Alaa Mouselli and Faraj Hamedy. We worked together and contributed equally to all steps, including research, planning, data collection, development, testing, and documentation, all the way through to final delivery.

We would like to extend our sincere thanks to Hoffi Palmqvist of Kemikaliedokumentation AB, who contributed to the concept and provided important guidance regarding documentation, which enhanced our understanding of the project context.

We owe a great deal of appreciation to our supervisor, Mohammed Ghaith Altarabichi, for his continuous support throughout the project and his constructive guidance at every stage.

We would also like to thank our examiner, Mahmoud Rahat, who not only evaluated our work but also contributed valuable suggestions, corrections, and questions that helped improve our project.

Contents

Abstract . . . . . **ii**
Sammanfattning . . . . . **iii**
Preface . . . . . **iv**
Contents . . . . . **v**
List of Figures . . . . . **vii**
List of Tables . . . . . **viii**
Listings . . . . . **ix**
Acronyms . . . . . **x**
1 Introduction **1**
1.1 Background 1
1.2 Problem Statement 2
1.3 Purpose and Goals 2
1.4 Requirements 3
1.5 Limitations 5
1.6 Related Works 5
2 Technical Background **7**
2.1 Programming Language 7
2.2 Natural Language Processing (NLP) 7
2.3 Large Language Models (LLMs) 8
2.4 Retrieval-Augmented Generation (RAG) 8
2.5 Human-in-the-Loop (HITL) 9
2.6 AI Response Classification 10
2.7 Database and API Management 10
2.8 Version Control and Git 11
3 Method **12**
3.1 Development Process 12
3.2 Technical Design 15
3.3 Data Usage and Evaluation Split 17
4 Implementation **19**
4.1 System Overview 20
4.2 Tools and Libraries 22
4.3 Deployment 22
5 Results **23**
5.1 Functional Results 24

5.2 Prompting Strategy Evaluation 27
5.3 Results Summary 28
6 Discussion **32**
6.1 Societal Aspects 33
7 Conclusion **34**
7.1 Future Work 35
References **37**

List of Figures

1.1 Comprehensive Model Comparisons 4

3.1 Flowchart illustrating the full system pipeline 18

5.1 Viewing the Received Email with the Report 25
5.2 Recording Final Report Sending Time 25
5.3 Reviewing Automation Check Results and Suggestions 25
5.4 User's Feedback and Requested Fixes 26
5.5 Integrating User Feedback into Final Results 26
5.6 Mean Classification Accuracy per Prompting Strategy 28
5.7 Zero-shot Prompting: Accuracy Across Runs 29
5.8 Static 2-shot Prompting: Accuracy Across Runs 29
5.9 Static + RAG Prompting: Accuracy Across Runs 30
5.10 Combined Accuracy Stability Across Runs 30

List of Tables

4.1 Example of a product row from the Excel dataset 20

Listings

3.1 Prompt used to check allergy risk with LLM (zero-shot role-based prompt) 12
3.2 Example of static example used in the prompt 16
3.3 Example of static + RAG prompt 16

4.1 Building RAG-enhanced prompt for GPT-4 Turbo 20
4.2 Overriding AI output with feedback and supporting RAG 21

Acronyms

AI Artificial Intelligence. ix

API Application Programming Interface. ix

CSV Comma-Separated Values. ix

GPT Generative Pre-trained Transformer. ix

GUI Graphical User Interface. ix

HITL Human-In-The-Loop. ix

IMAP Internet Message Access Protocol. ix

LLM Large Language Model. ix

NLP Natural Language Processing. ix

NoSQL Not Only SQL. ix

QA Quality Assurance. ix

RAG Retrieval-Augmented Generation. ix

SMTP Simple Mail Transfer Protocol. ix

SQL Structured Query Language. ix

UX User Experience. ix

Chapter 1

Introduction

1.1 Background

Chemical products are widely used across various industries, which creates challenges for companies in ensuring that these substances are handled safely. Maintaining a safe working environment is essential, both for protecting employees and for complying with European regulations such as REACH (Registration, Evaluation, Authorisation, and Restriction of Chemicals) [1], which aims to improve the protection of human health and the environment from chemical risks. Another critical regulation is CLP, which governs the classification, labelling, and packaging of chemical substances [2].

Chemical documentation (Cdoc) is a company that supports compliance with chemical legislation by providing services in documentation, management, and risk assessment of chemical products used by other companies. Cdoc uses a relational database to store operational data, which it has chosen to keep private for security reasons.

A large amount of data is stored and processed in the database daily to conduct new risk analyses and update existing documentation. However, as the data volume grows, the company faces significant challenges such as duplicate entries, missing information, and inaccurate descriptions—largely due to its reliance on manual data validation.

As the primary source of chemical risk documentation for its clients in Sweden, Cdoc holds legal responsibility in the event of any chemical-related incident that results from inaccurate or misleading instructions—for example, incorrect mixing of substances at a client's facility.

In recent years, artificial intelligence (AI) has opened new opportunities to automate tasks such as data extraction, validation, and correction through its ability to interpret and process natural human language. Despite known limitations such

as hallucinations and occasional inconsistencies [3], AI holds significant potential in this domain.

This project explores how AI can optimize chemical risk documentation by reducing human effort, improving accuracy, and maintaining high standards of safety and regulatory compliance.

1.2 Problem Statement

The manual checking of chemical databases presents several pressing challenges:

- Risk information may be outdated, missing, or incorrectly classified.
- Continuous review processes are expensive and time-consuming.
- Managing large chemical databases is difficult without intelligent assistance.

The core problem is that managing manual documentation efforts is inefficient, insecure, and prone to failure due to its reliance on human intervention. This results in manual entry errors and undetected issues within the database.

An example of the data stored in the database is a detailed explanation of various chemicals and how each should be handled by companies that rely on Cdoc to ensure the safe management of their chemical products. These data sheets must be updated regularly, and due to Cdoc's legal responsibility for its clients' safety, it must ensure that all information is accurate and current.

To address these challenges, Cdoc aims to implement an artificial intelligence system that understands its database to meet safety requirements and automatically identify documentation errors with minimal oversight. The system should also be capable of receiving and storing human feedback.

The central research question is: *Can a system combining artificial intelligence and structured human feedback loops automate the validation of chemical risk documentation in a safe and reliable way?*

1.3 Purpose and Goals

The project aims to use artificial intelligence to reduce the time, effort, and cost required to detect errors in chemical product data. In doing so, it improves the quality and accuracy of current information while minimizing the risk of human error.

The goal of this project is to develop an effective system powered by a large language model (LLM), capable of identifying faults in large volumes of existing data. The system is designed to provide relevant suggestions and solutions for correct-

ing documentation issues through carefully constructed prompts.

This solution enhances the company's data management by ensuring improved information quality, enabling more efficient decision-making, and offering protection against potential legal liabilities.

The project addresses known challenges such as LLM hallucinations, feedback reliability, and safety-critical application requirements. It offers a scalable foundation for semi-automated validation of chemical documentation in industrial environments.

The main process is to design, implement, and evaluate an AI-powered system (LLM-based) that:

- Detects errors and missing information in the database automatically.
- Suggests corrections using knowledge derived from prompts based on test data.
- Receives and stores user feedback via email and applies it to the corresponding product only.

To evaluate the system, the following research questions are explored through testing:

- How accurately can an LLM detect errors in chemical risk documentation?
- How effective is a product-specific feedback memory in preventing repeated issues?
- How can prompt engineering reduce the risks of LLM hallucinations?

1.4 Requirements

- **Functional Requirements**
  The system should be capable of detecting errors in the database, such as duplicate data, missing fields, or inaccurate descriptions. For example, chlorine is a hazardous and frequently used chemical. Long-term exposure to this chemical may harm the lungs. The potential risk factor for using this chemical is four. This data is compared to the output of the LLM, which is prompted to classify the chemicals on a predefined risk scale.

  The AI tool must notify the company when an error is detected and provide a correction suggestion. The AI will receive feedback from the user on whether the suggestion is acceptable. Each time the AI receives negative feedback, it saves this feedback in a field linked to the specific product.

  The system's performance is evaluated by introducing basic and well-known faults into the database and checking whether the system can detect them.

The results are then compared to a version of the data that has been manually verified for accuracy.

- **Nonfunctional Requirements**
  The AI tool must take into account economic cost, efficiency, and data security. The system should minimize the margin for false negatives to ensure that no critical errors are overlooked.

As shown in Figure 1.1, Model like GPT 4 Turbo demonstrate powerful capabilities in handling various languages, including Swedish, by producing contextually natural texts and accurately understanding Swedish expressions. However,the LLaMA3 model appears to only partially support Swedish and requires fine-tuning to improve its performance. [4].

The model GPT-4 Turbo is excellent at learning from small amounts of data, providing the model with a single example to help it understand the task, which in turn gives a more appropriate and accurate answer based on the example [5].

