Information Extraction from Safety Data Sheets

Master's degree in Data Science

João Pedro Calado Dinis

Leiria, April of 2026

Information Extraction from Safety Data Sheets

Master's degree in Data Science

João Pedro Calado Dinis

Project Report under the supervision of Professor Carlos Fernando de Almeida Grilo, Professor Rolando Lúcio Germano Miragaia, Professor José Carlos Bregieiro Ribeiro and Professor António Manuel de Jesus Pereira.

Leiria, April of 2026

Originality and Copyright

This project report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Acknowledgments

I would like to express my sincere gratitude to Professors Carlos Grilo, Rolando Miragaia, José Ribeiro, and António Pereira for their guidance and support throughout the development of this project. Their insights and feedback were invaluable in shaping this work.

I am also grateful to VLM Compliance Solutions for providing the original dataset used in this project. The availability of real, annotated data was essential for the development and evaluation of the proposed approaches.

I would like to thank the friends and colleagues I met during the Master's program, in particular Francisco Teixeira, Pedro Costa, and Telmo Gregório, for their support, collaboration, and companionship throughout this journey. Their presence made this experience both more enjoyable and more productive.

Finally, I would like to express my deepest gratitude to my family for their constant support, encouragement, and understanding along the way.

To all of you, thank you.

Abstract

Safety Data Sheets (SDS) are essential documents for ensuring the safe handling, storage, and transportation of chemical products. Despite their standardized structure, SDS documents are often distributed as semi-structured PDF files, making the automatic extraction of key information a challenging task. This work addresses the problem of information extraction from SDS documents, with the objective of developing and evaluating different Natural Language Processing (NLP) techniques for this purpose.

A complete data processing pipeline was developed to transform SDS PDF files into a structured, model-ready dataset. This pipeline includes document collection, text extraction, section segmentation based on the standardized 16-section SDS format, and the construction of a question-answering dataset linking document context with target fields. Multiple extraction strategies were explored, ranging from rule-based methods using regular expressions to machine learning approaches based on Bidirectional encoder representations from transformers (BERT), and more recent approaches based on Large Language Models (LLM).

The evaluation focused on a set of representative fields, including binary classification, single-label classification, and multi-label classification tasks. The results show that rule-based approaches perform well for highly structured patterns, such as hazard codes, while BERT-based models achieve strong performance in well-defined classification tasks, reaching an F1-score of 1.0 for binary classification and up to 0.86 for more complex classification scenarios. LLM-based approaches demonstrate competitive performance across all tasks, while offering greater flexibility and reducing the need for task-specific engineering, particularly in more complex cases.

The results indicate that while older methods remain effective for specific scenarios, LLMs provide a more generalizable solution for information extraction from semi-structured documents. This work contributes a reproducible pipeline for SDS processing, a structured dataset for evaluation, and a comparative analysis of extraction methodologies, providing a foundation for future research and practical applications in automated document understanding.

Keywords: *Safety Data Sheets, Information extraction, Natural Language Processing, Transformer Models, Document Processing*

Resumo

As Fichas de Dados de Segurança (FDS) são documentos essenciais para garantir o manuseamento, armazenamento e transporte seguros de produtos químicos. Apesar da sua estrutura padronizada, as FDS são frequentemente distribuídas sob a forma de ficheiros PDF semiestruturados, o que torna a extração automática de informação relevante uma tarefa desafiante. Este trabalho aborda o problema da extração de informação a partir de FDS, com o objetivo de desenvolver e avaliar diferentes técnicas de Processamento de Linguagem Natural (PLN) para este fim.

Foi desenvolvido um processo completo de processamento de dados para transformar ficheiros PDF de FDS num conjunto de dados estruturado e pronto para utilização em modelos. Este processo inclui a recolha de documentos, a extração de texto, a segmentação em secções com base no formato padronizado de 16 secções das FDS, e a construção de um conjunto de dados de perguntas e respostas que relaciona o contexto do documento com os campos de interesse. Foram exploradas múltiplas estratégias de extração, desde métodos baseados em regras utilizando expressões regulares, até abordagens de aprendizagem automática baseadas em BERT, bem como abordagens mais recentes baseadas em Modelos de Linguagem de Grande Escala (LLM).

A avaliação centrou-se num conjunto de campos representativos, incluindo tarefas de classificação binária, *single-label* e *multi-label*. Os resultados mostram que as abordagens baseadas em regras apresentam um bom desempenho em padrões altamente estruturados, como códigos de perigo, enquanto os modelos baseados em BERT alcançam um desempenho sólido em tarefas de classificação bem definidas, atingindo um F1-score de 1,0 para a classificação binária e até 0,86 para cenários de classificação mais complexos. As abordagens baseadas em LLM demonstram um desempenho competitivo em todas as tarefas, enquanto oferecem maior flexibilidade e reduzem a necessidade de esforço aplicado para cada tarefa, particularmente em casos mais complexos.

Os resultados indicam que, embora os métodos mais antigos continuem a ser eficazes em cenários específicos, as LLMs oferecem uma solução mais generalizável para a extração de informação a partir de documentos semiestruturados. Este trabalho contribui com um processo de trabalho reproduzível para o processamento de FDS, um conjunto de dados estruturado para avaliação e uma análise comparativa dos métodos de extração, proporcionando uma base para futuras investigações e aplicações práticas na compreensão automatizada de documentos.

Palavras-chave: *Fichas de dados de segurança*, *Extração de informação*, *Processamento de linguagem natural*, *Modelos Transformer*, *Processamento de documentos*

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations and Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CAGR	Compound Annual Growth Rate
CAS	Chemical Abstracts Service
CLP	Classification, Labelling and Packaging
CNER	Chemical Named Entity Recognition
CNN	Convolutional Neural Networks
EU	European Union
FDS	*Ficha de Dados de Segurança*
GHS	Globally Harmonized System of Classification and Labelling of Chemicals
GPT	Generative Pre-trained Transformer
HCS	Hazard Communication Standard
HMM	Hidden Markov Models
JSON	JavaScript Object Notation
LLaMA	Large Language Model Meta AI
LLM	Large Language Model
LSTM	Long Short-Term Memory Networks
MLB	MultiLabelBinarizer
MLM	Masked Language Modeling
MLP	Multilayer Perceptron
NER	Named Entity Recognition
NLP	Natural Language Processing
NMT	Neural Machine Translation
NSP	Next Sentence Prediction
OCR	Optical Character Recognition
OSHA	Occupational Safety and Health Administration
PDF	Portable Document Format

PPE	Personal Protective Equipment
RNN	Recurrent Neural Networks
SDS	Safety Data Sheet
SMT	Statistical Machine Translation
TF-IDF	Term Frequency–Inverse Document Frequency

1. Introduction

The rapid growth of digital information in recent years has significantly increased the availability of unstructured textual data across multiple domains. Extracting meaningful information from such data has become a central challenge in Data Science, particularly in contexts where documents have partial structure but also high variability in content and format. Natural Language Processing (NLP) has emerged as a key discipline to address this challenge, enabling the automatic identification and interpretation of relevant information within textual data [1].

In the domain of industrial safety and chemical regulation, Safety Data Sheets (SDS) represent a critical source of information [2]. These documents contain essential details regarding chemical substances, including hazard classifications, safety procedures, and regulatory requirements. Although SDS follow a standardized 16-section structure defined by international frameworks, in practice they show significant variability in formatting, language, and level of detail depending on the manufacturer and region. The data used in this work was provided by *VLM Compliance Solutions*, which operates within this domain and manages a large collection of SDS documents.

From a data processing perspective, SDS documents can be characterized as semi-structured, combining free text, data tables, and domain-specific terminology. This hybrid nature introduces challenges for automated analysis, as relevant information may appear in different forms and locations across documents. Consequently, identifying and extracting specific attributes, such as hazard indicators, signal words, or classification labels, requires methods capable of handling both linguistic variability and structural inconsistency.

The high volume of SDS documents further emphasizes the need for efficient automated approaches. In many practical scenarios, the primary challenge is not only accessing the documents, but also accurately extracting and classifying key pieces of information that can support downstream applications such as compliance verification or risk analysis.

In this context, the problem addressed in this work is the automation of information extraction and classification from Safety Data Sheets using Natural Language Processing techniques. This work focuses on the identification and classification of specific target attributes, exploring how different approaches, including rule-based methods, transformer-based models, and modern Large Language Models, can be applied and compared in terms of effectiveness and reliability.

1.1. Motivation and Objectives

In industrial and regulatory contexts, accurate access to chemical safety information is essential for ensuring compliance, supporting risk assessment, and maintaining safe operational practices. However, the manual processing of SDS documents is time-consuming, error-prone, and difficult to scale, particularly when dealing with large collections of documents.

From a Data Science perspective, SDS documents represent a realistic, challenging, and relevant use case for information extraction. Their semi-structured nature, combined with domain-specific terminology and variability across sources, makes them suitable for evaluating different Natural Language Processing approaches. They provide a realistic scenario for comparing traditional rule-based techniques with more recent machine learning models, such as transformer-based architectures.

In this context, the main objective of this work is to investigate the feasibility of applying different Natural Language Processing techniques to the extraction and classification of selected fields from Safety Data Sheets. Rather than addressing the full scope of SDS processing, this work focuses on a subset of sections and fields of different types, allowing for a controlled evaluation of the proposed methods.

To achieve this, several complementary approaches are explored, including rule-based methods using regular expressions, transformer-based models such as Bidirectional Encoder Representations from Transformers (BERT), and prompting techniques using Large Language Models (LLMs). These approaches are applied to the same set of target attributes and compared in terms of effectiveness and scalability.

This work aims to:

- Develop models that are able to extract and classify selected fields of different types from a subset of SDS;
- Apply and adapt different extraction strategies, including rule-based and machine learning approaches;
- Evaluate and compare the performance of these approaches using appropriate metrics;
- Assess the feasibility and limitations of each approach in the context of SDS information extraction.

1.2. Methodology Overview

The methodology adopted in this work follows a structured process, progressing from initial research to data preparation, model development, and evaluation. Each phase contributes to the development and assessment of the proposed approaches for information extraction from Safety Data Sheets.

1.2.1. Research Phase

An initial research phase was conducted to establish the theoretical and practical foundations of the work. This included a review of the evolution of Natural Language Processing techniques from early rule-based approaches to modern transformer-based models, as well as an analysis of related work on information extraction from Safety Data Sheets and similar documents.

This phase supported the identification of suitable methodologies and guided the design of the experimental approach. It also provided the necessary context for understanding the strengths and limitations of different techniques, which informed the selection of the models explored in this work.

1.2.2. Data Preparation

The data used in the work consists of Safety Data Sheets documents in Portable Document Format (PDF) provided by VLM Compliance Solutions. Besides the SDS documents, a metadata file was also provided, containing ground truth annotations for several field attributes associated with each document. This enables the supervised evaluation of the extraction and classification methods.

During the data preparation stage, the PDF documents are processed to extract their textual content. A segmentation step is then applied to divide each document into its corresponding sections, leveraging the standardized structure of SDS. The extracted text is subsequently organized into a structured, model-ready dataset, where selected fields are linked to their corresponding textual context.

1.2.3. Models and Evaluation

To address the extraction and classification tasks, multiple approaches are explored. These include rule-based methods using regular expressions, transformer-based models such as BERT for classification tasks, and prompting techniques using Large Language Models for information extraction.

The performance of these approaches is evaluated using appropriate metrics, allowing for a comparative analysis of their effectiveness and limitations. This evaluation provides insights into the suitability of each method for the task of SDS information extraction.

1.3. Structure of the Work

This report is organized into several chapters that describe the development of the proposed solution, from problem definition to evaluation and conclusions.

- **Chapter 2- Problem to be Solved:** introduces Safety Data Sheets and their role in industrial and regulatory contexts. It defines the problem addressed in this work, highlighting the challenges associated with the extraction of structured information from SDS documents and the motivation for automation.
- **Chapter 3 - Background:** presents the theoretical foundations of the work, focusing on Natural Language Processing. It provides an overview of the evolution of NLP techniques, from early approaches to modern transformer-based models, which form the base of the methods explored in this work.
- **Chapter 4 -** Related Work**:** reviews existing approaches to information extraction from technical documents, with a particular focus on Safety Data Sheets and similar domains. It analyses both academic and industrial solutions, providing context for the methods adopted in this project.
- **Chapter 5 - Data Preparation:** describes the dataset used in this work and the preprocessing steps applied to prepare it for analysis. This includes the extraction of text from PDF documents, segmentation of SDS sections, and the construction of a structured dataset suitable for model application.
- **Chapter 6 - BERT and RegEx Models and Evaluation:** presents the rule-based and transformer-based approaches used for information extraction and classification. It details their implementation and evaluates their performance using defined metrics.
- **Chapter 7 - LLM Models and Evaluation:** explores the use of Large Language Models for information extraction through prompting techniques. It presents the methodology adopted and evaluates the results in comparison with the previous approaches.
- **Chapter 8 - Conclusion:** summarizes the main findings of the work, discussing the performance and trade-offs of the different approaches explored. It highlights the advantages and limitations of each method and outlines directions for future work.

2. Problem to be Solved

Safety Data Sheets (SDS) are essential documents in industrial safety. They ensure the procedures for safe handling, storage, and disposal of hazardous materials. Each SDS contains comprehensive information about a chemical substance like its composition, potential hazards, proper handling procedures, and emergency control measures. Despite their critical importance, the length and complexity of these documents pose significant challenges in their handling and categorization.

2.1. Problem overview

This project aims to develop a proof-of-concept system (Figure 1) capable of automating the process of information extraction from Safety Data Sheets using advanced Natural Language Processing (NLP) techniques and other classification methods.

As illustrated in Figure 1, the proposed solution follows a simple processing pipeline. Safety Data Sheets are first provided as input, after which their textual content is extracted and processed using a combination of information extraction techniques. This processing stage is responsible for identifying and extracting relevant information from the documents. The extracted data is then structured into parsed results, which can be stored for further use, analysis, or integration with other systems.

The objective is to demonstrate that automation can streamline and enhance the extraction of crucial details from SDS, potentially leading to a more effective process that delivers precise and consistent data while significantly reducing the time and effort required for manual review.

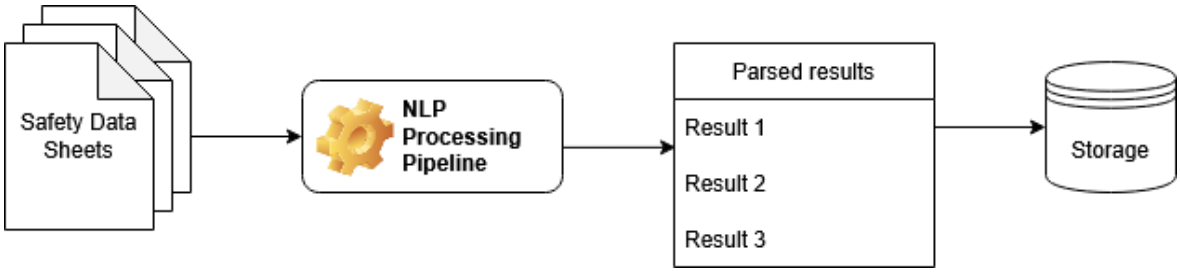

Figure 1:High-level architecture of the proposed system for processing SDS into structured outputs.