Whereas the other four models are less effective and require large datasets. Those models may provide a suitable answer from a single sample, but it does not learn as thoroughly as typical machine learning techniques do [6, 7].

The ability to manage databases also varies between models. GPT-4 Turbo is the first model capable of creating and managing database queries [8]. Some models lack the ability to extract data directly from the web, unlike GPT-4 Turbo and Claude 3 Opus, which can immediately read raw HTML, extract structured data, and even analyze screenshots with vision skills. [9].

The first two models are cloud-based and operate via external servers, which gives them a greater ability to process data and generate results more efficiently. In contrast, models that run directly on the user's device require high-performance hardware to function effectively [10, 11]

Model	Swidish support	One-shot learning	Database management	Web scraping	Type
GPT-4 Turbo	Advanced	Advanced	Advanced	Advanced	Cloud-based
Claude 3 Opus	Llimited	Moderate	Advanced	Advanced	Cloud-based
Mistral 7B	Limited	Moderate	Moderate	Limited	Cloud-based &Local model
LLaMA 3	Partial	Moderate	Moderate	Limited	Cloud-based &Local model
Falcon 40B	Limited	Moderate	Moderate	Limited	Cloud-based &Local model

Figure 1.1: Comprehensive Model Comparisons

1.5 Limitations

Several boundaries were defined for the project:

- The AI system focuses only on risk priority and allergy risk checks, and does not generate full Safety Data Sheets (SDS).
- The human feedback mechanism is based on simple email replies; no complex user interface was developed.
- Testing was conducted on simulated and anonymized datasets due to privacy and data access limitations.
- The LLM's performance can improve through better prompting, but the prompts used in this project were constrained to the test data provided, not real company data.

These limitations were necessary to keep the project focused, achievable, and safe within the available time and resources.

1.6 Related Works

The study *Automated Detection of Data Quality Issues* [12] investigates how large language models (LLMs) are becoming invaluable tools for automating error detection. These models can analyze both structured and unstructured data to identify missing values, outliers, and other inconsistencies in large datasets.

Another recent development is the use of Retrieval-Augmented Generation (RAG), which allows language models to access external information during the generation process. In their study, Lewis et al. [13] introduced a RAG-based system that combines a document retriever with a language model to improve performance on knowledge-intensive tasks. Their approach showed that the system could better identify relevant facts and produce more accurate results compared to models that rely only on pre-trained knowledge. This method has been successfully applied in areas such as question answering, data validation, and knowledge-based search. The ability to bring in up-to-date and domain-specific information makes RAG a promising technique for improving error detection in chemical databases.

Various studies have demonstrated the successful application of AI in error detection. For example, research conducted by Zhang, Lin, and Wang [14] focused on developing an AI-based detection system. Their study showed that AI could significantly reduce false positives while maintaining high accuracy in identifying data errors.

In another relevant study, Lee, Kim, and Park [15] explored how AI could enhance chemical risk assessment. They developed a predictive model trained on historical chemical incident data to identify potential risks. Their system was designed to detect inconsistencies and issue timely alerts. The results demonstrated that this

AI-based approach outperformed traditional manual assessments in both accuracy and response time, while also reducing the potential for human error.

These and similar studies provide a valuable foundation for this project by showcasing effective strategies for error detection and AI-driven risk analysis. The insights gained from their techniques help guide the design of a system capable of identifying faults in a chemical database and reporting them to the responsible company.

Chapter 2

Technical Background

This chapter provides an overview of the main tools, techniques, and concepts used to build and evaluate the system.

2.1 Programming Language

Python was selected as the programming language for this project. It is widely used in the machine learning field due to its strong compatibility with AI tools and its broad collection of libraries that support data handling and automation [16, 17].

Several Python libraries were used throughout the project:

- **Pandas:** Used to load, process, and analyze Excel files containing chemical product data.
- **OpenAI Python SDK:** Enabled the integration of GPT-4 Turbo, allowing structured prompts to be sent and responses to be received.
- **smtplib and imaplib:** Supported secure, automated email communication, including the handling of attachments and replies.
- **JSON and CSV libraries:** Facilitated the storage and parsing of structured data and user feedback.

2.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of AI that allows machines to understand, interpret, and generate human language. It bridges the gap between human communication and machine understanding, enabling AI models to extract meaning from text, identify entities, and carry out language-related tasks such as summarization, classification, and translation [18].

2.3 Large Language Models (LLMs)

Large Language Models (LLMs) are pre-trained AI systems that use (NLP) section 2.2 techniques to understand and interact with human language [19]. These models can be accessed via cloud platforms or installed locally. In this project, LLMs were applied to analyze chemical product data and identify inconsistencies by interpreting text-based information.

Various LLMs are available on the market, including options from Anthropic, Cohere, Mistral, and AI21. Each has unique features and trade-offs. As shown in Figure 1.1, GPT-4 Turbo—used in this project—performed with greater speed and reliability compared to alternative models [20]. It also offers a competitive pricing structure, with 1000 tokens costing approximately 0.11 SEK. Although locally hosted models may reduce operational costs over time, many lack essential functionality for this use case [21].

A critical capability required for the system was the ability to search external sources (e.g., online safety documentation) to validate product information. GPT-4 Turbo supports this functionality, making it suitable for detecting inconsistencies in product descriptions.

The company database contains Swedish-language entries, which not all LLMs can interpret reliably. While some models can be fine-tuned to support additional languages, GPT-4 Turbo offers built-in support for Swedish, making it a practical choice for this environment.

Because only a limited test dataset was provided, the system did not involve traditional model training. Instead, zero-shot, one-shot, and few-shot learning strategies were used. These strategies guide the model using carefully constructed prompts rather than retraining the model with new data.

To further improve model accuracy, the system was also extended with a Retrieval-Augmented Generation (RAG) section 2.4 mechanism, where validated feedback examples are dynamically retrieved from a feedback log and inserted into the prompt.

2.4 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a technique that enhances the accuracy of large language models by providing them with relevant external context during inference. Instead of relying solely on the model's pre-trained knowledge (because LLMs have limited knowledge in specialized domains [22]), RAG allows the system to dynamically retrieve useful examples or documents that inform the

model's response [13].

In this project, a lightweight RAG mechanism was implemented using a feedback log that stores validated user corrections. When analyzing a new product, the system selects two feedback examples that are most similar to the current product based on matching keywords in the *Ingredients* field and similarities in the *Product Name* or *Product Category* when available.

These retrieved examples are then inserted into the model prompt as "Prior Approved Examples." This technique allows the model to generalize from real-world corrections, reduces hallucination, and helps maintain consistency across similar products.

Unlike model fine-tuning, which requires retraining on labeled data, RAG can be applied flexibly and incrementally. As the feedback log grows, the system continuously improves its performance without modifying the underlying GPT-4 Turbo model. This makes RAG a practical and scalable solution for specialized domains such as chemical documentation validation, while providing overfitting.

2.5 Human-in-the-Loop (HITL)

Human-in-the-Loop (HITL) is an approach in AI system design where human input is used to refine or correct machine-generated outputs. This method allows systems to benefit from human oversight, particularly in complex or sensitive tasks such as chemical documentation validation [23].

In this project, the HITL process was implemented using an email-based feedback mechanism. After receiving the AI-generated report, the user could submit corrections by email. The system stores this feedback and applies it during future analyses of the same product to prevent similar errors.

In addition to applying corrections for the same product, the validated feedback stored in the feedback log also serves as a retrieval source for the system's Retrieval-Augmented Generation (RAG) mechanism. During analysis of new products, the system dynamically selects relevant prior feedback examples and inserts them into the model prompt. This enables the HITL process to contribute to continuous system improvement, helping the model generalize better to new and unseen cases without requiring explicit model retraining.

2.6 AI Response Classification

To evaluate the AI's performance, response results were categorized according to a standard confusion matrix:

- **True Positive (TP):** A document *with an error* is correctly identified *as having an error*.
- **False Positive (FP):** A document *without an error* is incorrectly identified *as having an error*.
- **True Negative (TN):** A document *without an error* is correctly identified *as error-free*.
- **False Negative (FN):** A document *with an error* is incorrectly identified *as error-free* [24].

In early tests, false positives were common. To address this, a Human-in-the-Loop (HITL) feedback mechanism was implemented, enabling users to review the AI's decisions and submit corrections when needed.

An automated email system supports the feedback loop. After each analysis cycle, a report is sent to the user, who can reply with feedback. This feedback is saved and linked to the specific product, ensuring that the same mistake is not repeated in future checks for that product.

2.7 Database and API Management

Database interactions in this project were carried out using HeidiSQL, a lightweight SQL client that facilitates relational database operations. Key functionalities used include:

- **Querying and filtering structured data:** Helped isolate relevant records for analysis.
- **Schema visualization:** Provided a graphical representation of table structures and relationships.
- **Data consistency checks:** Ensured that input data was accurate and reliable before processing.

API testing and interaction were conducted using Postman. This tool supported:

- **Secure API requests:** Enabled safe interaction with the test database.
- **Automated data retrieval:** Simplified the extraction of structured records.
- **Error detection in API responses:** Helped identify anomalies in returned data.

Because direct access to the production database was restricted, all relevant test data was extracted via HeidiSQL and Postman, then saved in Excel format for further analysis.

2.8 Version Control and Git

Version control was essential for tracking development progress, safeguarding work, and organizing feedback. Git was used as the primary version control tool throughout the project. The codebase was hosted on GitHub, which provided remote storage, backup, and collaboration functionality [25].

The use of Git enabled:

- Safe management of incremental system updates.
- Creation of backup branches for testing features.
- Rollback to previous working versions when needed.
- Integration of company feedback during development stages.

Chapter 3

Method

3.1 Development Process

To initiate the project, an account was created on the OpenAI platform to work with the Chat 4-Turbo module. The platform offers advanced technology for data learning, along with several advantages for testing and evaluating the AI model, which enhance efficiency, accuracy, and performance speed.

The next step was to collect essential company data safely and securely, using HeidiSQL to connect to the company's test database. Postman was also utilized to interface with the company's API.

After data collection, the information was organized and standardized using Python libraries suitable for handling large amounts of data, such as Pandas. This step involved using standardized units (e.g., temperature or currency), consistent date formats, and uniform naming conventions for terms that may otherwise vary. This preparation helped ensure that the data was suitable for accurate analysis.

Once organized, the data was saved in an Excel file, allowing the LLM to interact with it via prompt-based input. In the initial version of the system, Listing 3.1 is a simple role-based prompt that was used to determine allergy risk. This role-based prompt is effectively a **zero-shot** prompt, as it does not include any specific examples; the model is guided only by the task description and role assignment:

```
System:
You are a chemical safety expert.
Your task is to determine whether a product poses any allergy risk.
Respond with:
Risk: 1 (risk) or 0 (no risk)
Type: respiratory, skin, both, or none
```

Listing 3.1: Prompt used to check allergy risk with LLM (zero-shot role-based prompt)

Prompts serve as the primary method for communicating with the LLM.

In the final version of the system, multiple prompt configurations were implemented and evaluated: **zero-shot**, **static two-shot**, and **static + Retrieval-Augmented Generation (RAG)**. The RAG mechanism dynamically inserts validated feedback examples into the prompt, enabling the LLM to generalize from real-world corrections. This flexible approach allows the system to adapt to evolving feedback without requiring model retraining while avoiding overfitting.

There are several prompt types used in AI systems:

- **Instructional prompts:** Direct commands or tasks given to the model.
  Example: "`Write a text about .....`"
- **Question-based prompts:** Requests for information expressed as questions.
  Example: "`What is the .....?`"
- **Role-based prompts:** Assigning the model a role and requesting it to act accordingly.
  Example: "`Act as a singer and write a new song about .......`"
- **Few-shot prompts:** Providing several examples to demonstrate the expected input and output.
  Example:
  "`Translate the following to French:`
  `Hello → Bonjour`
  `Thank you → Merci`
  `Goodbye →`"

The quality of LLM responses depends greatly on selecting the appropriate prompt type. Depending on the specific task or the data involved, the instructions can be refined to improve reliability [26].

One of the prompts used, Listing 3.1, is a role-based prompt and served as the baseline **zero-shot** configuration during system testing.

Prompts were refined multiple times during the development process to improve response accuracy and achieve higher-quality final results. Several specific prompt engineering techniques were employed:

- **Clarification of expected formats:** Instructions were added to explicitly define the required response format, such as: "`Respond as: Risk Priority (0-5), Allergy (0 or 1), Type (respiratory, skin, both, or none).`" This reduced ambiguity in the model's output.
- **Role-based prompts:** The model was consistently assigned the role of a chemical safety expert, providing it with a clear domain context and expected behavior.
- **Cross-referencing:** The model was instructed to consider ingredient information in relation to known risk classifications, encouraging it to base its reasoning on product content.
- **Static examples:** Representative examples of correctly classified products

were included in the prompt to demonstrate the desired input-output relationship (static two-shot prompting).

- **Dynamic feedback examples (RAG):** When using the static + RAG prompting configuration, two validated examples from the feedback log were dynamically retrieved and inserted into the prompt as "Prior Approved Examples." This enabled the model to generalize from prior human corrections.
- **Iterative phrasing refinement:** Prompt phrasing was iteratively tested and adjusted across development cycles to improve consistency, reduce hallucination, and minimize variation between runs.

These techniques collectively enhanced the reliability and accuracy of the AI system's outputs, allowing it to perform effectively in the specialized domain of chemical risk documentation.

GitHub was used to manage collaboration and track ongoing changes, while also preserving the full version history of the implementation. The project agency funded the software infrastructure, including cloud-based services that enabled efficient financial data management and operational support.

3.2 Technical Design

The system architecture consists of several interconnected modules responsible for data validation, AI-based evaluation, and communication with users. Initially, the product database is loaded from an Excel file and passed through a validation pipeline to detect missing values, inconsistent formats, and duplicate entries. These checks ensure data quality before AI evaluation begins. Although the system currently interacts with Excel files, it is designed to support future integration with the company's database.

Once validated, the data is processed by the AI analysis module, which connects to the GPT-4 Turbo model via the OpenAI API. Each product entry is interpreted individually using prompts. The LLM evaluates the product's risk priority and allergy classification, flagging any issues or inconsistencies. When necessary, the model generates a descriptive explanation and suggests a correction.

To include expert oversight, the system incorporates a feedback mechanism. User feedback is logged in a CSV file and is consulted before new reports are generated. If prior feedback exists for a product, it overrides the AI's judgment. If the latest feedback contradicts earlier entries, the system replaces the previous one to avoid conflicts and reduce file growth. If no previous input exists, the new feedback is added normally. This process ensures that AI suggestions reflect prior human input and allows the feedback log to serve as a continuously growing source of trusted domain knowledge.

An important enhancement introduced during development was the use of feedback examples as part of Retrieval-Augmented Generation (RAG). Rather than only applying overrides for individual products, the system also retrieves two relevant prior feedback examples and inserts them into the LLM prompt to guide classification of new products. This allows the system to generalize from validated human corrections and improve performance over time.
The project tested three prompting strategies:

- **Zero-shot prompting:** The role-based prompt without examples (Listing Listing 3.1).
- **Static two-shot prompting:** Prompt includes two fixed examples of classified products.
- **Static + RAG prompting:** Prompt includes two static examples plus two dynamically retrieved feedback examples.

For the static + RAG prompting configuration, the system retrieves exactly two feedback examples to balance relevance and generalization. Using a small number of examples ensures that the model does not overfit to specific prior cases or become biased toward frequent products in the feedback log. Including too many examples can also exceed the model's context window, reducing performance. By

selecting two of the most relevant validated examples, the system provides targeted guidance while maintaining flexibility for new, unseen products.
The selection mechanism is based on simple similarity matching. For each new product being analyzed, the system compares the *Ingredients* field to all validated entries in the feedback log, looking for the highest overlap of ingredient keywords. Additionally, it considers similarities in the *Product Name* and, when available, the *Product Category*. The two feedback examples with the highest combined similarity are selected and inserted into the prompt. This ensures that the guidance provided to the model is both relevant to the current product and based on real-world corrections.

Listing 3.2: Example of static example used in the prompt

```
Product ID: 00011560-5526-11e9-9082-af5c16251e75
Product Name: Hardtop Optima Comp A (ENG)
Ingredients: silane, trimethoxy[3-(oxiranyl-methoxy)propyl]- (CAS 2530-83-8),
xylene (CAS 1330-20-7), and stoddard solvent (CAS 8052-41-3)
Risk Priority: 3
Allergy: 1
```

The complete prompt structure for the static + RAG configuration is shown in Listing 3.3:

Listing 3.3: Example of static + RAG prompt

```
You are a chemical safety expert.
Your task is to classify the following product according to risk priority and
    allergy risk.
Risk Priority: 0 (none) to 5 (extreme)
Allergy: 0 (no risk) or 1 (risk)
Type: respiratory, skin, both, or none.

Static Examples:
Product ID: 00011560-5526-11e9-9082-af5c16251e75
Product Name: Hardtop Optima Comp A (ENG)
Ingredients: silane, trimethoxy[3-(oxiranyl-methoxy)propyl]- (CAS 2530-83-8),
xylene (CAS 1330-20-7), and stoddard solvent (CAS 8052-41-3)
Risk Priority: 3
Allergy: 1
----
Product ID: 00099999-aaaa-bbbb-cccc-ddddeeeeefff
Product Name: Sample Industrial Cleaner X
Ingredients: ethanol (CAS 64-17-5), ammonia solution (CAS 1336-21-6)
Risk Priority: 2
Allergy: 0
----

Prior Approved (Feedback) Examples:
Product ID: FEEDEX-12345
Product Name: Example Product A
Ingredients: propanol (CAS 71-23-8), water
Risk Priority: 2
Allergy: 0
----
Product ID: FEEDEX-67890
Product Name: Example Product B
Ingredients: sodium hypochlorite (CAS 7681-52-9), water
```

```
Risk Priority: 4
Allergy: 1
----

Now classify this product:
Product ID: CURRENT-PRODUCT-ID
Product Name: CURRENT PRODUCT NAME
Ingredients: CURRENT INGREDIENTS
Respond as:
Risk Priority: (0-5)
Allergy: (0 or 1)
Type: respiratory, skin, both, or none.
```

After analysis, the system generates a natural language report listing all flagged entries, describing the issues found, and including the AI's suggestions. This output enables quick identification of problems without the need to manually inspect large volumes of data.