2.2. Importance of the Problem

Automating the information extraction process for SDS can have several advantages:

1. **Efficiency in Data Handling:** The manual extraction of data from SDS is both time-consuming and prone to errors. Automating this process significantly speeds up data handling, allowing safety professionals to concentrate on more critical tasks such as

risk assessment and mitigation. This improvement in efficiency helps organizations operate more smoothly and respond quickly to safety concerns;

2. **Resource Optimization:** By automating the extraction process, organizations can optimize their resources, reducing the need for extensive manual labour and minimizing associated costs. This optimization enhances overall productivity and allows human resources to focus on more strategic and high value activities;
3. **Compliance:** Timely and accurate extraction of safety information is vital for organizations to comply with regulatory requirements and ensure the safety of their personnel. This contributes to maintaining high safety standards and adhering to regulations such as the Classification, Labelling and Packaging (CLP) [3];
4. **Improved Hazard Communication**: Automation ensures that hazard information is accurately extracted and communicated, supporting better decision-making in workplaces and enhancing overall safety;
5. **Scalability and Large-Scale Processing:** As organizations handle increasing volumes of Safety Data Sheets manual processing becomes impractical. An automated extraction system enables scalable processing of hundreds or thousands of SDS documents with consistent performance and standardized outputs.

2.3. Safety Data Sheets

A Safety Data Sheet (SDS) is a comprehensive document prepared by the manufacturer of any chemical substance that provides critical information about the properties, hazards, safe handling, and emergency measures related to said substance. The primary objective of a SDS is to communicate critical safety information to ensure that the products are handled, stored, and disposed of safely. This information is not meant for the general consumer, instead, this information is focused on the industries that work with these chemicals.

2.3.1. SDS Contents Structure

A SDS follows a standardized format with 16 sections [4] that are designed to cover all critical aspects of chemical safety. These sections provide essential information such as product identification, hazard identification, and chemical composition. Additionally, they outline first-aid measures, fire-fighting procedures, and protocols for accidental releases. Safe handling and storage practices, exposure controls, and details on physical and chemical properties are also included. It also contains information about any regulatory requirements mandatory for the handling and disposal of the chemical.

- **Section 1 - Identification of the substance/chemical and manufacturer**
    - Product identification, manufacturer or distributor details, recommended use, and restrictions on use. It provides basic information about the chemical and its supplier.
- **Section 2 - Hazards identification**
    - Details the classification of the chemical, signal words, hazard statements, symbols, and precautionary statements. It helps users understand the hazards associated with the chemical.

- **Section 3 - Composition/information on ingredients**
    - Lists the chemical ingredients, including common names, synonyms, and Chemical Abstracts Service (CAS) Registry Numbers. It provides details on the chemical composition and any hazardous components.
- **Section 4 - First aid measures**
    - Describes the necessary first-aid steps to take in case of exposure to the chemical. It includes instructions for different types of exposure such as inhalation, skin and eye contact, and ingestion.
- **Section 5 - Firefighting measures**
    - Lists suitable extinguishing methods, specific hazards arising from the chemical combustion, and special protective equipment for firefighting of this substance. This section helps prepare for fire emergencies involving the chemical.
- **Section 6 - Accidental release measure**
    - Outlines procedures for containment and cleanup of spills or leaks. It includes recommendations for personal precautions, protective equipment, and emergency procedures.
- **Section 7 - Handling and storage**
    - Provides guidelines on safe handling practices and conditions for safe storage. This section helps prevent accidents and ensures that the chemical is stored under proper conditions.
- **Section 8 - Exposure controls/personal protection**
    - Details exposure limits, engineering controls, and personal protective equipment (PPE). It helps users understand how to protect themselves from harmful exposure to the chemical.
- **Section 9 - Physical and chemical properties**
    - Lists the chemical's physical and chemical characteristics such as appearance, odour, pH, melting point, boiling point, flash point, and solubility. This section provides essential information about the chemical's properties.
- **Section 10 - Stability and reactivity**
    - Describes the chemical's stability and potential hazardous reactions under various conditions. It includes information on reactivity, chemical stability, and conditions to avoid.
- **Section 11 - Toxicological information**
    - Provides details on the health effects and symptoms of exposure. It includes information on routes of exposure, related symptoms, acute and chronic effects, and numerical measures of toxicity.
- **Section 12 - Ecological information**
    - Covers the environmental impact of the chemical, including its toxicity to aquatic and terrestrial life, bioaccumulation potential, and mobility in soil. This section helps assess the environmental risks of the chemical.

- **Section 13 - Disposal considerations**
    - Recommends proper disposal methods for the chemical and its container. It includes information on disposal, recycling, and incineration practices.
- **Section 14 - Transport information**
    - Provides guidelines for the safe transportation of the chemical. It includes the UN number (United Nations number), proper shipping name, transport hazard class, packing group, and any special precautions.
- **Section 15 - Regulatory information**
    - Lists relevant regulations and legislation specific to the chemical. This section ensures that users are aware of the legal requirements for handling and using the chemical.
- **Section 16 - Other information**
    - Includes any additional information that may be relevant, such as the date of preparation or last revision of the SDS, and where to find more information if needed.

The solution implemented in this project leverages this predictable structure to efficiently extract the required information from the Safety Data Sheets.

2.3.2. Regulations

Safety Data Sheets (SDS) are governed by various global and regional regulations to ensure consistency and reliability of information across different countries and industries. One of the most important regulatory frameworks is the Globally Harmonized System of Classification and Labelling of Chemicals (GHS) [5], developed by the United Nations. The GHS (Figure 2) provides a universal standard for classifying chemicals and communicating hazards through SDSs and chemical labels. Its objective is to protect human health and the environment by ensuring that information on chemical hazards is consistent and easily understandable worldwide [2].

In addition to the GHS, regional regulations are vital in governing SDS. In the European Union, the Classification, Labelling and Packaging (CLP) Regulation [6] mandates that manufacturers and importers provide an SDS for hazardous chemicals. This regulation aligns with the GHS and guarantees that chemical safety information is standardized across EU member states. Similarly, in the United States, OSHA's Hazard Communication Standard (HCS) requires chemical manufacturers, distributors, and importers to provide SDSs to communicate chemical hazards in the workplace. These regulations ensure that workers have access to detailed information about the chemicals they handle, promoting workplace safety and regulatory compliance [7].

Figure 2: GHS hazards pictograms [7]

2.4. Summary

This chapter defines the problem addressed in this project: the automation of structured information extraction from Safety Data Sheets. Although SDS documents follow a standardized 16-section structure mandated by international regulatory frameworks, their length, linguistic variability, and semi-structured nature make manual extraction inefficient and difficult to scale.

The importance of this problem is discussed in terms of operational efficiency, resource optimization, regulatory compliance, hazard communication, and scalability. These factors highlight the practical and industrial relevance of developing automated methods capable of reliably extracting key safety information.

To address this challenge, it is necessary to understand the technological foundations that enable automated text processing. The next chapter presents the theoretical background of Natural Language Processing (NLP), outlining its evolution from early rule-based approaches to modern transformer-based architectures, which form the basis of the methods explored in this work.

3. Background

This chapter provides the foundational knowledge necessary to understand the key concepts and methodologies used throughout this project. This background is essential for contextualizing the applicability of Natural Language Processing (NLP) to the task of automating the information extraction from Safety Data Sheets (SDS). By reviewing the evolution of NLP, its current challenges, and modern approaches, we can better appreciate and understand the complexity of this field and how advancements like deep learning and the transformer architectures, have reshaped the landscape of this field in the past years.

These foundations serve to justify the choice of methods used in this project and demonstrate the relevance of state-of-the-art NLP models in automating the information extraction from SDS.

3.1. Natural Language Processing

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) that combines linguistics and computer science to enable computers to understand, interpret, and generate human language [8]. The scope of NLP encompasses a wide range of tasks, from basic text analysis to complex language generation and understanding, facilitating the extraction and processing of valuable information from unstructured text data.

NLP has grown significantly in importance over the past decade due to the rapid increase in digital text data and the need for efficient ways to process human language. According to a report by “Markets and Markets”, the global NLP market is projected to grow from $18.9 billion dollars in 2023 to reach $68.1 billion dollars by 2028, with a Compound Annual Growth Rate (CAGR) of 29.3% [9]. This growth reflects the expanding role of NLP in industries seeking to leverage large volumes of textual data for better decision-making and user interaction.

3.1.1. NLP Applicability in Modern Technology and Industries

NLP plays a crucial role in a wide array of applications, showcasing its versatility and importance in both consumer-facing and enterprise-level technologies. Some of the most prominent applications include:

- **Virtual Assistants:** Companies like Apple (Siri)[10], Google (Google Assistant/Gemini)[11], and Amazon (Alexa)[12] use NLP to power their virtual assistant’s software, enabling natural language interactions with users.
- **Search Engines:** Google's BERT (Bidirectional Encoder Representations from Transformers) model has revolutionized search engine capabilities by better understanding the context of search queries.
- **Machine Translation:** NLP powers translation tools like Google Translate and DeepL, which handle hundreds of languages and serve millions of users daily. These

systems leverage deep learning models to enhance translation accuracy, context comprehension, and fluency, making multilingual communication more effective and natural.

- **Healthcare:** NLP is used to extract information from medical records, assist in clinical decision support, and improve patient care. Tools like the "IBM watsonx Assistant AI healthcare chatbots" use NLP to analyze unstructured medical data.
- **Customer Service:** NLP enables companies to deploy chatbots that efficiently manage customer inquiries, provide quick responses when handling common issues, reducing the need for human agents and, mostly, improving customer satisfaction.

The applications of NLP extend beyond these examples, demonstrating its versatility and importance across various modern industries. The release of OpenAI's GPT-3.5 in November 2022 further spotlighted the rapid advancements in NLP, drawing attention to the potential of large-scale language models to handle increasingly complex language tasks. As NLP technologies continue to evolve, particularly with advancements in Large Language Models (LLMs), their impact on technology and industry will only deepen, making them indispensable tools for a wide variety of tasks that require language understanding and generation.

3.1.2. History of NLP

Natural Language Processing (NLP) today powers diverse applications, from virtual assistants and automated translation systems to sentiment analysis in social media and sophisticated information extraction in specialized domains. However, this level of sophistication is the product of decades of gradual advancement, rooted in simpler, rule-based approaches that have since evolved into complex, AI-driven models. The history of NLP reflects this journey, marked by significant paradigm shifts that each addressed the limitations of earlier methods, ultimately paving the way for the robust applications we see today.

Early Stages of NLP

The roots of NLP trace back to the 1950s, coinciding with the birth of artificial intelligence. Early NLP efforts focused on symbolic approaches, where language was analyzed through manually crafted rules. A notable milestone from this period was the 1954 Georgetown-IBM experiment, which demonstrated the potential of machine translation by translating over 60 Russian sentences into English [13]. Though limited, this experiment showed that computers could process language for translation tasks, igniting interest in further NLP research.

In 1957, Noam Chomsky's publication of Syntactic Structures revolutionized linguistic theory with his ideas on transformational grammar, significantly influencing early NLP by providing a structured, formal approach to syntax [14]. While groundbreaking, these early systems were limited in scope, focusing mainly on syntax and operating with simple, manual rule-based models.

A decade later, ELIZA (Figure 3) by Joseph Weizenbaum in 1966 was a very significant development. ELIZA, a "chatbot", simulated a person-centered therapist, engaging in dialogue through pattern matching and substitution techniques. Despite its simplicity, ELIZA showed the potential for natural language interaction between humans and machines, a crucial milestone that laid the groundwork for future conversational AI [15].

While ELIZA was limited to basic pattern-matching, today's top-tier conversational agents, such as GPT-4, can generate conversations with remarkable coherence, leveraging vast datasets and advanced neural architectures. It is intriguing that the concept of conversational AI dates to the 1960s; however, the implementation and capabilities of modern systems differ profoundly, showcasing the extraordinary advancements in NLP and AI since ELIZA's time.

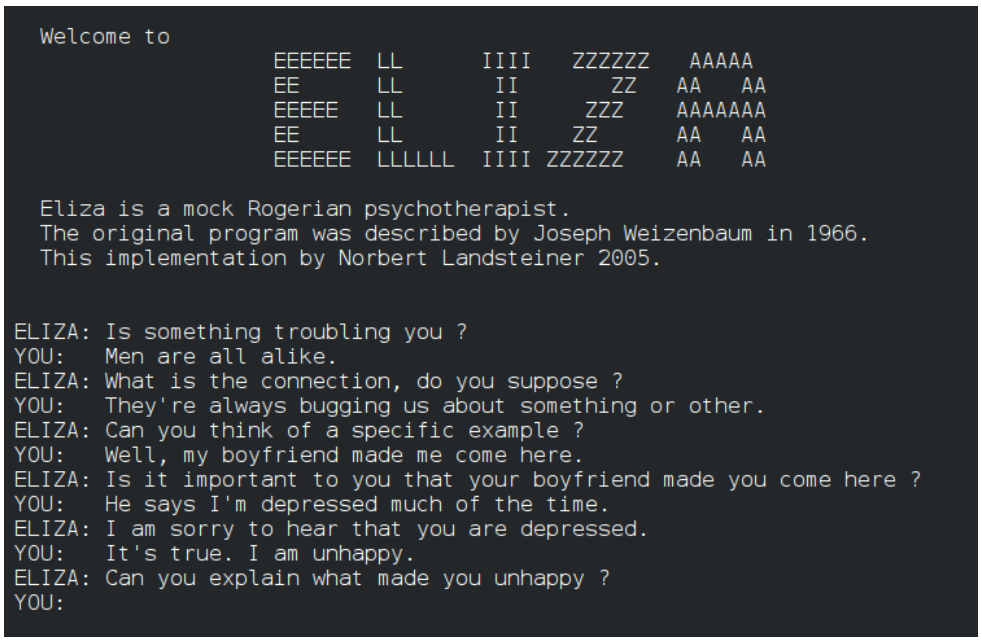

Figure 3: ELIZA's chat like interface [16]

The Shift to Statistical Methods

The 1980s and 1990s marked a pivotal shift in NLP with the adoption of statistical methods. The increasing availability of digital text corpora and advances in computing allowed researchers to move away from rule-based systems towards probabilistic models, driven by empirical data. Techniques such as Hidden Markov Models (HMMs) and Naive Bayes classifiers became popular, enabling more flexible and scalable approaches to tasks like part-of-speech tagging, text classification, and information retrieval.

One of the era's breakthroughs was IBM's work on Statistical Machine Translation (SMT) in 1988. By leveraging large parallel corpora, IBM researchers developed a system that improved translation quality through statistical analysis, representing a departure from traditional rule-based translation and broadening the scope of NLP applications [17]. This statistical paradigm laid the foundation for machine translation and other NLP tasks, proving that language could be effectively modeled with probabilistic methods.

The Rise of Machine Learning and Deep Learning

In the early 2000s, machine learning emerged as a dominant paradigm in NLP, especially with the development of support vector machines, neural networks, and more advanced algorithms. This period saw the introduction of word embeddings, such as Word2Vec, developed by Mikolov in 2013. Word2Vec (Figure 4) represented words in continuous vector spaces, capturing semantic relationships and enabling more nuanced language understanding [18].

However, deep learning truly transformed NLP in the 2010s. Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Convolutional Neural Networks (CNNs) allowed for more sophisticated language modeling, leading to breakthroughs in tasks like machine translation, named entity recognition, and sentiment analysis. A notable advancement was the introduction of sequence-to-sequence (Seq2Seq) models using RNNs in 2014, which dramatically improved machine translation and other sequence transduction tasks [19].