The final component is a coordination module that orchestrates all system functions. It manages tasks such as data loading, validation, LLM interaction, feedback integration, report creation, and email communication. Utility functions support shared logic for formatting, parsing and control flow.

3.3 Data Usage and Evaluation Split

Due to the limited size of the available test dataset, the system was not trained in the traditional machine learning sense, but instead evaluated using different prompting strategies (zero-shot, static two-shot, and static + Retrieval-Augmented Generation (RAG)).

The used product dataset consisted of approximately 90 chemical products for testing. To avoid overfitting and provide more robust evaluation, the tests were structured into three independent runs per prompting strategy, with each run evaluating a different random subset of 10 products (without overlap between subsets). This produced a total of 30 test products per strategy.

Importantly, the feedback log used for RAG examples was kept separate from the current test set during each run. For each product being evaluated, feedback examples were retrieved only from prior validated feedback, ensuring that no example from the current test set was used in its own prompt. This avoided data leakage and allowed us to observe true generalization performance.

Since the system does not rely on model retraining, but on dynamic prompting, this evaluation design provided a practical method for tracking how prompt design and RAG integration affect performance without introducing overfitting.
A flowchart showing the system's internal interactions is provided in Figure 3.1. The system includes an email interface that automatically sends the report to the

relevant user. Upon receiving the message, the user can reply with feedback, which is then parsed, validated, and stored. This feedback is used both to correct future evaluations of the same product and as a retrieval source for the RAG prompting mechanism.

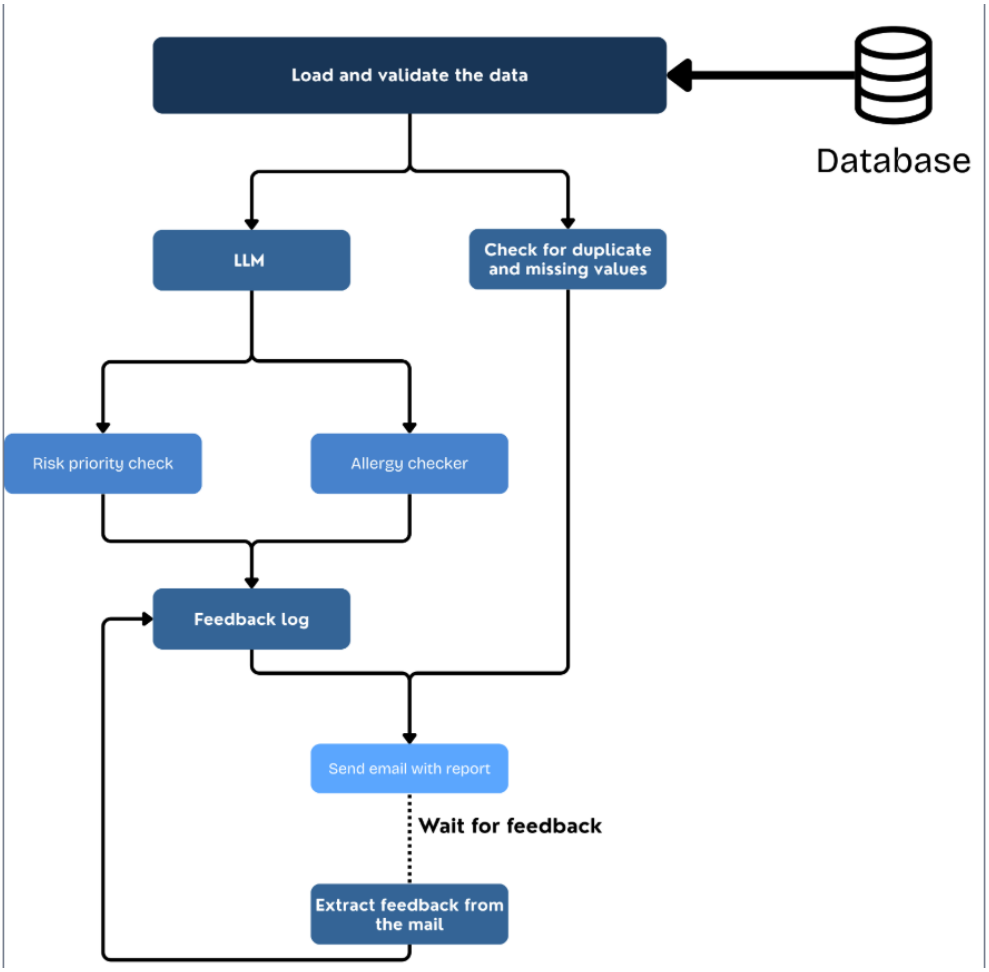

Figure 3.1: Flowchart illustrating the full system pipeline

Chapter 4

Implementation

This chapter describes the implementation of the AI-powered system for validating chemical product data. The implementation followed the architecture described in chapter 3, using Python libraries and external services such as the OpenAI API, HeidiSQL, and SMTP email servers. Each component was designed with maintainability, testability, and scalability in mind, following best practices from agile and modular software engineering.

Several technical motivations guided the chosen architecture:

- **Separation of Concerns:** Each module is designed with a single responsibility, enabling independent testing, modification, and debugging.
- **Extensibility:** New validation rules, feedback types, or AI models can be integrated without overhauling the system. For instance, replacing GPT with another model would only affect the `AI_Checker` module.
- **Feedback Loop Integration:** Centralizing feedback in a dedicated module (`FeedbackAnalyzer.py`) allows corrections to be decoupled from the original data and reused in future runs, promoting consistency and guiding the model through Retrieval-Augmented Generation (RAG).
- **Automation with Human Oversight:** The system automates routine analysis while retaining human judgment through email-based feedback. This hybrid approach mitigates hallucinations and increases trust.
- **Email-Based Interaction:** Email was chosen over GUI or CLI input to allow lightweight, distributed deployment in departments where email remains the dominant form of communication.
- **Prompts:** Due to limited access to the original data structure, the prompts were tailored to handle the test data. The code format allows prompts to be easily modified, and supports different prompting strategies (zero-shot, static two-shot, and static + RAG).

4.1 System Overview

The system was implemented in Python 3.11 and organized into multiple components:

- **Data Loader and Validator:** Reads chemical product data from an Excel spreadsheet and prepares it for the LLM. Each row represents a product and includes details such as:
  - Name
  - ID
  - Risk priority
  - Allergy risk
  - Supplier ID and other identifiers

  The number of products processed can be limited using the `config.py` module. Table 4.1 shows an example of the dataset structure.

ProductID	ProductName	Ingredients	RiskPriority	AllergyWarning
12345	Product Alfa	Ethanol, Ammonia	3	None

Table 4.1: Example of a product row from the Excel dataset

- **AI Checker:** Connects to the OpenAI API using an API key, builds prompts based on the selected prompting strategy, sends them to GPT-4 Turbo, and receives structured answers based on the model's evaluation.
  The system supports three prompting strategies:
  - **Zero-shot:** Role-based prompt without examples.
  - **Static two-shot:** Role-based prompt with two fixed static examples.
  - **Static + RAG:** Role-based prompt with two static examples and two dynamically retrieved feedback examples.

  For the Static + RAG configuration, the system compares the current product's *Ingredients*, *Product Name*, and (if available) *Product Category* to entries in the feedback log. The two most similar feedback examples are selected and inserted into the prompt.

  Using only two feedback examples was an intentional design choice to balance relevance and generalization. Including too many examples could lead the model to overfit on frequent products or biased corrections, or exceed the context window. The current design helps maintain flexibility for unseen products while providing targeted guidance.

```
prompt = f"""
You are a chemical safety expert.
Your task is to classify the following product according to risk priority and
     allergy risk.
Risk Priority: 0 (none) to 5 (extreme)
Allergy: 0 (no risk) or 1 (risk)
```

```
Type: respiratory, skin, both, or none.

Static Examples:
{static_example_1}
----
{static_example_2}
----

Prior Approved (Feedback) Examples:
{feedback_example_1}
----
{feedback_example_2}
----

Now classify this product:
Product ID: {product['ProductID']}
Product Name: {product['ProductName']}
Ingredients: {product['Ingredients']}
Respond as:
Risk Priority: (0-5)
Allergy: (0 or 1)
Type: respiratory, skin, both, or none.
"""
```

Listing 4.1: Building RAG-enhanced prompt for GPT-4 Turbo

The AI_Checker module can be configured to use any of these prompting strategies by setting the desired mode in `config.py`.

- **Feedback Analyzer:** Processes saved feedback and performs two key functions:
  - **Overrides:** If feedback exists for the exact product, it overrides the AI output (for consistency and user control).
  - **RAG Support:** The feedback log is also used as a retrieval source for RAG prompting. The module selects the two most similar feedback examples based on product similarity to guide the LLM when evaluating new products.

```
def apply_feedback_overrides(df, feedback_log):
    for _, row in feedback_log.iterrows():
        mask = df['ProductID'] == row['ProductID']
        if mask.any():
            df.loc[mask, 'RiskPriority'] = row['CorrectedRisk']
            df.loc[mask, 'Allergy'] = row['CorrectedAllergy']
    return df

def select_feedback_examples(current_product, feedback_log):
    # Example pseudo-code: rank by similarity on:
    # - Product Name (string similarity)
    # - Product Category (if available)
    ranked_examples = rank_feedback_similarity(current_product, feedback_log)
    return ranked_examples[:2]
```

Listing 4.2: Overriding AI output with feedback and supporting RAG

- **Email Sender and Receiver:** Sends reports via SMTP and retrieves feedback responses via IMAP.

    - **SMTP:** Formats and transmits messages using Python's `smtplib` [27].
    - **IMAP:** Fetches and reads user feedback using `imaplib` [28].
- **Issue Reporter:** Generates readable reports based on validation and LLM output.
- **Coordinator Script:** Manages the entire pipeline from data loading to report delivery and feedback processing.

4.2 Tools and Libraries

A mix of open-source and cloud-based tools was used:

- `pandas, openpyxl` – for reading and transforming Excel data [29]
- `openai` – to communicate with the GPT-4 Turbo API [30]
- `smtplib, imaplib, email` – for sending and reading emails
- `re, datetime, os, json` – for regex operations, time validation, file handling
- `HeidiSQL, Postman` – for initial database/API exploration
- `GitHub` – for version control and code collaboration

4.3 Deployment

The deployment steps were documented and delivered to the company as a manual, including installation instructions and required modifications.

1. Install dependencies using `pip install -r requirements.txt`.
2. Configure API keys and credentials in the `Config.py` file.
3. Update the Excel file path or switch to database loading as needed.
4. Set the sender and receiver email addresses in `Config.py`.
5. Run the coordinator script: `python main.py`.
6. Output reports will be saved to the `reports/` folder and emailed automatically.
7. Users reply to the same email with feedback including product name and correction.
8. A confirmation is sent when the system registers the feedback.

The company has begun testing the program, and it is currently undergoing validation.

Chapter 5

Results

This project implemented a data verification system for chemical products, based primarily on GPT-4 Turbo, using structured prompts and a Retrieval-Augmented Generation (RAG) mechanism. The system detects missing and incorrect data values by validating product fields and cross-referencing risk and allergy classifications against known domain knowledge and prior feedback. It also sends an email to the user indicating the location of errors and suggesting appropriate corrections, citing the source (either LLM output or user feedback). If the user approves the correction, it is saved in a separate feedback log.

These results address the main research question: “Can a system combining AI and structured human feedback loops automate the validation of chemical risk documentation safely and reliably?” Based on testing, the answer is positive. The system can automatically verify chemical product data and suggest corrections with increasing accuracy as prompt engineering techniques and feedback integration are applied. While hallucinations cannot be fully eliminated, their impact was substantially reduced through iterative prompt refinement and the use of RAG-based guidance.

The testing process began by preparing a `data.csv` reference file containing verified chemical product data that had been manually checked to ensure it was error-free.

A series of test datasets was then created. For each test run, a new set of 10 unseen chemical products was selected, each containing a mix of correct and known erroneous values. Typical errors included incorrect risk priority values, incorrect allergy classifications, missing fields, and inconsistencies in product descriptions. This process ensured that the LLM was evaluated on new, unseen data for each run.

The primary goal of the tests was to evaluate the impact of different prompt configurations on system accuracy and robustness, while also assessing the effect of

incorporating validated feedback through the RAG mechanism.

The system was tested across three prompting strategies:

- **Zero-shot prompting:** Role-based prompt without examples (baseline).
- **Static two-shot prompting:** Role-based prompt with two static examples.
- **Static + RAG prompting:** Role-based prompt with two static examples and two dynamically retrieved feedback examples.

For each strategy, three independent test runs were performed. Each run used a different dataset of 10 unseen products. In total, 90 unique products were evaluated (30 products per strategy). This structure ensured that the system's performance was not overfitted to any particular set of products and provided a more reliable assessment of generalization.

Importantly, for the RAG-based runs, the feedback examples were selected from prior validated feedback that did not overlap with the current test dataset. This avoided data leakage and ensured that improvements reflected true generalization.

5.1 Functional Results

In addition to classification accuracy, the system's functional requirements were verified during testing.

The system successfully identified duplicate entries, missing fields, and inconsistencies using built-in validation logic. LLM-based classification of risk priority and allergy risk was integrated seamlessly into the report generation and feedback loop.

Figures 5.1 and 5.2 illustrate the correct timestamp recording and email-based feedback mechanism. The system processed only feedback received after the report was sent.

Figure 5.3 shows the report attachment summarizing detected errors and AI-generated correction suggestions. For example, the product "STATOIL ANTIFOAM 1en" was flagged for potential skin allergy, contradicting the original entry.

Based on user feedback (Figure 5.4), corrections were applied, such as adjusting the risk priority for "TRAFFIC MAKER Extra Vit." The final file structure was preserved, as shown in Figure 5.5.