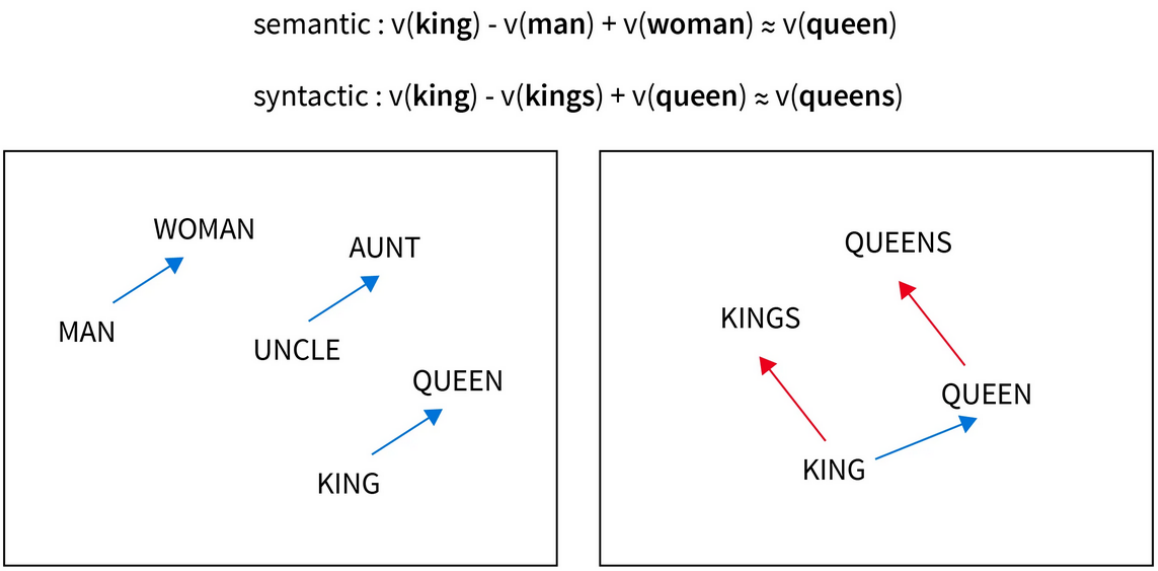

Figure 4: Representation of the relation between words in word2vec (adaptation from [20])

The Era of Transformer Models

The most significant recent advancement in NLP came with the introduction of the Transformer architecture by a team working to improve Google Translate capabilities. These Transformers models replaced the more commonly used RNN used in sequence-to-sequence tasks like translation at the time. With the self-attention mechanism, Transformers are able to capture contextual relationships across entire sequences without relying on sequential processing. This approach led to more powerful and efficient models, transforming NLP across various applications [21].

Following the Transformer, models such as BERT (Bidirectional Encoder Representations from Transformers) [1] and GPT (Generative Pre-trained Transformer) [22] achieved state-of-the-art performance across NLP tasks, from question answering to text generation. The

release of GPT-3 in 2020 marked an unprecedented advancement in language generation, demonstrating the potential of large-scale language models to perform a wide range of tasks with minimal task-specific training [23].

Transformers have propelled NLP into a new era (Figure 5), where large pre-trained models set new standards for performance and flexibility. These developments set the stage for this project, which leverages state-of-the-art Transformer models to automate information extraction from Safety Data Sheets, showcasing the practical applications of cutting-edge NLP technologies in specialized domains.

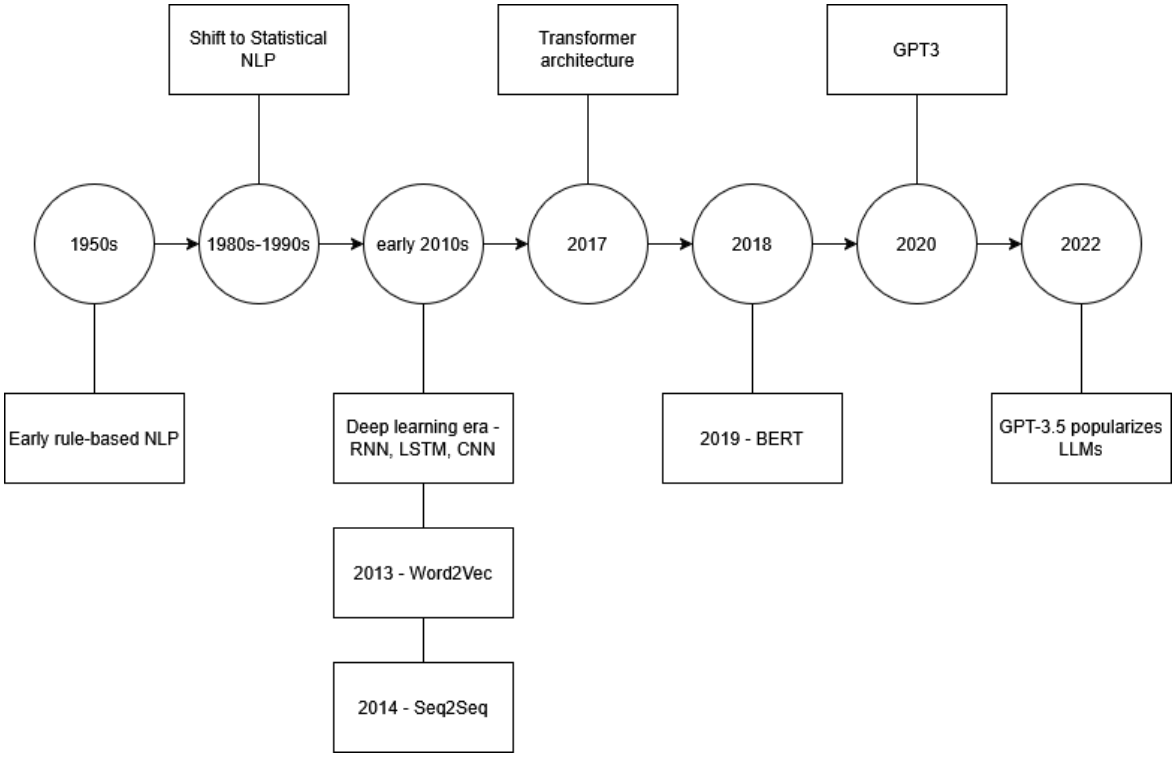

Figure 5: Representation of the evolution of NLP techniques in time

3.1.3. Challenges of NLP

As explored above, advances in NLP can be traced to key moments in its history. Despite these remarkable developments, NLP continues to face significant hurdles that affect both the efficacy and applicability of language models. These challenges highlight the need for continuous innovation and explain the emergence of more sophisticated models, such as BERT and other transformer-based architectures.

Language Ambiguity

One of the most fundamental challenges in NLP is addressing language ambiguity. This challenge is made even more complex by the existence of hundreds of human languages, each with its own unique structures, idioms, and cultural nuances. However, despite that, even within a single language, significant ambiguity can arise. Words and phrases often have

multiple meanings that change based on context. For example, the word ”run” can mean operating a machine, managing a business, the physical action of moving quickly on foot, and it can even be both a verb or a noun, depending on the context.

While humans can effortlessly interpret these variations using contextual clues and background knowledge, machines face substantial difficulties in making such distinctions. Overcoming this ambiguity requires NLP models to develop a deep and nuanced understanding of context, a challenge that has driven the advancement of attention-based models like BERT, which excel at capturing contextual relationships within language.

Complexity of natural language

Even within the same language, syntax, grammar, and semantics can vary depending on region, dialect, or even individual style. This variability makes it difficult for models trained on one dataset to generalize effectively to other types of text. Figures of speech such as idioms, metaphors, and sarcasm pose additional challenges, as they often require understanding what goes beyond literal interpretation. Addressing these complexities is essential for real-world applications like chatbots and social media analysis, where nuanced language comprehension is critical for delivering accurate and contextually appropriate results.

Data quality and availability

Advanced NLP models based on deep learning require large amounts of high-quality data to be trained effectively. Acquiring annotated datasets that accurately represent the language across different contexts poses a significant challenge. Issues such as data biases, gaps in training corpora, and lack of diversity can severely impact the generalization ability of NLP models, leading to suboptimal performance in real-world applications.

In the context of this project, which focuses on extracting information from highly specialized documents like Safety Data Sheets (SDS), data quality and availability become even more critical. Insufficient or biased training data can result in models that fail to accurately interpret domain-specific terminology or document structures, undermining the effectiveness of the system and potentially leading to inaccurate or incomplete information extraction.

Computational Challenges

The rapid evolution of NLP has brought with it a significant increase in computational demands. Training large-scale models such as those based on transformer architectures requires considerable processing power and memory. The complexity of these models not only makes them resource-intensive but also raises barriers to entry for smaller organizations or research teams that may not have access to high-performance computing resources.

3.2. Pre Transformer NLP Deep Learning

Deep learning has profoundly transformed the landscape of Natural Language Processing (NLP), addressing many inherent limitations of more traditional approaches. Unlike classical methods reliant on manually engineered features, deep learning utilizes neural networks to automatically learn sophisticated textual representations. These techniques have significantly improved the most common NLP tasks like text classification, information extraction, named entity recognition (NER), machine translation, and question-answering systems.

3.2.1. Neural networks

The widespread adoption of neural networks in NLP was facilitated in the early 2010s by several converging factors like the increasing availability of large-scale text corpora, substantial improvements in computing power (notably GPU acceleration), and the emergence of open-source deep learning frameworks such as Theano, TensorFlow, and PyTorch [24]. These advances made it feasible to train deep, data-driven models capable of learning complex representations of language without relying on hand-engineered features.

At their core, neural networks are composed of layers of interconnected nodes or "neurons", where each neuron applies a linear transformation to its inputs followed by a non-linear activation function. When multiple layers are stacked, the network can learn increasingly abstract features. In the context of NLP, neural networks map sequences of tokens into dense vector representations that encode semantic and syntactic information [25].

Multilayer Perceptron

Multilayer Perceptrons (MLP) are among the simplest neural architectures applied to natural language processing during the early stages of deep learning adoption. In these models, input text is typically converted into fixed-size vector representations – such as bag-of-words, TF-IDF (Term Frequency-Inverse Document Frequency) scores [26], or average word embeddings – and passed through one or more fully connected layers. MLPs proved effective in tasks like document classification and sentiment analysis, where a global understanding of the text could often be inferred from the presence of key terms, rather than their order [27].

However, a major limitation of MLPs is their inability to account for the sequential structure of language. Since they process inputs independently and lack any notion of temporal or syntactic relationships, MLPs are poorly suited for tasks that depend on understanding word order, grammatical structure, or contextual nuance, such as advance NLP tasks like information extraction.

Recurrent Neural Networks

As the limitations of feedforward models became apparent for tasks involving sequential data, Recurrent Neural Networks (RNNs) emerged as a natural next step. Unlike MLPs, RNNs are explicitly designed to process sequences by maintaining a hidden state that is

updated at each time step. This allows information from earlier tokens to influence the interpretation of later ones, enabling the model to learn dependencies across a sequence [27].

This capacity made RNNs particularly well-suited to a wide range of NLP applications, including language modeling, part-of-speech tagging, and machine translation—where context and word order are essential. By iteratively processing one token at a time and updating internal memory, RNNs introduced a degree of temporal awareness previously lacking in neural NLP systems.

However, despite their theoretical advantages, standard RNNs struggle in practice with learning long-range dependencies. As sequences grow longer, the influence of earlier inputs tends to diminish due to the vanishing gradient problem, which prevents the model's from retaining important information over time. This issue, along with training inefficiencies related to their inherently sequential nature, motivated the development of more robust architectures.

Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory (LSTM) networks are an evolution of standard Recurrent Neural Networks (RNNs) designed to overcome their most significant limitation—the previously mentioned vanishing gradient problem.

LSTMs address this issue by introducing a specialized architecture with a memory cell and a set of gates, *input*, *forget*, and *output* gates, that regulate the flow of information over time. These gates function as control mechanisms, allowing the network to selectively retain, update, or discard information at each time step. This architecture enables LSTMs to maintain relevant context across long sequences, making them particularly effective for complex NLP tasks such as machine translation, text summarization, speech recognition, and question answering [25, 26].

One of the most notable applications of LSTM networks in natural language processing was Google Translate's Neural Machine Translation (NMT) system, introduced in 2016. Previously, *Google Translate* relied on phrase-based statistical machine translation, which segmented text into fixed-length phrases without considering long-range dependencies between words. The adoption of an LSTM-based NMT model transformed the system’s performance, allowing it to capture the full context of entire sentences rather than just local phrases. This resulted in more accurate, fluent, and contextually coherent translations [30].

The success of Google’s LSTM-based translation system highlighted the ability of LSTMs to effectively model long sequences and contextual relationships, solidifying their role as a foundational architecture in NLP during the 2010s.

However, despite their success, LSTMs are computationally complex due to their sequential nature, which prevents efficient parallelization during training. This limitation, coupled with the increasing demand for faster and more scalable models, paved the way for a new paradigm in neural NLP – the Transformer architecture, which revolutionized the field by offering a highly scalable, parallelizable approach to sequence processing.

3.3. The transformer Architecture

Natural Language Processing (NLP) has undergone a series of transformative advancements over the years, but few innovations have been as impactful as the introduction of Transformers. Introduced by Vaswani et al. in 2017 in the popular paper Attention Is All You Need, Transformers are a type of neural network architecture that fundamentally changed how machines understand and generate text [21].

Transformers were designed to overcome the inherent limitations of earlier sequential models, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. These traditional models processed text one word at a time, a sequential approach that led to several critical issues:

- **Sequential Processing:** RNNs and LSTMs process text one token at a time, making them slow and difficult to parallelize. This significantly limited their scalability, especially for long texts;
- **Vanishing Gradient Problem:** Even with advancements like LSTMs, these models struggled to learn long-range dependencies because gradient values could diminish or explode during training;
- **Limited Contextual Awareness:** RNNs and LSTMs could only capture a narrow window of context, making it difficult for them to understand complex relationships between distant words in a sentence.

These challenges highlighted the need for a new approach—one that could efficiently capture complex relationships between words across entire texts without the limitations of sequential processing. This need led to the development of the Transformer Architecture, a revolutionary neural network architecture that overcame these issues and redefined Natural Language Processing (NLP).

3.3.1. How Transformers Work

Transformers operate using a fundamentally different mechanism from previous neural networks models; self-attention, which allows each word in a text sequence to directly attend to every other word, regardless of their distance. This **self-attention mechanism** is combined with a process called **multi-head attention**, which allows the model to learn multiple attention patterns at once, capturing diverse relationships between words.

Self-Attention Mechanism

The core of the Transformer architecture is self-attention. In this mechanism, each word in a sentence is transformed into three vectors: **Query (Q)**, **Key (K)**, and **Value (V)**:

- **Query (Q):** Represents the word we are focusing on (the word for which we want to determine relevant context).
- **Key (K):** Represents each word in the text sequence that the model can attend to.

- **Value (V):** Contains the contextual information of each word that can be used for generating a weighted representation.