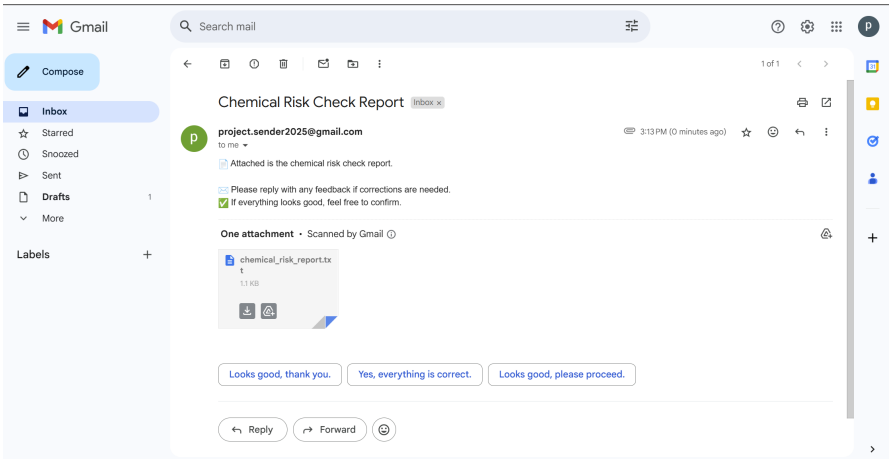

Figure 5.1: Viewing the Received Email with the Report

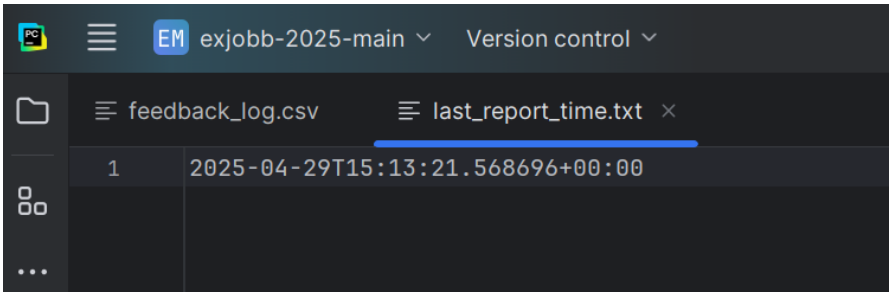

Figure 5.2: Recording Final Report Sending Time

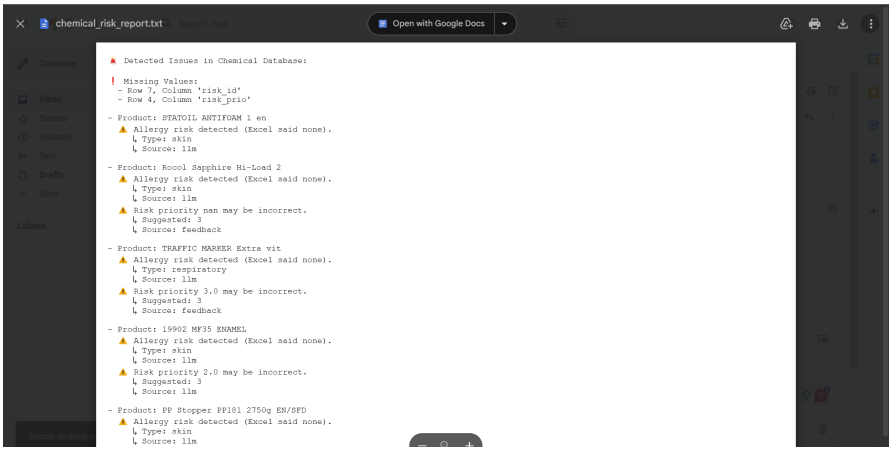

Figure 5.3: Reviewing Automation Check Results and Suggestions

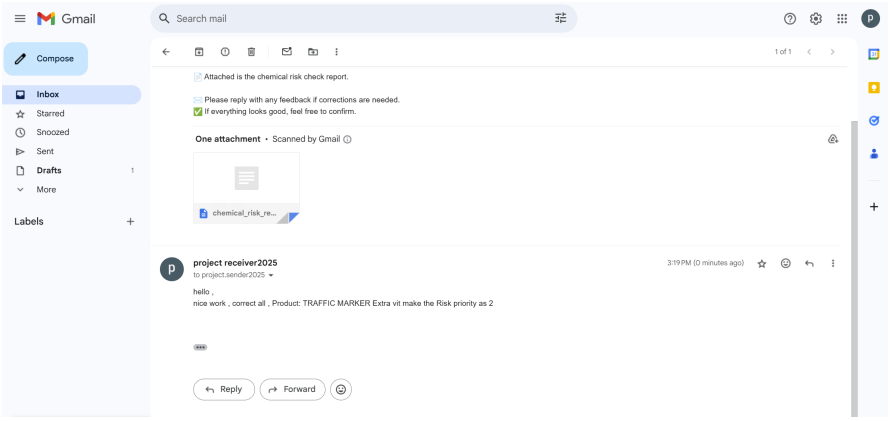

Figure 5.4: User's Feedback and Requested Fixes

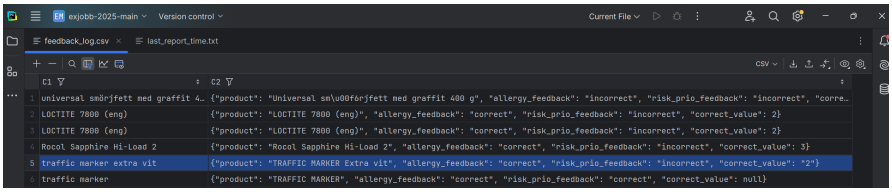

Figure 5.5: Integrating User Feedback into Final Results

5.2 Prompting Strategy Evaluation

To evaluate the impact of prompt design on system performance, a structured series of tests was conducted. The purpose was to measure how different prompting strategies affected the accuracy and stability of the AI model when classifying unseen chemical products.

The test results were analyzed both per individual run and as an average (mean) across three independent runs for each strategy. Calculating the mean accuracy provides a more robust estimate of overall performance by reducing the influence of random variation in any single test run.

Since Large Language Models (LLMs) are known to hallucinate [31], these tests also aimed to measure how much prompt design and feedback integration could reduce hallucination effects and improve classification accuracy.

The following metrics were calculated for each test run:

- **Risk Priority Accuracy (%):** Percentage of products for which the AI correctly classified the risk priority (matching ground truth).
- **Allergy Accuracy (%):** Percentage of products for which the AI correctly classified allergy presence and type (matching ground truth).

Test Structure:

- **Prompting Strategies Tested:**
  - Zero-shot (role-based prompt without examples)
  - Static two-shot (prompt with two static examples)
  - Static + RAG (prompt with two static examples + two dynamically retrieved feedback examples)
- **Runs per Strategy:** 3 independent test runs per strategy
- **Products per Run:** 10 unseen chemical products per run
- **Total Products Tested:** 90 products (30 per strategy)

Important Notes:

- All test products were unseen during prior development and not present in the feedback log.
- For Static + RAG runs, feedback examples were selected dynamically based on ingredient similarity and product name/category, using validated prior feedback that did not overlap with the current test set.
- This test design minimizes overfitting and ensures that observed improvements reflect true generalization, not memorization.

5.3 Results Summary

Mean Classification Accuracy per Prompting Strategy

Figure 5.6 shows Mean Classification Accuracy per Prompting Strategy. The left bar in each group shows the mean Risk Priority accuracy, while the right bar shows the mean Allergy accuracy. Results are averaged over three independent test runs, each on 10 unseen chemical products. Higher accuracy reflects better classification performance.

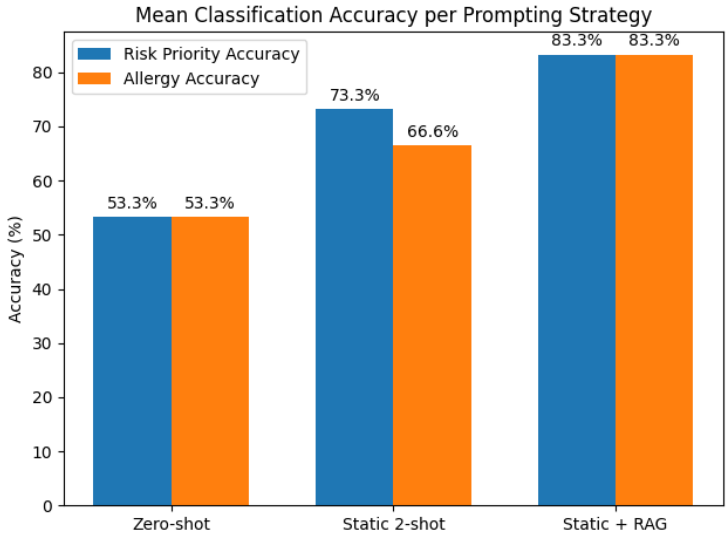

Figure 5.6: Mean Classification Accuracy per Prompting Strategy

Accuracy Stability Across Runs — Zero-shot Prompting

Figure 5.7 shows Zero-shot Prompting Accuracy Across Runs. This chart shows the accuracy variation across three test runs for Zero-shot prompting. Blue line: Risk Priority accuracy. Orange line: Allergy accuracy. This allows assessment of performance stability when using Zero-shot prompts.

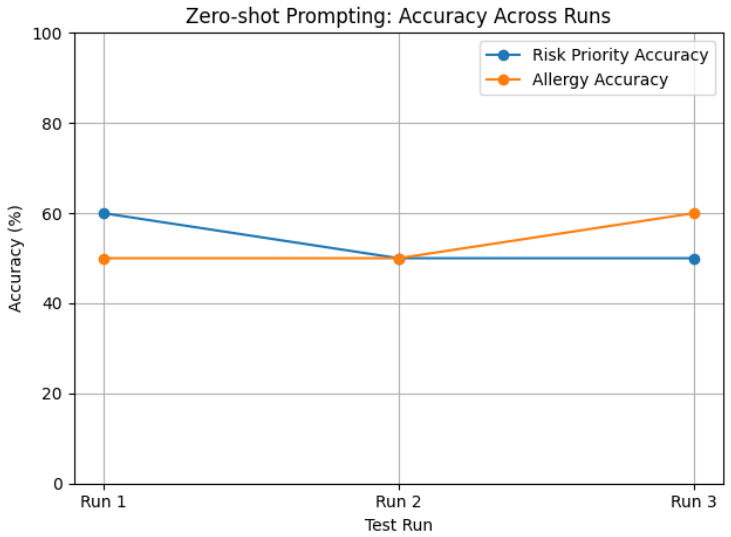

Figure 5.7: Zero-shot Prompting: Accuracy Across Runs

Accuracy Stability Across Runs — Static 2-shot Prompting
Figure 5.8 shows Static 2-shot Prompting Accuracy Across Runs. This chart shows the accuracy variation across three test runs for Static 2-shot prompting. Blue line: Risk Priority accuracy. Orange line: Allergy accuracy. The chart demonstrates performance stability and improvements compared to Zero-shot.

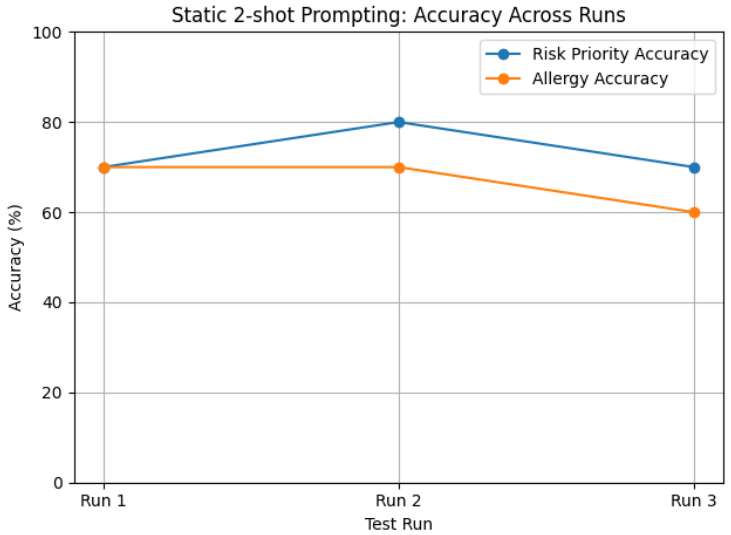

Figure 5.8: Static 2-shot Prompting: Accuracy Across Runs

Accuracy Stability Across Runs — Static + RAG Prompting
Figure 5.9 Static + RAG Prompting: Accuracy Across Runs. This chart shows the accuracy variation across three test runs for Static + RAG prompting. Blue line:

Risk Priority accuracy. Orange line: Allergy accuracy. The stability observed here reflects the generalization benefits of using prior validated feedback.

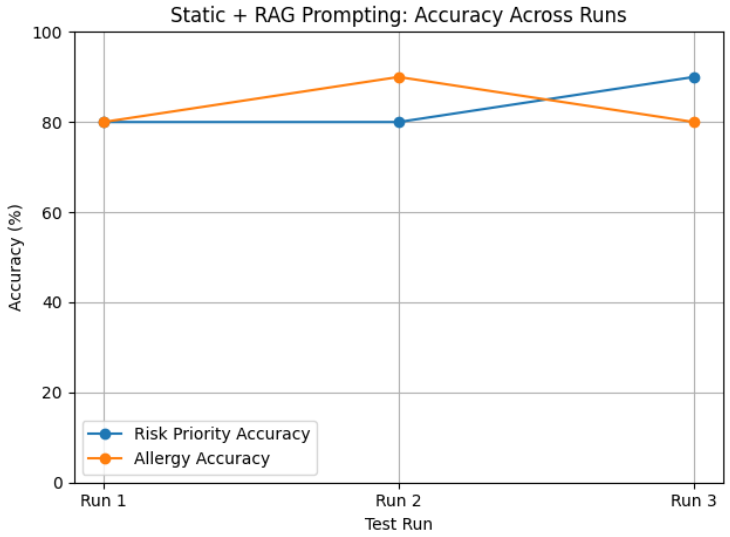

Figure 5.9: Static + RAG Prompting: Accuracy Across Runs

Combined Accuracy Stability Across Runs — All Strategies
Figure 5.10 Combined Accuracy Stability Across Runs. This chart compares the accuracy stability of all three prompting strategies across test runs. Blue line: Risk Priority accuracy. Orange line: Allergy accuracy. The chart highlights the impact of prompting technique on classification accuracy and stability when applied to unseen data.

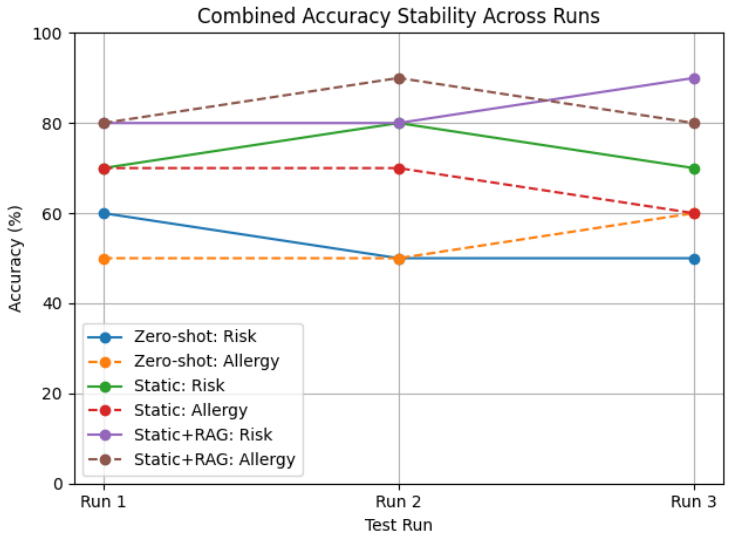

Figure 5.10: Combined Accuracy Stability Across Runs

Preliminary Observations:

- Zero-shot prompting produced baseline accuracy for both risk priority and allergy classification.
- Static two-shot prompting improved accuracy, especially in more complex products with diverse ingredients.
- Static + RAG prompting achieved the highest accuracy and demonstrated the most stable performance across runs.
- RAG integration provided improved generalization by incorporating validated domain corrections from prior human feedback.
- Hallucinations were not fully eliminated but were significantly reduced as prompt quality and feedback guidance improved.

All submitted feedback during testing was correctly registered in the feedback log. The system successfully applied this feedback both for direct overrides and as guidance in future RAG-enhanced prompts.

Chapter 6

Discussion

The results demonstrate that the system can identify errors in the database on a considerable scale from the first run. However, it is fully committed to user feedback, integrating it into each subsequent cycle to refine future results and improve classification consistency.

One example of hallucination was a false positive case in which the LLM confidently stated that a product did not have an allergy risk, even though this was incorrect. These hallucinations were successfully addressed through the human-in-the-loop feedback mechanism and further mitigated by improving the prompts and integrating relevant feedback examples through Retrieval-Augmented Generation (RAG).

The use of different prompting strategies like (Zero-shot, Static two-shot, and Static + RAG) helped highlight how much prompt design impacts system accuracy and stability. The introduction of RAG, in particular, demonstrated that generalization to new, unseen products can be improved when the model is guided with validated prior corrections rather than relying solely on its internal knowledge. This reinforces the role of human feedback in shaping the system's behavior beyond basic overrides.

The system could achieve even better results and further reduce hallucinations if more detailed prompts and a larger, more diverse dataset were available. However, one limitation was the restricted access to real company data and the reliance on a small, anonymized test dataset provided by the company. The current evaluation used entirely unseen products to avoid overfitting, and future work with a larger dataset would help further validate the observed improvements.

It is important to note that no fine-tuning of the LLM itself was performed in this project. The focus was on prompt engineering and feedback-driven generalization via RAG. Fine-tuning would require access to large, domain-specific datasets and specialized infrastructure, beyond the project scope. Still, the results show that

significant improvements can be achieved without model retraining, using carefully crafted prompts and human-in-the-loop guidance.

Finally, regarding the concern about the role of AI versus a simple lookup table, while exact feedback matches are directly applied, the RAG-enhanced prompts enable the model to generalize patterns from past corrections and apply them to new, unseen products with similar characteristics. This moves the system beyond simple memorization and allows it to assist users in cases where no prior feedback exists for the current product.

6.1 Societal Aspects

Artificial intelligence has become a vital tool in enhancing the quality of life. Its use is no longer confined to a single field but has expanded into nearly every aspect of daily life [32]. When discussing AI, both its benefits and potential risks must be acknowledged. Society should harness AI's strengths to serve humanity, foster development, and uphold justice and equality.

The current project relies on AI technology—specifically GPT-4 Turbo—as an innovative solution to automate the verification of chemical data accuracy and suggest appropriate corrections when inconsistencies are found.

This project enhances several critical areas. Most notably, it improves data reliability and aligns company practices with European legislation, such as REACH and CLP, requiring transparent chemical information to protect human health and the environment. Reducing the time and resources needed for error detection also contributes to sustainability goals.

The system respects privacy by not requiring direct access to company data. Organizations like CDOC, which manage sensitive chemical documentation, can use anonymized or manually uploaded datasets. This preserves data privacy and reduces the risk of unauthorized information disclosure.

Importantly, the project reinforces ethical principles by keeping a human in the decision-making loop. The AI system identifies and suggests corrections, but users validate the findings, approve suggested changes, or provide manual feedback.

The solution is adaptable to other industries that manage large-scale data, including healthcare, finance, and regulatory compliance. The system promotes transparency and accountability by identifying potential risks and proposing actionable improvements. Therefore, the project demonstrates that AI cannot replace human judgment but can collaborate, creating a balanced, ethical, and socially responsible workflow.

Chapter 7

Conclusion

In conclusion, this project demonstrates an innovative solution for integrating large language models into the process of improving chemical hazard documentation through validation, evaluation, and corrective suggestions. This, in turn, enhances data accuracy and reliability while reducing the cost and time required for error detection. The task was addressed by developing an AI-powered tool based on the GPT-4 Turbo model.

This tool successfully detected errors and duplicates across large volumes of chemical hazard data. NLP-based code and techniques were developed to understand, analyze, and verify each row in the company's dataset. This involved ensuring correct spelling, detecting missing values, eliminating duplicate IDs, and flagging inaccurate data. Prompt engineering guided the model to provide accurate corrections based on recognized standards. The use of Retrieval-Augmented Generation (RAG) further improved classification accuracy by allowing the model to generalize from validated human feedback.

A reporting system was also built to send automated reports via email. Each report was sent to the administrator using the SMTP protocol and included detailed error findings and correction suggestions.