These vectors are used to calculate attention using the following formula:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where:

- ***T*** represents the number of tokens in the input sequence. Tokens are not necessarily full words, and depending on the model's tokenization strategy can also be sub words, characters and even punctuation;
- d_k represents the dimensionality of the query and key vectors. This is the number of numerical values (features) in each vector. It is typically set as the model's embedding dimension divided by the number of attention heads (described in the next section).
- ***Softmax*** is a normalization function that converts the attention scores into a probability distribution, where each score becomes a weight between 0 and 1, and all weights sum to 1.

This formula computes a weighted sum of the value vectors, where the weights are determined by the similarity between the query and key vectors, scaled by the square root of their dimension (d_k). This process allows each word to focus on other words that are most relevant to it, effectively capturing complex relationships within the text. This is visualized in Figure 6 [31].

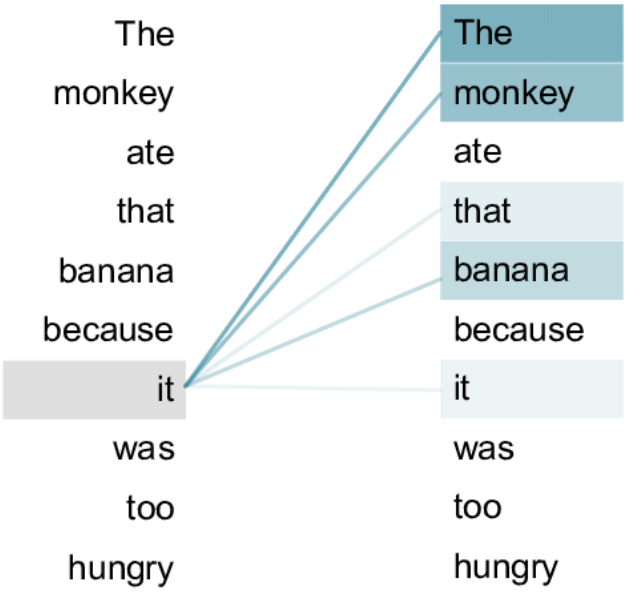

Figure 6: Visualization of the attention mechanism [31]

Multi-Head Attention

Multi-head attention extends the concept of self-attention by allowing the model to learn multiple attention patterns at once. Instead of calculating attention just once, the model uses several independent attention heads, each capturing different aspects of the text.

Each head has its own set of Query, Key, and Value matrices, allowing it to focus on different relationships in the text. The results of all heads are then combined, providing a richer representation of the text. This is defined by the following formula below:

$$MultiHead(Q,K,V) = Concat(head_1, head_2, \dots, head_h)W^O$$
$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right),$$

where:

- $\boldsymbol{h}$ is the number of attention heads;
- $\boldsymbol{W_i^Q}$, $\boldsymbol{W_i^K}$, $\boldsymbol{W_i^V}$ are the learned weight matrices specific to each head;
- $\boldsymbol{W^O}$ is the output projection matrix that combines the heads, ensuring that the final multi-head attention output has the correct dimensionality.

Multi-head attention is essential because it enables the model to capture multiple perspectives of the text in parallel. Each head can focus on different aspects of the language, such as identifying grammatical structure in one head, recognizing semantic relationships in another, and capturing contextual nuances in others. By combining these perspectives, the model achieves a much richer and more comprehensive understanding of the text.

Positional Encoding

Since Transformers process text in parallel, they lack a natural understanding of word order. To overcome this, positional encodings are added to each word representation. These encodings provide each word with information about its position in the text, ensuring that the model can maintain a sense of sequence.

Encoder-Decoder Structure

The original Transformer architecture is designed around an encoder-decoder structure Figure 7:

- **Encoder:** Takes the input text, processes it, and learns a contextual representation using self-attention and feedforward layers;
- **Decoder:** Uses this contextual representation to generate output text, also using self-attention but with an added focus on the encoder output (encoder-decoder attention).

This architecture is highly versatile, capable of performing tasks like machine translation, text generation, and text summarization. However, many of the most influential models based on Transformers are simplified variations of this architecture:

- **BERT (Bidirectional Encoder Representations from Transformers):** Uses only the encoder, making it exceptionally effective for natural language understanding tasks;
- **GPT (Generative Pre-trained Transformer):** Uses only the decoder, making it highly effective for text generation.

These models highlight the adaptability of the Transformer architecture, and they represent the foundation of state-of-the-art NLP. They also directly connect to this project, where BERT and LLaMA3 (Large Language Model Meta AI 3) are employed for information extraction from Safety Data Sheets (SDS).

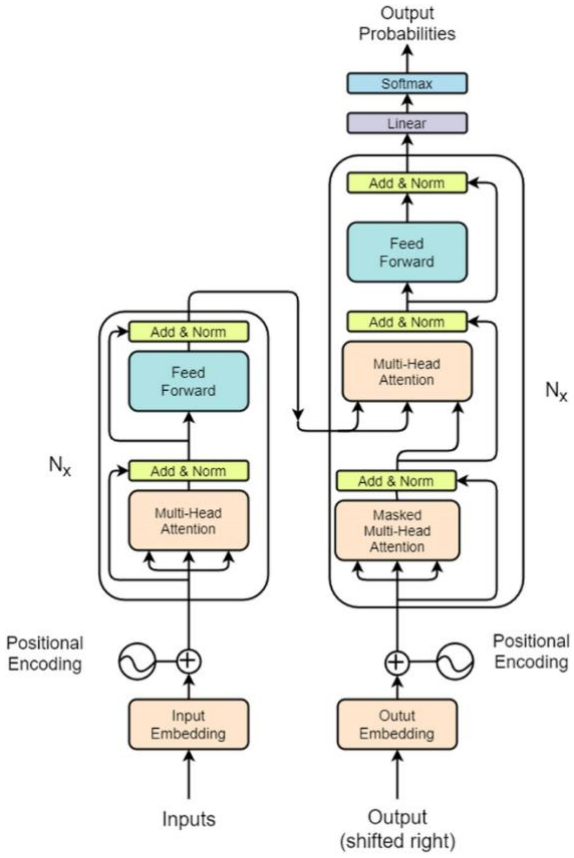

Figure 7: Transformer model architecture [21]

3.3.2. Key transformer models

The introduction of Transformers gave rise to a series of highly influential Large Language Models (LLMs), each designed to leverage the architecture's unique strengths. Among these, three models stand out for their impact and versatility: BERT, GPT, and LLaMA (Large Language Model Meta AI). These models exemplify how the Transformer architecture could be adapted for different purposes, from text understanding to text generation.

BERT (Bidirectional Encoder Representations from Transformers)

Introduced in late 2018, BERT presented a new approach to text representation by employing deep bidirectional learning. Unlike previous models that processed text

sequentially or independently in both directions, BERT simultaneously analyzes the entire context of a word by looking both backward and forward in the sentence. This bidirectional capability greatly improves its performance in natural language understanding tasks such as text classification, named entity recognition, and question answering.

The architectural innovation of BERT comes mostly from its pre-training techniques, namely Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM trains the model to predict randomly masked (hidden) tokens in the text, enabling it to learn the context of each word in the sentence. NSP further improves BERT sentence-level relationships by training the model to determine if pairs of sentences logically follow each other. These methodologies allowed BERT [1] to achieve state-of-the-art performance in numerous NLP benchmarks.

In this project, BERT was specifically used as a targeted classifier, where each individual model was trained to identify a single property within Safety Data Sheets documents. This is intended to produce precise and reliable classification outcomes, leveraging BERT's suitability for fine-grained, property-specific extraction tasks.

GPT (Generative Pre-trained Transformer)

Introduced by OpenAI in 2018, the GPT family of models focuses predominantly on text generation and excels in tasks involving coherent and contextually relevant text composition. GPT utilizes a unidirectional, auto-regressive approach, predicting the next word based solely on preceding words. This characteristic makes GPT especially effective for tasks such as text summarization, content generation, and dialogue systems, where generating coherent and contextually consistent output is essential [22].

OpenAI's ChatGPT, based on GPT architecture, played a significant role as a catalyst for the widespread adoption and rapid advancement of AI technologies in recent years. Its accessibility and ability to generate human-like conversational responses sparked broad public interest and increased investment in NLP research and applications, driving the current AI boom.

LLaMA3 (Large Language Model Meta AI 3)

Introduced by Meta in 2024, LLaMA3 is part of an open-source initiative aimed at advancing research and accessibility in the NLP community. As a Transformer-based model, LLaMA3 provides robust and flexible text processing capabilities. Unlike proprietary alternatives, its open-source nature allows researchers and developers to customize and adapt it to various specific tasks [32].

In this project, LLaMA3 was leveraged primarily due to its openness and capability to handle multiple information extraction tasks simultaneously. It was particularly effective for answering multiple queries and extracting several properties from Safety Data Sheets in a single operation, thus reducing the complexity and computational resources required compared to using multiple targeted classifiers, like the BERT approach.

3.4. Summary

This chapter introduces the core concepts of Natural Language Processing, tracing its evolution from rule-based and statistical methods to modern deep learning approaches. The recent emergence of transformer architecture marked a significant shift in the field, enabling effective handling of context, ambiguity, and large-scale textual data.

Transformer-based models, including architectures such as BERT and GPT models, play a central role in current information extraction tasks. Their ability to model contextual relationships in text makes them very suitable for processing semi-structured and domain-specific documents like Safety Data Sheets.

These foundations directly support the methodology choices of this work. The next chapter reviews existing approaches to document information extraction, with a focus on technical and semi-structured documents.

4. Related Work

The extraction of structured information from unstructured or semi-structured documents has become increasingly important with the rise of machine learning and natural language processing (NLP). In fields such as chemistry, healthcare, and finance, automating the retrieval of key information reduces the burden of manual entry and enables scalable processing of large document collections.

Safety Data Sheets (SDS) represent a particularly challenging type of document. Their layouts vary significantly between providers, often mixing free text, tabular data, and domain-specific terminology. Over time, both academic and industrial efforts have addressed the problem of automatically extracting chemical composition, hazard classifications, and regulatory details from SDS and related technical documents. This section reviews a selection of relevant approaches.

4.1. Academic Approaches to Document Information Extraction

Research on information extraction from documents covers a broad range of strategies that have evolved alongside advances in natural language processing and machine learning. Over time, studies have explored how to convert the unstructured or semi-structured content of technical documents into machine-readable formats, each adopting different assumptions about how information is organized and how meaning can be derived. Within this context, several academic directions stand out as particularly relevant for SDS: those centered on recognizing domain-specific entities, those that model the spatial and visual structure of documents, and those that combine statistical learning with rule-based logic to manage real-world variability.

4.1.1. Chemical Named Entity Recognition

A basic research direction is chemical named entity recognition (CNER), which aims to detect and classify mentions of chemical substances and formulas within unstructured text. This is a critical capability for SDS parsing, since much of the valuable information, such as substance names or hazard descriptors, appears as free text.

A recent contribution to hazardous chemical named entity recognition [33] introduced a model named BERT-BiLSTM-Self-Attention-CRF specifically designed for recognizing hazardous chemical information. By combining pre-trained BERT embeddings with sequential modeling and an attention mechanism, their system achieved an F1 score of 94.57%, outperforming earlier architectures such as BiLSTM-CRF and IDCNN-CRF that formed its base. Importantly, they also constructed a domain-specific dataset by crawling and annotating hazardous chemical texts, addressing the lack of pre-existing corpora for this domain.

Although their work was not directly applied to SDS, there is a clear connection. SDS frequently contain the same entity types such as chemical names, abbreviations, and hazards embedded in free text sections like Section 2 (Hazard Identification) or Section 3 (Composition/Information on Ingredients) of Safety Data Sheets. The methodology therefore provides a tested approach for the first step of SDS extraction: identifying and normalizing chemical entities. The main limitation, however, is data dependency: this method needs carefully annotated domain corpora, and such resources are scarce for SDS in Portuguese. This motivates the use of hybrid strategies or rule-based approaches in projects constrained by language or dataset size, like this one.

4.1.2. Layout-Aware and Multimodal Models

While Chemical NER approaches treat documents as plain sequences of text, SDS are inherently visually structured. Tables defining ingredient properties with multiple columns are common, and hazard pictograms convey standardized information graphically. Processing these documents requires models that not only read text but also interpret its spatial organization and visual cues.

LayoutLM [34] marked a breakthrough by incorporating 2D positional embeddings into a BERT-like transformer, enabling the model to understand relationships between tokens based on their coordinates on the page. This was followed by XYLayoutLM [35], which introduced richer multimodal interactions by integrating visual features extracted from the document image alongside text and position. These models achieved state-of-the-art results in form understanding, invoice parsing, and other tasks where layout carries semantic weight.

DocLLM [35] explored generative approaches, where the model produces structured outputs such as key–value pairs or tabular records directly from combined text and layout input. This paradigm is particularly attractive for SDS, where the goal is to generate clean, structured records ready for easily searchable storage.

However, their practical deployment is hampered by two key challenges. First, they require large annotated training sets with bounding boxes and layout metadata, which are rare for SDS, especially in Portuguese. Second, they are computationally demanding, often requiring specialized hardware and extensive fine-tuning. In resource-constrained environments, this makes them less feasible, even if they represent the cutting edge of document understanding.

4.1.3. Hybrid SDS-Specific Pipelines

The most directly applicable research to SDS extraction is the hybrid approach proposed by Suman et al. [36]. Recognizing that SDS are semi-structured but inconsistent across manufacturers and regions, they developed a multi-stage ensemble pipeline that combines deterministic and statistical methods. Rules and regular expressions are used to detect section boundaries and CAS number patterns, while machine learning models, including BERT-based NER and table detection networks, handle more ambiguous tasks.

The pipeline was trained and evaluated on a large-scale dataset of 20,000 annotated SDS, one of the most extensive corpora of its kind. It achieved document-level precision of 0.93, proving both the feasibility and robustness of the method. Notably, the authors emphasize breaking down the complex task of SDS parsing into manageable subtasks: detecting composition sections, extracting tables, identifying chemical names, and aligning them with CAS numbers and weight percentages. This modular design allows the system to achieve high precision while mitigating errors that would accumulate in a monolithic model.

For the present project, Suman et al.'s work is particularly instructive. It demonstrates that hybrid pipelines combining rules, regular expressions, and ML models are well-suited to the messy reality of SDS, where layout inconsistencies and domain-specific conventions challenge purely statistical models. It also highlights the gap between industrial-scale efforts (with access to large annotated corpora and compute infrastructure) and smaller research or applied projects, which must find ways to adapt these principles under tighter resource constraints.

4.2. Industry Use Cases and Tools

Beyond academic work, the existence of commercial SDS extraction platforms illustrates the practical demand for these technologies. Regulatory compliance, workplace safety, and supply chain management all require accurate and up-to-date chemical information, often across thousands of documents in multiple languages. Manual transcription is slow, error-prone, and costly; automated SDS parsing addresses these bottlenecks by delivering structured data that can feed into compliance databases, risk assessment systems, or inventory software.

Two examples highlight how industry has approached the problem. SDS Manager [37] provides an Application Programming Interface (API) for parsing and structuring SDS content, supporting multilingual inputs and exporting in machine-readable formats such as JSON (JavaScript Object Notation). This allows companies to integrate extracted chemical information directly into regulatory and inventory workflows. EcoOnline [38], meanwhile, embeds SDS extraction as part of a broader chemical safety management platform, linking parsed data with risk assessment tools and occupational safety dashboards. In both cases, the extraction process is not an end but a gateway to broader decision-support systems.

While the technical methods behind these services remain undisclosed, their adoption demonstrates that automated SDS parsing is both feasible and valued in practice. At the same time, the black-box nature of commercial systems underscores the role of academic and open-source efforts in exploring transparency, benchmarking, and adaptability across domains and languages.

4.3. Summary

The reviewed literature and industry efforts show that extracting structured data from SDS is both technically challenging and practically important. Academic studies propose approaches ranging from lightweight chemical NER to layout-aware transformers and hybrid pipelines, each balancing accuracy, cost, and adaptability. Meanwhile, commercial platforms demonstrate the real-world demand for SDS automation in compliance and risk management, though their proprietary nature leaves open questions of transparency, generalizability, and language coverage.

This project situates itself at the intersection of academic approaches and industrial needs. By combining rule-based extraction, a BERT-based classifier, and lightweight prompting with earlier-generation LLMs, it adopts a pragmatic methodology that remains feasible under real-world constraints. The focus on SDS in Portuguese and the use of resource-efficient techniques suitable for local deployment highlight both the limitations and opportunities encountered when applying state-of-the-art NLP methods to domain-specific, multilingual contexts.

The next chapter turns from this broader landscape to the details of the present work, beginning with the preparation of the SDS dataset and the processing steps required to make it suitable for analysis.

5. Data Preparation

Before any model could be developed, it was necessary to prepare a dataset that could be reliably used for training and evaluation. The original data available for this project consisted of two simple CSV files - *products.csv* and *metadata.csv* - each containing references to Safety Data Sheets (SDS) and related metadata. The *products.csv* file included a list of products with the URL of their respective SDS in PDF format, along with several properties of that product. The *metadata.csv* file contained additional information about each field/property of the corresponding SDS, including its data type and the section of the SDS where it appears. Although the information in these files was valuable, it was not directly suitable for use by Natural Language Processing (NLP) models.

To address this, a data preparation pipeline was designed to transform these raw inputs into a structured and model-ready corpus. The main objective was to produce a dataset that could be easily used by the different approaches explored in this work, including rule-based extraction methods (e.g., regular expressions), transformer-based models such as BERT, and large language models (LLMs) such as LLaMA3. Regardless of the model type, all approaches require data to be well-organized, clearly labelled, and contextually coherent.

The chosen strategy was to create a flexible JSON structure capable of combining the textual content of each SDS section with its corresponding properties, data types, and correct answers. This structure ensures that the same dataset can serve multiple experiments with minimal additional processing, providing a common foundation for both classical and modern NLP techniques.

The overall process was divided into three main phases:

1. **Phase I - Data Acquisition:** collecting all SDS documents in PDF format from the URLs provided in the *products.csv* file, ensuring a consistent mapping between each product and its source document;
2. **Phase II - Text Extraction and Segmentation:** extracting the textual content from each SDS and dividing it into 16 separate text files, corresponding to the standardized SDS sections defined by international safety regulations;
3. **Phase III - JSON Construction:** combining the extracted texts with the SDS information and metadata to produce a structured dataset ready for model input.

This chapter describes each of these phases in detail, from the collection of the original PDFs to the generation of the final JSON files. The result of this process is a reproducible and versatile dataset capable of supporting multiple NLP-based extraction methods, ensuring that every model developed in this work has access to consistent and well-prepared data.

5.1. Original Data

The original dataset used in this project was provided by *VLM Compliance Solutions* for academic purposes only. It consisted of two comma-separated values files (.csv), *products.csv* and *metadata.csv*, each containing complementary information related to Safety Data Sheets (SDS). These files served as the foundation for the entire data preparation process.

5.1.1. Products File

The *products.csv* file contained one entry per product, corresponding to a single Safety Data Sheet (SDS). Each entry included a URL pointing to the PDF version of the product's SDS. In addition to the SDS reference, each row contained several descriptive properties extracted from the document, such as product type, hazard classification, precautionary statements, and supplier information.

These properties represented the *ground-truth* values that would later serve as targets for model training and evaluation. This file was essential for both data acquisition and model validation phases of the project.

In total, the file contained 134 properties per Safety Data Sheet. Table 1 below lists some of the most relevant fields used throughout this work.

Table 1: Used properties of the *products.csv*

Field	Description	Use
fds	URL pointing to the original PDF Safety Data Sheet	Used to download the original SDS document
dangerous	Indicates whether the product described in the SDS is considered dangerous	Used for model training and evaluation
signal_word	Signal word associated with the product	Used for model training and evaluation
select_hs	Hazard statements (H-codes) associated with the product	Used for model training and evaluation

5.1.2. Metadata file

The *metadata.csv* file complemented the product information by describing each property listed in the *products.csv* file. Each row in this file represented a specific field or attribute present in the Safety Data Sheet (SDS) and included its section name, internal field identifier, human-readable label, and data type, as shown on Table 2.

Table 2: Description of the *metadata.csv* file

Field	Description
seccao	Name of the section where the property is found
campo	Internal property identifier
legenda	Human-readable label
tipo de dados	Type of data of the property

The purpose of this file was to provide contextual information about each field contained in the *products.csv*, such as the section of the SDS where the field appears and its general data type. However, the original column "tipo de dados" was too generic, as most entries were simply labelled as "general". To make the metadata more informative and aligned with the needs of the data-preparation pipeline, this file was later enriched and expanded with additional fields, including:

- **Section number:** Instead of relying only on the textual section title, a numerical identifier was introduced for easier indexing and programmatic access;
- **Detailed data type:** Since most fields were initially classified as *"general"*, a new field was added to specify whether a property corresponded to a number, date, string, or even a single or multi-value entry;
- **Natural Language Question:** In preparation for LLM-based model experimentation, a natural-language question was added to query the model about each specific property (e.g., "Is this product dangerous?").

These additions made the *metadata.csv* a critical component of the data preparation process. In Phase III, this enriched metadata guided the creation of the final JSON files by defining, for each property, where it should be found within the SDS text and how it should be represented in a structured question-answer format.

5.2. Phase I – Data Acquisition

The first phase of the data preparation pipeline focused on collecting all the Safety Data Sheets (SDS) in their original PDF format from the URLs provided in the previously mentioned *products.csv* file.

The process began by reading the *products.csv* file using the *pandas* library, which allowed efficient iteration over the entries and simplified the association between each product identifier and its corresponding SDS link. The field 'fds' in this file contained the URL pointing to the SDS document for each product. For each entry, an HTTP request was issued using the *requests* library in Python to download the PDF file directly from the web.

To ensure traceability, every downloaded file was named after a sequential identifier, following the pattern: 00001.pdf, 00002.pdf, 00003.pdf, etc.

This naming convention guaranteed a one-to-one mapping between each dataset record and its corresponding SDS document. Before each download, the script verified whether a file with the same identifier already existed locally, avoiding redundant requests. Basic validation checks, such as HTTP response codes, were performed to confirm successful downloads.

In total, the original 687 SDS entries were processed. Out of these, 634 documents were successfully downloaded and validated, while the others failed due to broken or unreachable URLs. Only the successfully collected documents formed the dataset and were used in the subsequent stages of the pipeline.

5.3. Phase II – Text extraction and Segmentation

After collecting the original PDF documents, the next step of the data preparation process involved converting those 634 PDF files into plain text files. This step is intended to transform the SDS files into a machine-readable format that can be processed with different models, in this case, plain text to create the corpora of the project.

The extraction of textual content from the PDF files was performed using the *PyMuPDF* library (formerly *fitz*), a lightweight and efficient tool for reading and parsing PDF documents in Python. This library was chosen for its ability to accurately extract the embedded text layer of a document while preserving the reading order as much as possible.

During this phase, no Optical Character Recognition (OCR) was applied. All extracted files contained an embedded text layer, and so OCR was unnecessary. Reconstructing tables or improving layout fidelity would also require a separate, specialized workflow, that would need to be adjusted for each SDS provider since each one has a few variations in their SDS. Being this the case, it was considered outside the scope of this work. Minor normalization steps were still applied, such as removing excessive whitespace and ensuring consistent encoding, to make the text easier to process in the following stages.

5.3.1. Segmentation

Once the textual content was extracted, each document was divided into its 16 standardized SDS sections, as defined by international regulations. Each SDS follows this structure consistently, with specific information always located within the same numbered section. Dividing each SDS into its sections allowed the models to focus on the precise part of the document where a given property could be found, improving both efficiency and contextual relevance, especially important in models with smaller context windows.

The identification of sections was performed using a regular expression specifically designed for the Portuguese SDS format:

$$SE(?:C)?\,\text{ÇÃO}\ \backslash d+\backslash s*:.*?\,(?=SE(?:C)?\,\text{ÇÃO}\ \backslash d+\backslash s*:|\$)$$

This expression searches for each section header beginning with the word “SEÇÃO” or “SECÇÃO”, followed by the corresponding section number and its textual content.

Table 3 shows the breakdown of its components.

Table 3: Breakdown of the RegEx used to split SDS sections

RegEx Section	Description
SE(?:C)?ÇÃO	matches either "SEÇÃO" or "SECÇÃO", capturing both common Portuguese spellings.
\d+	matches one or more digits, identifying the section number.
\s*:	allows for optional spaces before the colon that typically separates the section number from its title.
.*?	captures the text content of the section, using a non-greedy quantifier to avoid consuming text beyond the intended boundary.
(?=SE(?:C)?ÇÃO \d+\s*:|$)	is a lookahead assertion that stops the match either when the next section header begins or when the document ends.

This approach made it possible to automatically extract the text of each section while maintaining a clear correspondence with its section number.

To further verify that only the correct sections were collected, a simple mechanism was created to merge any repeated sections into the same text. For example, some documents had the section header in every page of that section.

Each successful processed document resulted in 16 text sections. These sections were stored into plain text file with the following naming convention.

$$\#\#\#\#\# - nn.txt,$$

where ##### is the numeric identifier of the SDS (e.g., 00001, 00123) and nn is the numeric identifier of the section (e.g., 01, 15).

Only SDS files where all sixteen sections were correctly identified and contained meaningful text were accepted for the next phase. Files that lacked sections, were malformed, or produced empty text segments were discarded. In total, 519 out of the 634 collected SDS documents met these criteria and were successfully converted into segmented text files.

5.4. Phase III – JSON Construction

After segmenting the text files, the final phase of the data preparation process consisted of combining those segments with the product and metadata information to create a structured and model-ready dataset. The goal of this phase was to produce a flexible JSON-based corpus capable of serving as input for multiple Natural Language Processing approaches, including rule-based methods, transformer models, and Large Language Models (LLMs).

This stage integrated three data sources:

1. The **sectioned text files** produced in the previous phase, which contained the textual content of each SDS section;
2. The *products.csv* file, which held the ground-truth values for each field and property;
3. The *metadata.csv* file, which defined the corresponding SDS section, data type, and natural-language question for each property.

Once again, the integration of these elements was implemented in Python using the *pandas* library for data handling and alignment between files. The process ensured that every property from the *products.csv* was correctly linked to the corresponding section text, as defined by the metadata file. The final output was a single, consolidated JSON file containing all the information required for model training and subsequent evaluation.

5.4.1. Structure of the JSON Dataset

The final JSON file is composed by a list of objects. Each entry represented one SDS section and contained the textual context of that section, together with all related questions and their correct answers.

An example of this structure is shown in Listing 1.

```
[
    {
        "documentID": 1,
        "section_num": 2,
        "context": "SECÇÃO 2: Identificação dos perigos\n2.1. (...)"
        "qas": [
            {
                "question": "A substância é considerada perigosa
(Sim/Não)?",
                "answer": true,
                "question_type": "boolean",
                "field": "dangerous"
            },
            {
                "question": "Qual a palavra sinal (Nenhum, Atenção ou
Perigo)?",
                "answer": "Nenhum",
                "question_type": "singlevalue-string",
                "field": "signal_word"
            },
            {
                "question": "Quais as Advertências de perigo (Códigos
H***)?",
                "answer": [
                    "H412 - Nocivo para os organismos aquáticos com efeitos
duradouros."
```

```
                ],
                "question_type": "multivalue-string",
                "field": "select_hs"
            },
            {
                "question": "Quais as Recomendações de prudência (Códigos
P***)?",
                "answer": [
                    "P273 - Evitar a libertação para o ambiente.",
                    "P501 - Eliminar o conteúdo/recipiente de acordo com a
legislação em vigor."
                ],
                "question_type": "multivalue-string",
                "field": "select_safety_adv"
            },
        ]
    },
    (...more items...)
]
```

Listing 1: Final JSON dataset format for NLP models input

Each JSON object includes:

1. **documentID:** The document identifier, corresponding to the SDS number;
2. **section_num:** The section number of this document;
3. **context:** The full text of this section;
4. **qas:** A list of question–answer pairs. Includes the information that is used to train and test the model. It is composed by:
    a. **question:** The natural-language question derived from the metadata. Used in LLMs models;
    b. **answer:** The correct answer obtained from the *products.csv* file, the ground truth;
    c. **question_type:** The question type (e.g., boolean, single-value string, multi-value string);
    d. **field:** The field internal name.

This structure was created to support several experiments. Each entry can be used directly for supervised learning tasks, such as BERT-based classification or question-answering, and can also be adapted for zero-shot evaluation using Large Language Models. Because the JSON format explicitly links every question to its corresponding answer and section context, it becomes straightforward to evaluate model performance on individual SDS fields with precision and consistency.

After the models were trained, the same JSON structure was further enriched by adding a new field, *model_answer*, inside each element of the *qas* list. This field stored the answer predicted by the model for that specific question, allowing the same dataset to be reused seamlessly for evaluation. By keeping the ground truth (*answer*) and the predicted value

(*model_answer*) together within the same structure, the evaluation process became significantly easier to automate.

5.4.2. Automatic Dataset Construction

To streamline the creation of the final dataset, a small automation mechanism was implemented to assemble the JSON structure from the available data sources. This mechanism allowed the dataset to be generated programmatically based on a list of selected target fields, ensuring flexibility across different experiments.

For each property chosen for extraction, the process followed four main steps:

1. **Locate the correct SDS section**: Using the enriched *metadata.csv*, the script identified the section number associated with each field;
2. **Retrieve the corresponding text**: The script loaded the segmented *.txt* file for that section, ensuring that the model would receive only the relevant context;
3. **Obtain the ground-truth value**: The correct answer for each property was extracted directly from the *products.csv* file;
4. **Construct the JSON entry**: All information was combined into the unified JSON schema, which included the section text, the natural-language question, the field identifier, the data type, and the correct answer.

This mechanism, illustrated on Figure 8, allowed for rapid generation of multiple versions of the dataset, each tailored to a specific subset of properties or modelling objectives. For instance, separate JSON files could be created for hazard classification experiments, or multi-label classification tasks. Because the assembly process was fully automated and based on explicit metadata mappings, the resulting datasets were consistent and easy to regenerate.