The administrator could either approve the AI's suggestion or manually correct the errors. IMAP was used to receive user feedback, enabling fast and direct interaction between the user and the system. In the final stage, the corrected data was verified and saved in a separate file, preserving the original dataset.

The project fulfilled all specified requirements and demonstrated the effectiveness of GPT-4 Turbo in automating error detection, classifying problems by row, column, or product name, and suggesting corrections based on European chemical safety standards. Users can review and approve suggested corrections before any modifications are made.

It is important to note that no fine-tuning of the LLM itself was performed in this project. Instead, improvements were achieved through careful prompt engineering, integration of human feedback, and dynamic use of prior corrections through RAG. This approach avoids overfitting to the small test dataset and ensures that the system remains flexible and adaptable to new data.

This is a practical step toward future systems that leverage AI for both human benefit and sustainability. It is impactful on multiple levels, particularly in reducing human error caused by stress, which in turn decreases the likelihood of accidents that may harm workers or the environment.

By staying aligned with current information and improving compliance with global safety standards, the company can enhance its professional credibility with customers and regulatory authorities.

One of the most important outcomes of this project is that it does not replace human input. Instead, it promotes collaboration between high-speed AI technologies and human judgment by integrating AI with human-in-the-loop decision-making.

7.1 Future Work

The current system could be enhanced by integrating or replacing parts of it with Natural Language Understanding (NLU), a specialized subset of NLP. NLU could improve the system's ability to interpret user feedback by detecting intent, resolving ambiguities, and understanding the full context of a message.

While NLP typically segments text and performs grammatical analysis, NLU adds semantic interpretation—connecting the meaning of phrases and extracting deeper insights. NLU also offers better multilingual support and scalability to broader markets and industries [33].

While the current system is effective in identifying problems after the data is uploaded, sending reports, and storing corrections, it could be further improved by integrating directly into the data entry interface. This would allow real-time validation, on-the-fly corrections, and immediate documentation of changes, their reasoning, and trusted sources—all within the same workflow.

Additionally, future versions could allow the system to directly apply verified corrections to the database itself, rather than limiting it to report generation and manual follow-up. Another potential direction would be to explore fine-tuning the LLM on a larger domain-specific dataset to further enhance its specialized knowledge, provided that sufficient high-quality training data becomes available.

References

[1] European Chemicals Agency, *REACH*, https://echa.europa.eu/regulations/reach/understanding-reach, Accessed: 2025.

[2] European Chemicals Agency, *Understanding clp*, https://echa.europa.eu/regulations/clp/understanding-clp, Accessed: 2025.

[3] *Automatically correcting large language models*, https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00660/120911/Automatically-Correcting-Large-Language-Models, Accessed: 2025.

[4] M. Marcińczuk, Ł. Dadas, and P. Przybyła, *Evaluation of large language models for the automatic extraction of chemicals and safety information from safety data sheets*, https://arxiv.org/abs/2501.09154, Accessed: 2025.

[5] Z. Liu, M. T. Darestani, and M. Warschauer, *Evaluating large language models on reading and language assessments*, https://arxiv.org/abs/2501.06658, Accessed: 2025.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, and et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2025, Accessed: 2025. [Online]. Available: https://arxiv.org/pdf/2005.14165.

[7] H. W. Park, R. Dueñas, A. F. T. Martins, and A. Lavie, "Few-shot learning significantly improves large language models for low-resource cross-lingual summarization," in *Proceedings of the 4th Workshop on Low-Resource Machine Translation (LoResMT)*, Accessed: 2025, 2024. [Online]. Available: https://aclanthology.org/2024.loresmt-1.6.pdf.

[8] OpenAI and contributors, *Gpt-4 technical report*, https://cdn.openai.com/papers/gpt-4.pdf, Accessed: 2025.

[9] H. Ramadhan, *Web scraping experiment with ai (parsing html with gpt-4 and gpt-4o)*, https://serpapi.com/blog/web-scraping-and-parsing-experiment-with-ai-openai/, Accessed: 2025.

[10] Microsoft, *Deployment types for azure ai foundry models*, https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/deployment-types, Accessed: 2025.

[11] Microsoft, *How to deploy models locally with azure machine learning*, https://docs.azure.cn/en-us/machine-learning/how-to-deploy-local?view=azureml-api-2, Accessed: 2025.

[12] T. Pärnamaa, M. Sajjadi, H. Larochelle, S. Hooker, J. Pineau, J. Uszkoreit, P. Abbeel, B. Kim, and O. Russakovsky, "Datasets and benchmarks," in *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, Accessed: 2025, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/83fc8fab1710363050bbd1d4b8cc0021-Paper-Datasets_and_Benchmarks.pdf.

[13] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *arXiv preprint arXiv:2005.11401*, 2020, Accessed: 2025. [Online]. Available: https://arxiv.org/pdf/2005.11401.

[14] Z. Zhang and S. Lin, "Analysis of the effectiveness and countermeasures of high-quality economic development in the pearl river delta city cluster," *American Journal of Industrial and Business Management*, vol. 12, pp. 984–994, 2022. DOI: 10.4236/ajibm.2022.125050. [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=117391.

[15] S. H. Lee, J. Kim, J. Kim, J. Park, S. Park, and K.-B. Kim, "Current trends in read-across applications for chemical risk assessments and chemical registrations in the republic of korea," *Journal of Toxicology and Environmental Health, Part B: Critical Reviews*, vol. 25, no. 8, pp. 393–404, 2022. DOI: 10.1080/10937404.2022.2133033.

[16] N. Schell and A. Gelman, "Language models and statistics: A symbiotic relationship," *Technometrics*, vol. 66, no. 1, pp. 1–12, 2024. DOI: 10.1080/00401706.2023.2237826. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00401706.2023.2237826.

[17] *Leveraging python in ai and machine learning: A survey of techniques and educational approaches in software engineering*, https://www.researchgate.net/publication/388852036_Leveraging_Python_in_AI_and_Machine_Learning_A_Survey_of_Techniques_and_Educational_Approaches_in_Software_Engineering, 2025.

[18] A. Rayhan, R. Kinzler, and R. Rayhan, *Natural language processing: Transforming how machines understand human language*, Aug. 2025. DOI: 10.13140/RG.2.2.34900.99200.

[19] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020, Accessed: 2025. [Online]. Available: https://arxiv.org/pdf/2111.01243.

[20] A. Kakar, P. Arora, F. Song, S. Sengupta, C. Yuan, L. Cheng, H. Liu, and J. J. Liu, "Extracting structured information from unstructured histopathology reports using generative pre-trained transformer 4 (gpt-4)," *The Journal of Pathology*, vol. 256, no. 2, pp. 145–155, 2025, Accessed: 2025. DOI: 10.1002/path.6232. [Online]. Available: https://doi.org/10.1002/path.6232.

[21] S. Sengupta, S. Bhattacharya, S. Das, and M. Roy, *Comparative evaluation of gpt-3.5 and gpt-4 in generating rationales for multiple-choice medical questions*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10918540/, Accessed: 2025.

[22] R. Bommasani, D. A. Hudson, and et al., "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021, Accessed: 2025. [Online]. Available: https://arxiv.org/abs/2108.07258.

[23] A. Holzinger, "Interactive machine learning for health informatics: When do we need the human-in-the-loop?" *Brain Informatics*, vol. 3, pp. 119–131, 2016. DOI: 10.1007/s40708-016-0042-6.

[24] Google Developers, *Thresholds and the confusion matrix | machine learning | google for developers*, https://developers.google.com/machine-learning/crash-course/classification/thresholding, Accessed: 2025.

[25] S. Chacon and B. Straub, *Pro git*, https://git-scm.com/book/en/v2, Accessed: 2025.

[26] *An active inference strategy for prompting reliable responses from large language models in medical practice | npj digital medicine*, https://www.nature.com/articles/s41746-025-01516-2, 2025.

[27] Cloudflare, *What is smtp?* https://www.cloudflare.com/learning/email-security/what-is-smtp/, Accessed: 2025.

[28] TechTarget, *What is imap (internet message access protocol)?* https://www.techtarget.com/whatis/definition/IMAP-Internet-Message-Access-Protocol, Accessed: 2025.

[29] DataCamp, *Python pandas tutorial: The ultimate guide for beginners*, https://www.datacamp.com/tutorial/pandas, Accessed: 2025.

[30] IEEE Xplore, *Openai ieee paper*, https://ieeexplore.ieee.org/abstract/document/10963975, Accessed: 2025.

[31] "Measuring and reducing llm hallucination without gold-standard answers," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2402.10412.

[32] "Artificial intelligence revolution," *Nature Communications*, 2019. DOI: 10.1038/s41467-019-14108-y.

[33] TechTarget, *Natural language understanding (nlu)*, https://www.techtarget.com/searchenterpriseai/definition/natural-language-understanding-NLU, Accessed: 2025.