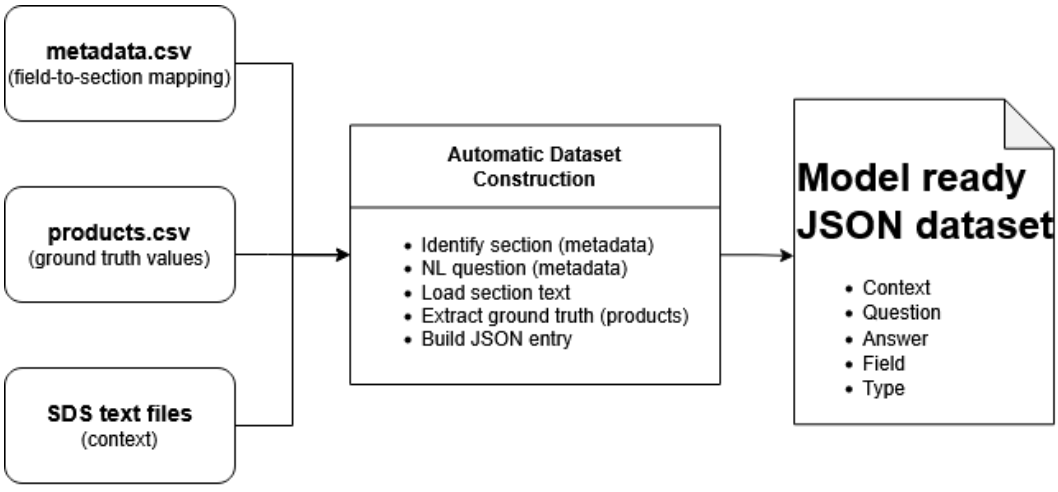

Figure 8: Overview of the process used to construct the model dataset from input files, including metadata, product data, and SDS text.

5.5. SDS fields used in the experiments

Although the data preparation pipeline was designed to support a wide range of Natural Language Processing (NLP) approaches, including Large Language Models (LLMs), the initial experimentation in this work focused on more constrained and controllable models. At the time this project was started, the LLMs that could be executed locally presented significant limitations in terms of performance, reasoning capability, and context handling. These constraints made it difficult to rely on them for systematic experimentation and objective evaluation, particularly when dealing with long technical documents such as Safety Data Sheets (SDS), even when divided by sections.

For this reason, the first modeling experiments were conducted using **BERT-based classifiers**, which offered a stable and well-established baseline for text classification tasks. BERT models allow supervised training with clearly defined inputs and outputs, making them well suited for evaluating the effectiveness of the prepared dataset and for validating the overall data preparation pipeline.

Since a BERT classifier can only be trained to predict **one target variable at a time**, it was necessary to select a limited subset of properties from the full dataset. The selection focused on variables that met three main criteria:

I. clear and unambiguous ground-truth labels in the *products.csv* file,
II. consistent location within the SDS structure (section), and
III. representation of different classification problem types.

Based on these criteria, three properties were selected for initial experimentation:

- **dangerous:** a boolean variable indicating whether the product described in the SDS is considered dangerous;
- **signal_word:** a single-value categorical variable representing the signal word associated with the product (e.g., "*Nenhum*", "*Atenção*", or "*Perigo*");
- **pictogram:** a multi-value categorical variable representing one or more hazard pictograms associated with the product.

Together, these three variables enabled the evaluation of BERT-based models across different classification paradigms: **binary classification**, **single-label multiclass classification**, and **multi-label classification**. This made it possible to assess how well the prepared dataset and section-based text segmentation supported increasingly complex extraction tasks.

In addition to these learned models, a **rule-based extraction** approach was also applied to hazard statements. Hazard codes follow a standardized and highly regular naming convention (e.g., H412, H413), accompanied by a textual description that may vary. Because there are several hundred possible hazard codes and their format is strictly defined, training a BERT classifier for this task was impractical. Instead, a **regular-expression-based method** was employed to identify and extract hazard codes directly from the text. This

approach proved more appropriate for this specific type of information and served as a complementary baseline to the machine-learning models.

5.5.1. Dangerous

The **dangerous** variable represents a binary classification task indicating whether the product described in a Safety Data Sheet is considered dangerous or not. This property is one of the most fundamental indicators in an SDS and plays a central role in hazard communication and regulatory compliance.

The distribution of this variable in the final dataset is shown in Table 4. While both classes are well represented, the dataset is not perfectly balanced, with a clear predominance of products classified as dangerous, expected in these kinds of documents.

Table 4: Class distribution of the binary "dangerous" variable

dangerous	Count	Percentage
False	140	26.97%
True	379	73.03%

This imbalance reflects the nature of the dataset, which focuses on industrial and chemical products where hazardous classifications are common. Despite the imbalance, the presence of a substantial number of non-dangerous samples allows the task to be framed as a realistic binary classification problem. The clarity of the label definition and the consistent location of the relevant information within the SDS, where is always on Section 2, make this variable a suitable starting point for evaluating supervised classification models.

5.5.2. Signal Word

The **signal_word** variable corresponds to a single-label categorical classification task. Signal words are standardized indicators used to convey the severity of hazards associated with a product, typically taking values such as "Perigo" (Danger), "Atenção" (Warning), or "Nenhum" (None). Table 5 presents the distribution of signal words across the dataset.

Table 5: Distribution of the "signal_word" variable in the dataset

signal_word	Count	Percentage
None (*Nenhum*)	168	32.37%
Warning (*Atenção*)	97	18.69%
Danger (*Perigo*)	254	48.94%

The distribution shows that *"Perigo"* is the most frequent class, followed by *"Nenhum"*, while *"Atenção"* is less represented. This uneven distribution introduces, again, class imbalance.

5.5.3. Pictogram

The **pictogram** variable represents a multi-label classification task, as each product may be associated with zero or more hazard pictograms. It is defined by an explicit image in the Safety Data Sheet. However, the present pipeline does not process any images. Instead, classification must rely exclusively on the textual content. Pictograms inferred either from explicit textual references or indirectly through hazard descriptions and classifications contained in the SDS. Table 6 summarizes the number of pictograms associated with each document.

Table 6: Distribution of the number of pictograms per document

Number of Pictograms	Count	Percentage
0	168	32.37%
1	154	29.67%
2	115	22.16%
3	48	9.25%
4	27	5.20%
5	7	1.35%

Most SDS documents contain between zero and two pictograms, while cases with four or five pictograms are comparatively rare. The distribution of pictograms by type is on Table 7.

Table 7: Distribution of pictogram types in the dataset

pictogram	Count	SDS percentage with pictogram
GHS07 - Health Hazard	214	41.23%
GHS05 – Corrosive	140	26.97%
GHS02 – Flammable	114	21.97%
GHS08 - Serious Health Hazard	93	17.92%
GHS09 – Hazardous to the Environment	62	11.95%
GHS06 – Toxic	26	5.01%
GHS03 – Oxidizing	14	2.70%
GHS04 - Compressed Gas	8	1.54%

This variable represents a more complex challenge than the previous ones. It requires the model to identify multiple hazard indicators from text alone. The data is also unbalanced and each product may have none, or several hazards.

5.5.4. H codes (RegEx methods)

In addition to the supervised classification tasks, a rule-based approach was implemented to extract standardized safety codes present in Safety Data Sheets, namely **hazard statements (H-codes)**, **precautionary statements (P-codes)**, and **supplemental hazard statements (EUH-codes)**. Under the EU CLP regulation [39], these codes follow a fixed and highly regular naming convention with a letter prefix (H, P, or EUH) followed by digits. This code is paired with a standardized textual description. The European Commission provides extensive lists of these phrases, where the code functions as a stable identifier across languages and documents.

Examples of these statements include:

- H412 – Harmful to aquatic life with long lasting effects (*Nocivo para os organismos aquáticos com efeitos duradouros*);
- P273 – Avoid release to the environment (*Evitar a libertação para o ambiente*);
- EUH212 – Warning! Hazardous respirable dust may be formed when used. Do not breathe dust (*Atenção! Podem formar-se poeiras inaláveis perigosas ao pulverizar. Não respirar as poeiras*).

These properties were treated differently from the other targets (e.g., **dangerous**, **signal_word**, **pictogram**) for two main reasons.

I. The number of possible codes is very large (hundreds of distinct H-, P-, and EUH-phrases), and many appear only rarely, which makes supervised training of a classifier impractical in this context;
II. Because the code format is standardized and visually distinctive in text, it can be reliably identified using simple pattern-based rules, making regular expressions a natural and efficient solution.

For the purpose of rule-based extraction and downstream modeling consistency, only the code identifier itself was retained in data preparation pipeline, rather than the full phrase. For example, the statement:

- "P273 – Avoid release to the environment."

Was reduced to:

- "P273"

This choice ensures that extraction is independent of variations in punctuation, language, or phrasing, since the descriptive portion may differ across SDS while the code remains constant. Additionally, extracted codes were normalized to a consistent format (e.g., lowercasing to p273) to simplify evaluation later.

Some SDS documents also contain combined codes (for example, multiple H-codes written together that form a unique code). These cases introduce additional complexity, particularly

when codes appear concatenated or grouped, and they are addressed in more detail later in the modeling and evaluation chapter.

5.6. Summary

This chapter describes the data preparation process required to transform raw Safety Data Sheets into a structured and model-ready dataset. Starting from the original PDF documents, the text was extracted, segmented according to the standardized SDS sections, and organized to ensure that relevant context could be associated with each target field.

A key component of this process was the automatic construction of the final JSON dataset, which combined SDS sections text, metadata mappings, natural language questions, and ground-truth values into a unified structure. This approach enabled consistent dataset generation across different experiments and simplified the integration with the models training and evaluation pipeline.

The resulting dataset provides a flexible and reusable input for the experiments presented in the following chapters, where different extraction approaches are applied and evaluated.

6. BERT and RegEx Models and Evaluation

With the dataset fully prepared and structured, the next step of this work focuses on the application and evaluation of different methods for automatically extracting and classifying information from Safety Data Sheets. This chapter presents the modeling approaches adopted in this project, focusing on BERT-based classification models and rule-based extraction methods using regular expressions.

The experiments in this chapter focus on supervised classification tasks and rule-based extraction. BERT models are applied to a selected set of properties with well-defined labels, allowing a controlled evaluation of binary, single-label multiclass, and multi-label classification scenarios for the properties introduced in Section 5.5. Regular expressions are also used to extract standardized safety codes, where the structured and highly regular nature of the data makes rule-based methods more appropriate than supervised learning.

This chapter describes the experimental environment, the dataset splitting strategy, and the modeling and evaluation procedures for each selected property. The results obtained with these methods are then discussed, providing a basis for assessing their suitability and limitations.

6.1. Environment

All experiments presented in this chapter were conducted on a local workstation equipped with an NVIDIA RTX 3080 GPU with 10 GB of VRAM. This hardware configuration provided sufficient computational capacity for fine-tuning transformer-based models like BERT but also limiting the use of more advance models like LLMs.

To ensure reproducibility and portability, all experiments were executed inside a Docker container-based environment with GPU support. The environment was built on top of the official *tensorflow/tensorflow:2.18.0-gpu-jupyter* image, which includes native CUDA support and an integrated *Jupyter Lab* interface. This way was possible to maintain a consistent execution environment independent of the host operating system. Running the experiments in a container also made it possible to access the environment remotely, allowing development and evaluation from different machines without changes to the setup.

Although the base image relied on TensorFlow, the environment was configured to support both *TensorFlow* and *PyTorch* workflows. This was necessary due to the use of the *Hugging Face* Transformers library and related tooling, which internally relies on *PyTorch* for several components. All additional dependencies required for data processing, modeling, and evaluation, like libraries for transformer models, PDF parsing, data manipulation, and visualization, were installed through a project-specific *requirements.txt* file, loaded into the base docker container at creation.

This environment provided a controlled and reproducible setup for model training and evaluation, while remaining representative of the computational limitations encountered in local and small-scale deployments.

6.2. Datasets

All models discussed in this chapter were conducted using the same base dataset, i.e., the JSON structure described in the previous chapter. This dataset contained the section text of each Safety Data Sheet together with the corresponding questions and ground-truth answers for each of the target properties.

For each classification task, only the information relevant to that task was extracted from the base dataset. In the case of the BERT classifiers, this consisted of the textual content of the appropriate SDS section paired with the corresponding target property. For instance, binary classification models used the section text as input and a boolean target as output, while single-label and multi-label classification models relied on the same textual input paired with their respective categorical labels. Where required by the learning algorithms, target labels were encoded into numerical representations (e.g., boolean values mapped to binary labels 1 and 0).

The dataset was randomly split into training, validation, and test subsets using a fixed random seed for the initial shuffling. The same split was reused across all experiments and model variants. The split proportions were defined as follows:

- 80% for training;
- 10% for validation;
- 10% for testing.

This resulted in 415 samples in the training set, 52 samples in the validation set, and 52 samples in the test set, as shown on Table 8. The randomization process was performed once, before the split, using a fixed seed (*random.seed(123)*), and the resulting partitions were preserved and reused for all models and tasks.

Table 8: Distribution of dataset samples across training, validation, and test subsets

Subset	Percentage	Items
Train	80%	415
Validation	10%	52
Test	10%	52

It is important to note that the original dataset, as described in Chapter 5, had class imbalance in several of the selected properties. The dataset split was performed randomly and was not stratified by class label. As a result, the class distributions in the training, validation, and test sets may not perfectly reflect the overall distribution of the original dataset.

The test dataset was reserved exclusively for final evaluation and comparison between models and was not used during training or fine-tuning. This ensured that performance comparisons between different BERT models were conducted under identical conditions, using the same unseen data.

By maintaining a common dataset split and reusing the same base JSON structure across all experiments, this setup allowed differences in performance to be attributed primarily to the models and techniques choices rather than to variations in the underlying data.

6.3. Models used

The experiments in this chapter were conducted using two variants of the BERT model: *bert-base-uncased* [40] and *bert-base-multilingual-cased* [41], both available through the Hugging Face platform [42]. These models were selected to evaluate the impact of language-specific versus multilingual pre-training in the context of Portuguese Safety Data Sheets.

The *bert-base-uncased* model is a monolingual English model, trained on large English corpora. Despite being designed for English text, it was included in this work as a baseline to assess how a general-purpose model performs when applied to a different language without explicit multilingual support.

In contrast, the *bert-base-multilingual-cased* model is trained on multiple languages, including Portuguese. This makes it more suitable for the dataset used in this work and allows for a direct comparison with the monolingual model, providing insight into the importance of language alignment between the training data and the target domain.

Both models were fine-tuned for the specific classification tasks described in the following sections, including binary, multiclass, and multi-label problems. By using these two variants, it was possible to evaluate not only the effectiveness of BERT-based approaches for SDS information extraction, but also the impact of multilingual capabilities on model performance.

Unlike the LLM-based approach presented in Chapter 7, these models operate as task-specific classifiers, requiring dedicated training for each target field. This distinction is essential for the comparison between the two approaches explored in this work.

6.4. BERT Binary – Dangerous

The **dangerous** property represents a binary classification task indicating whether the product described in a Safety Data Sheet is classified as dangerous. This property was selected as one of the simplest targets in the dataset, making it suitable for an initial evaluation of supervised classification approaches.

For this task, the input to the model consists of the textual content of Section 2 (Hazard Identification) of each Safety Data Sheet. The output is a binary label corresponding to the ground-truth value provided in the dataset, either 1 for true or 0 for false.

6.4.1. Model setup and training

For the binary classification of the dangerous property, both BERT variants *bert-base-uncased* and *bert-base-multilingual-cased* were fine-tuned using the same training configuration. As the objective of this experiment was to evaluate and compare baseline model performance rather than to optimize results through extensive hyperparameter tuning, a single, shared configuration was adopted for both models. This choice also helped limit computational cost and training time while ensuring comparability between model variants.

The most relevant training parameters are summarized in Table 9.

Table 9: Hyperparameters used for BERT model training for the 'dangerous' field

Parameter	Value
Training epochs	30
Batch size	16
Best models metric	Cross-entropy Loss

Besides the 30 maximum epochs, an early stopping mechanism was implemented with a patience of 5 epochs to prevent unnecessary training and reduce the risk of overfitting. Although early stopping was enabled, it was not triggered for either model. Validation performance continued to improve throughout training, or the patience 5 was not reached.

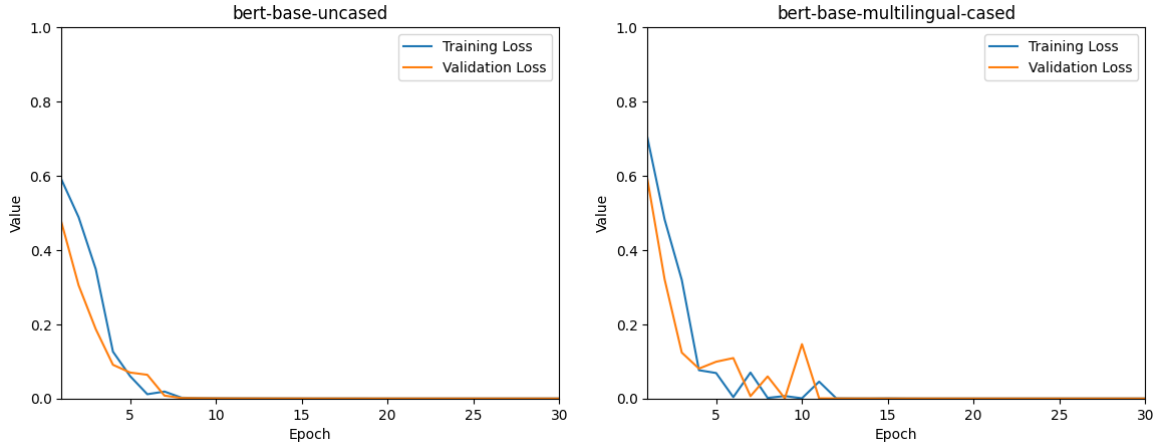

Figure 9: Training and validation loss for the "dangerous" field

During the training phase, both models converged rapidly and achieved near-perfect performance on the training and validation sets (Figure 9). This behaviour suggests that the binary nature of the task, combined with the strong and explicit hazard-related cues present in the input text, makes the dangerous property relatively easy to learn. However, this must be tested in the unseen test dataset.

6.4.2. Evaluation and Results

Both BERT models were evaluated on the held-out test dataset using standard classification metrics, namely *accuracy*, *precision*, *recall*, and *F1-score*. The test dataset was not used during training and so ensures an unbiased estimate of model performance.

Table 10 summarizes the results obtained for both BERT variants.

Table 10: BERT models performance on the "dangerous" field

Model	Accuracy	Precision	Recall	F1-score
BERT base	1.0	1.0	1.0	1.0
BERT multilingual	1.0	1.0	1.0	1.0

Both models achieved perfect performance across all evaluation metrics, correctly classifying every sample in the test dataset. This result indicates that the information required to determine whether a product is classified as dangerous is consistently and explicitly present in the input text used for this task.

The strong performance observed can be attributed to the binary nature of the problem and to the fact that the relevant hazard information is typically stated clearly in the Safety Data Sheets. Under these conditions, both monolingual and multilingual BERT models were able to learn and generalize the decision effectively, with no observable difference between the two variants.

While these results confirm the suitability of BERT-based classifiers for this type of task, they also highlight the limited complexity of the dangerous property when treated in isolation. More challenging scenarios, involving a larger number of classes, implicit information, or multiple simultaneous labels, are explored in the following sections.

6.5. BERT Multiclass Single-label – Signal Word

The **signal_word** property represents a **single-label multiclass classification** task, indicating the signal word associated with a Safety Data Sheet. Signal words are standardized indicators defined by the CLP regulation and are used to communicate the overall severity of the hazards associated with a product. In the dataset, this property can take one of three values: "Perigo" (Danger), "Atenção" (Warning), or "Nenhum" (None).

As in the previous experiment, the input to the model consists of the textual content of Section 2 (Hazard Identification) of each Safety Data Sheet. The task is formulated as a single-label classification problem, where the model predicts a categorical label corresponding to the signal word, which is then compared against the ground-truth label during training and evaluation.

Both BERT variants introduced earlier, *bert-base-uncased* and *bert-base-multilingual-cased*, were trained and evaluated on this task using the same dataset split.

6.5.1. Model setup and training

For the **signal_word** classification task, the two BERT variants were fine-tuned using the same training procedure. The overall configuration followed the same baseline approach used in the binary experiment, with limited hyperparameter tuning.

The most relevant training parameters are summarized in Table 11.

Table 11: Hyperparameters used for BERT model training for the 'signal_word' field

Parameter	Value
Training epochs	30
Batch size	16
Best models metric	Cross-entropy Loss

Training was performed with evaluation at the end of each epoch and automatic selection of the best checkpoint according to validation loss. An early stopping mechanism with a patience of five epochs was enabled again, meaning that training is stopped if validation loss did not improve for five consecutive evaluations/epochs. Unlike the binary task, early stopping was triggered for both models in this experiment, indicating that the models reached a point where additional training no longer resulted in validation improvements. The final model was also reverted to the best epoch, 12th on the *bert-base-uncased*, and 11th on the *bert-base-multilingual-cased*.

Figure 10: Training and validation loss for the "signal_word" field

During training, both models showed a rapid decrease in training and validation loss during the initial epochs, indicating effective learning of the task. However, their convergence behaviour differed noticeably, as illustrated in the training and validation loss curves (Figure 10).

For *bert-base-uncased*, validation loss decreased steadily during the first few epochs but began to fluctuate after reaching its minimum. While training loss continued to decrease throughout the training, validation loss increased in later epochs, suggesting that the model was beginning to overfit to the training data. This triggered the early stopping mechanism, preventing further degradation of the model.

In contrast, *bert-base-multilingual-cased* demonstrated a more stable and consistent training process. Both training and validation loss decreased smoothly and remained closely aligned throughout most of the training, with validation loss reaching lower values before early

stopping was also activated. The smaller gap between training and validation loss indicates better generalization during training and suggests that the multilingual model was able to capture the relevant patterns of the task more effectively.

While these training dynamics provide useful insight into model convergence and generalization behaviour, final conclusions regarding model performance must be based on evaluation using the unseen test dataset, which is presented in the following section.

6.5.2. Evaluation and Results

Both BERT models were evaluated on the test dataset using accuracy, precision, recall, and F1-score metrics (weighted). The results are shown on Table 12.

Table 12: BERT models performance on the "signal_word" field

Model	Accuracy	Precision	Recall	F1-score
BERT base	0.67	0.75	0.67	0.66
BERT multilingual	0.87	0.89	0.87	0.86

Although the multilingual BERT model clearly outperformed the uncased variant across all evaluation metrics, the results indicate that it was not a trivial task and far from perfectly solved. An accuracy of approximately 0.87 and a F1-score of 0.86, while substantially higher than those achieved by the uncased model, still reflect a non-negligible number of misclassifications in the test dataset.

The confusion matrices show more information about the errors of each model. Bert-base-uncased (Figure 11), shows a very high degree of confusion between the classes "Atenção" and "Perigo". This behaviour suggests difficulty in distinguishing between different hazard severity levels when they are expressed using similar linguistic patterns.

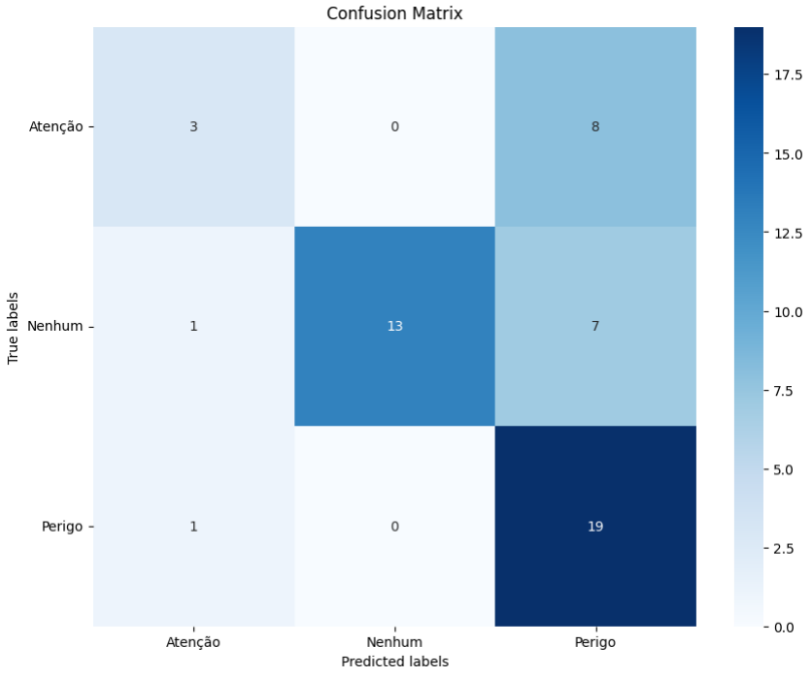

Figure 11: Confusion Matrix for the bert-base-uncased in the "signal_word" field

As for the multilingual model, bert-base-multilingual-cased (Figure 12), shows a much clearer diagonal, indicating more consistent predictions. However, misclassifications are still present. Notably, confusion between the "Nenhum" and "Perigo" classes is the main source of errors. Although these classes represent opposite ends of the hazard severity spectrum and would be expected to be easier to separate, this distinction proved challenging for both models. While the multilingual model reduces the number of such errors, it does not eliminate them entirely.

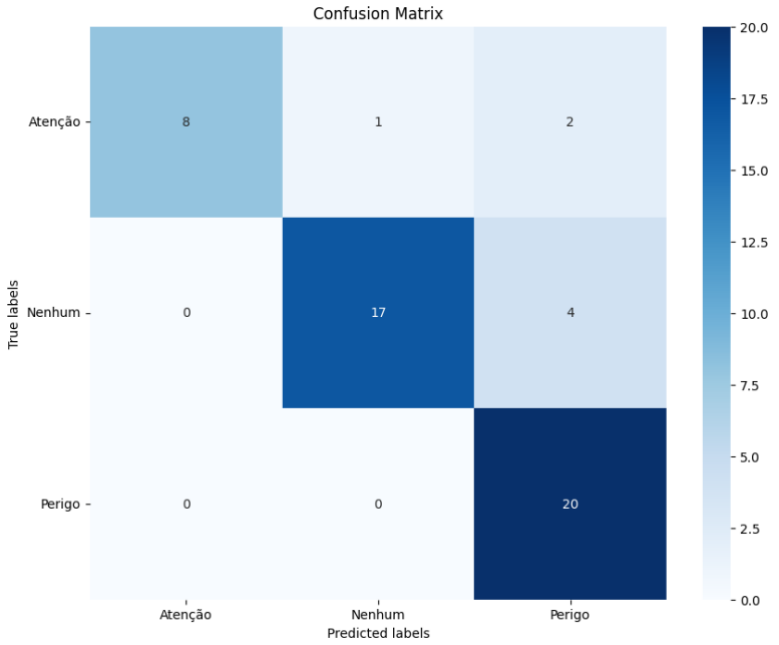

Figure 12: Confusion Matrix for the bert-base-multilingual-cased in the "signal_word" field

Both models achieved strong performance in detecting the Perigo classification, with the multilingual model obtaining a perfect score on the test dataset for this class. The class "Perigo" is typically associated with explicit and standardized language in Safety Data Sheets, often accompanied by clear hazard statements and strong regulatory phrasing.

These results highlight that, despite the improvement achieved by the multilingual BERT model, single-label multiclass classification of signal words remains a challenging task when based only on text context. The overlap in language used across different hazard categories and the relatively small size of the dataset likely contribute to these limitations.

6.6. BERT Multiclass Multi-label – Pictograms

The pictogram property represents a multi-label classification task, as a single Safety Data Sheet product may be associated with multiple hazard pictograms. These pictograms are part of the Globally Harmonized System (GHS) and are typically presented as visual symbols intended to convey different types of chemical hazards.

In this work, however, the classification of pictograms is performed without access to the pictogram images themselves. The data preparation pipeline extracts and processes only the textual content of the Safety Data Sheets. As a result, models must infer the presence of one or more pictograms based solely on textual information, either through explicit references to hazard classifications in the text, or through implicit cues contained in hazard descriptions and regulatory statements.

As in the previous classification tasks, the input to the models consists of the text content of Section 2 (Hazard Identification) of each Safety Data Sheet. The output is a set of one or more labels corresponding to the pictograms associated with the product. The result values are also encoded to facilitate evaluation later.

Both BERT models, base and multilingual, were trained and evaluated for this task and following the same protocols.

6.6.1. Model setup and training

For the multi-label classification of hazard pictograms, both BERT variants were fine-tuned using the same training configuration. As in the previous experiments, a shared setup was adopted to ensure comparability between models and to avoid extensive hyperparameter tuning.

The most relevant training parameters are summarized in Table 13.

Table 13: Hyperparameters used for BERT model training for the 'pictograms' field

Parameter	Value
Training epochs	50
Batch size	16
Best models metric	Cross-entropy Loss

Training was performed with evaluation at the end of each epoch and automatic checkpoint selection based on validation loss. An early stopping mechanism with a patience of five epochs was enabled to interrupt training once validation performance stopped improving.

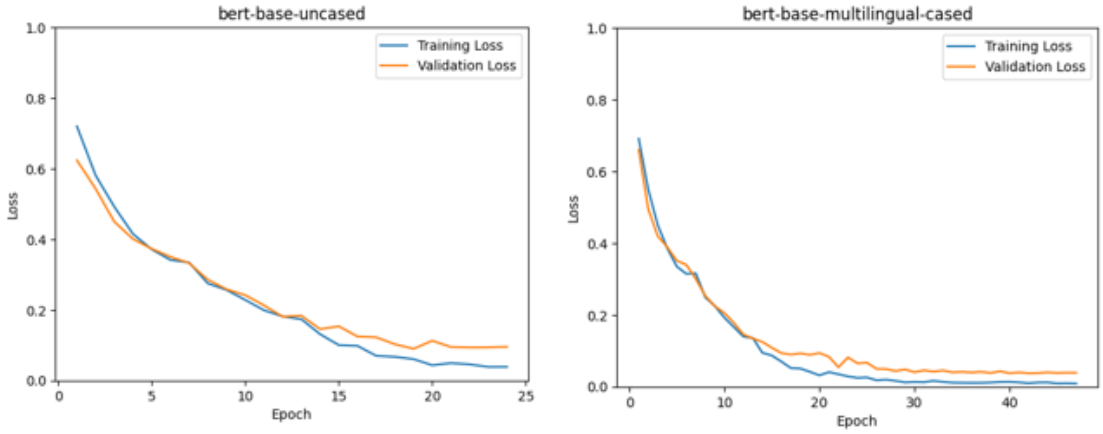

Figure 13: Training and validation loss for the "pictograms" field

The training and validation loss curves for both models are shown in Figure 13. In both cases, loss decreases steadily during the initial epochs and then reaches a plateau. For **bert-base-uncased** (Figure 13 - left), this stabilization occurs earlier, and early stopping is triggered at epoch 24.

The multilingual model (Figure 13 - right) follows a similar trend but over a longer training period. Although early stopping is only triggered at epoch 47, validation loss remains nearly flat from approximately epoch 30 onward, indicating that the model had already converged. The final performance of both models is assessed using the unseen test dataset, which is discussed in the following section.

6.6.2. Evaluation and Results

Both BERT models were evaluated on the unseen test dataset using weighted accuracy, precision, recall, and F1-score, accounting for class imbalance across pictogram labels. The results are shown on Table 14.

Table 14: BERT models performance on the "pictograms" field

Model	Accuracy	Precision	Recall	F1-score
BERT base	0.93	0.75	0.85	0.79
BERT multilingual	0.94	0.79	0.91	0.84

Just like in the previous task, the multilingual BERT model outperformed the base variant across all evaluation metrics. The improvement is most noticeable in recall and F1-score, indicating that the multilingual model was more effective at identifying multiple relevant pictogram labels from the text.

Although the multilingual BERT model outperformed again the uncased variant, the results show that the pictogram classification task is still challenging. While overall accuracy is high for both models, the precision and F1-score values indicate that a noticeable number of labels are still predicted incorrectly or missing.

6.7. Hazard Codes – Rule-based Extraction & BERT

Hazard-related codes such as **H*** (hazard statements), **P*** (precautionary statements), and **EUH*** (supplementary hazard information) are standardized identifiers defined by regulatory frameworks and appear explicitly in the text of Safety Data Sheets. As with other properties explored in this work, these codes follow well-defined conventions and are part of a standardized reporting structure.

However, unlike previously studied properties, the set of possible hazard codes is extensive, comprising more than one hundred distinct identifiers, and not all valid codes are necessarily represented in the available dataset. Since the BERT-based approaches used in this chapter operate as closed-set classifiers, they are limited to predicting labels that have been observed during training. This characteristic raises concerns regarding coverage and generalization when dealing with an open-ended and evolving set of standardized codes.

For this reason, a *rule-based extraction* approach using regular expressions was considered a natural first solution. Given the strict and consistent syntax of hazard codes, such an approach allows the extraction of any valid code appearing in the text, including codes that were not present in the dataset. In this task, the descriptive portion associated with each code was intentionally removed, and the focus was placed exclusively on the code identifier itself.

A BERT-based multi-label classifier was also evaluated for this task using the same training, validation, and testing procedure applied in previous tasks. This allowed a direct comparison between rule-based extraction and supervised classification under the same experimental conditions.

6.7.1. Methodology

The extraction of hazard-related codes was performed using a rule-based approach, consisting of three main steps:

1. Normalization;
2. Encoding;
3. Pattern-bases extraction.

First, all hazard codes present in the dataset were normalized to ensure consistency. Each occurrence of a hazard statement was reduced to its identifier only. For example, an entry such as "*H225 – Líquido e vapor facilmente inflamáveis*" was transformed into "*h225*". This normalization step intentionally removes the descriptive text and retains only the standardized code, allowing the task to focus exclusively on identifying which hazard codes are present in the document.

Next, all unique normalized codes appearing in the dataset were collected and encoded using a MultiLabelBinarizer (MLB). This encoding allows each Safety Data Sheet to be represented as a binary vector indicating the presence or absence of each hazard code, enabling direct comparison between extracted codes and ground-truth labels for evaluation.

Hazard codes (H codes) were then extracted directly from the textual content of the relevant SDS sections using a regular expression designed to match standardized code patterns.

$$\backslash bH\backslash d\{3\}[a-zA-Z]*(?:\backslash s*\backslash +\backslash s*H\backslash d\{3\}[a-zA-Z]*)*\backslash b$$

This pattern captures single hazard codes (e.g., H225) as well as composite forms where multiple codes are combined (e.g., H300+H310). All extracted matches were subsequently normalized to a consistent format. For the purposes of this experiment, the resulting set of codes was mapped to a binary representation using the previously fitted MultiLabelBinarizer (MLB), enabling evaluation against the ground-truth labels.

It should be noted that the encoding step is not intrinsic to the extraction process itself and was introduced solely to support experimental evaluation. In a practical deployment scenario, the objective would simply be to extract and return the hazard codes present in the text, without any need for encoding. The rule-based extraction method itself does not rely on training or tuning and applies deterministic rules to the input text, allowing it to generalize naturally to any valid hazard code that conforms to the defined syntactic pattern, regardless of whether it appears in the original dataset or not.

6.7.2. Evaluation and Results

Although the rule-based extraction method does not involve training or model selection, its performance was evaluated on the test dataset to ensure consistency with the evaluation protocol used in the previous classification tasks. The same metrics were calculated: accuracy, precision, recall, and F1-score.

Table 15: Rule-based approach performance on the "hazard codes" field

Metric	Score
Accuracy	0.98
Precision	0.9
Recall	1.0
F1-Score	0.94

The results in Table 15 show that the rule-based approach achieves perfect recall, meaning that all hazard codes present in the test dataset were successfully found and extracted. Precision is also high, although not perfect, indicating the presence of some false positive predictions in the extracted hazard codes. Overall, the high F1-score reflects a strong balance between completeness and correctness.

Just for comparison, the same task was also evaluated using a BERT-based multi-label classifier (only the multilingual variant), following the same train/validation/test split and

evaluation procedure applied in previous sections and using the MLB fitted to the entire dataset. The results obtained are shown in Table 16.

Table 16: Performance comparison between the BERT model and Rule-based method

Method	Accuracy	Precision	Recall	F1-score
BERT (multilingual)	0.94	0.56	0.41	0.45
Rule-based (RegEx)	0.98	0.90	1.00	0.94

While the BERT-based classifier achieved relatively high accuracy, its precision and recall were substantially lower, indicating difficulty in reliably identifying the correct set of hazard codes. This outcome was expected, as the task was framed as a classification problem with many possible labels, many of which are sparsely represented or absent from the dataset. Under these conditions, a classifier may not be able to generalize.

In contrast, the rule-based approach benefits directly from the strict and standardized syntax of hazard codes and is not constrained by the number of classes. Any valid code conforming to the defined pattern can be extracted, including codes not present in the available data. These results confirm that, for this specific task, rule-based extraction aligns more naturally with the structure of the problem and provides a more reliable solution than supervised classification.

6.8. Summary

This chapter explores different approaches for extracting structured information from Safety Data Sheets using BERT-based classifiers and rule-based methods applied to a small, yet representative, subset of properties. Rather than adopting a single generalized solution, each property required a tailored approach, either through dedicated supervised training or through the development of custom extraction rules.

Across the supervised learning experiments, the multilingual BERT model consistently outperformed the monolingual variant, reflecting its broader linguistic coverage and better alignment with the Portuguese SDS texts used in this work. While the classifiers achieved reasonable performance for several properties, they also exhibited some limitations in more advanced tasks. Even when results were strong, each model required dedicated preparation, training, and evaluation tailored to a single property.

For hazard code extraction, a rule-based approach proved effective due to the strict and standardized syntax of these codes. However, this approach also required a carefully designed regular expression specific to this task and relied on discarding descriptive information to focus exclusively on code identifiers. While reliable in this context, such rule-based methods are not easily transferable to most other SDS properties, which lack comparable structural regularity.

A key limitation common to all approaches presented in this chapter is scalability. Each method, either learning-based or rule-based, was designed to extract a single property at a

time. Given that the dataset contains more than one hundred distinct fields, extending this strategy would require substantial additional effort, making it impractical for broader SDS information extraction.

These limitations motivate the next stage of this work, which explores the use of large language models (LLMs) as a more flexible alternative. The following chapter presents a set of exploratory experiments using a locally deployed LLaMA-based model to extract multiple properties simultaneously from SDS text. Due to practical constraints, this experiment is limited in scope but aims to assess whether a single model can reduce per-property engineering effort while maintaining acceptable extraction quality.

7. LLM Models and Evaluation

The previous chapter explores the use of BERT-based classifiers and rule-based methods for extracting structured information from Safety Data Sheets. While these approaches showed promising results for individual properties, they required property-specific configurations, either through dedicated model training or through custom rule design. This limitation raises concerns regarding scalability when dealing with a large number of SDS fields.

To address this, this chapter investigates an alternative approach based on *Large Language Models* (LLMs). Instead of training or designing a separate solution for each field, a single model is used to process the document text and answer multiple questions, each corresponding to a specific field of the SDS. The goal is to evaluate whether this approach can provide a more flexible and scalable solution for information extraction.

The experiments presented in this chapter were conducted using the *LLaMA 3.3 70B* model. Due to the computational requirements of this model, the evaluation was limited in scope and performed as a set of exploratory experiments rather than a fully optimized pipeline. These experiments aim to provide insight into the potential of LLMs to reduce per-field engineering effort while maintaining an acceptable performance.

7.1. Environment

The experiments in this chapter were conducted on a dedicated machine equipped with an NVIDIA A100 GPU, which enabled the execution of large-scale models that could not be run on local hardware.

As in the previous setup, a containerized environment using Docker was used. A Jupyter Server container was again deployed for developing and executing the Python scripts responsible for data processing, prompt generation, and evaluation of results.

A new Ollama container was used to run the LLM model and expose it as a local service. The Jupyter environment interacted with this service by sending prompts to the model and receiving the generated responses as part of the automated processing pipeline.

Another container running Open WebUI was deployed to provide a chat-based interface for manual interaction with the LLM model. This interface was also connected to the same Ollama service, allowing for quick testing and refinement of prompts before executing the full pipeline over the dataset.

Within this setup (Figure 14), the Jupyter environment executed the main data pipeline, while Open WebUI provided a convenient way to validate prompt behaviour and validate model outputs interactively, particularly important during the prompt design and refinement phase.

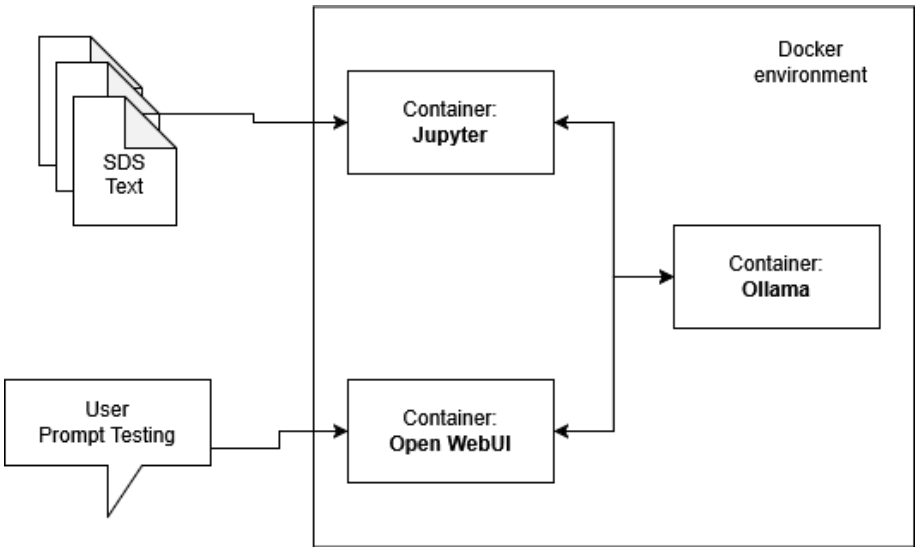

Figure 14: Containerized architecture for LLM-based processing, including Jupyter, Ollama, and Open WebUI components

7.2. Model Description

The experiments in this chapter were conducted using the LLaMA 3.3 70B Instruct model, a large-scale transformer-based language model developed by Meta and made available through the Hugging Face platform [45]. This model belongs to the LLaMA family of open-weight models and is specifically fine-tuned for instruction following tasks, enabling it to generate structured and context-aware responses based on natural language prompts.

With approximately 70 billion parameters, this model is significantly larger than any model used in previous chapters. Due to its size and computational requirements, it required access to a much more capable dedicated GPU, such as the A100 in the environment described in the previous section. Despite these requirements, it remains a practical option in environments where reasonably high-performance computing resources are available.

Unlike the BERT-based models used earlier in this work, which operate as task-specific classifiers, LLaMA is a generative model capable of producing free-form text conditioned with a given prompt. This allows it to be applied in a zero-shot or few-shot settings, where the model can perform tasks without explicit training on the target dataset, relying instead on its pre-training and instruction tuning.

This different strategy enables a more flexible approach to information extraction. Rather than training a separate model for each property, a single LLM can be prompted to answer multiple questions about the same input text. In this work, this capability is explored to assess whether a generative, prompt-based approach can reduce the need for per-field model design while still producing satisfactory results.

7.3. Dataset

The experiments in this chapter used the same base dataset described in Chapter 5, consisting of 519 Safety Data Sheets processed into structured JSON format. It considered the same set of fields used in the previous experiments, allowing a direct comparison between the LLM-based approach and the BERT-based classifiers.

Each entry in the dataset contains the relevant section text, associated metadata, and the corresponding ground-truth value for each field. While the ground-truth values were included in the dataset, they were not provided to the model during inference and were used exclusively for evaluation purposes later.

Unlike the previous experiments, where each model was trained to predict a single field, the LLM-based approach was designed to extract multiple fields from the same document. For each SDS, the model was prompted sequentially with a set of natural language questions, each corresponding to a specific field. The input provided to the model consisted of the text of the relevant SDS section, ensuring consistency with the context used in earlier experiments.

Since the model was used in a zero-shot setting, no training or fine-tuning was performed. As a result, evaluation was conducted in two ways. First, the model was applied to the full dataset of 519 documents to assess overall performance. Second, results were also computed on the previously defined test subset of 52 documents, enabling a direct comparison with the BERT-based models presented in Chapter 6.

7.4. Prompt design

The behaviour and performance of LLMs are highly dependent on prompt design. Unlike the supervised approaches presented in the previous chapter, where model behaviour is learned during training, LLMs rely on structured prompts to guide their responses. As such, defining a clear and consistent prompting strategy is essential to obtain reliable and reproducible results.

In this project, a structured prompting approach was adopted to guide the model to extracting information in a controlled and machine-readable format. The prompt design was divided into two main components: a *system prompt*, responsible for defining the extraction rules and providing the relevant SDS context, and a *question prompt*, used to query each target field individually.

Because of the generative nature of LLMs, the prompts were designed to try to enforce a strict output format and to minimize variability in the generated responses. This also allowed the outputs to be consistently processed and evaluated. The following sections describe each component of the prompting strategy in detail.

7.4.1. System prompt

The system prompt was designed to establish the overall behaviour of the model before any field-specific queries were asked. For each document section, a new interaction session was initialized, in which the system prompt defined the rules, expected output format, and extraction constraints, as well as provided the relevant SDS section text.

The system prompt had three main objectives:

I. Explain the model its objective and the "rules of the game",
II. Ensure that all extracted information was grounded in the provided context, and
III. Standardize responses across different types of fields to allow automatic evaluation, in this case by enforcing a JSON-like structure.

To achieve these objectives, the system prompt was structured into several components. First, it defined the general interaction format between the model and the "user", including how questions would be presented and how answers should be structured. It also included a set of explicit rules stating how the model should extract information from the provided context (Listing 2).

```
**Formato de Pergunta:**
Pergunta: Qual a <entidade>?
Dica:"<dica_da_pergunta>

**Formato de Resposta Esperado:**
{{"<entidade>": ["<valor1>", "<valor2>", ...]}}
```

Listing 2: Question and response format template for LLM extraction

The model was instructed to extract information strictly from the given context, avoiding the inclusion of external knowledge or inferred information. It was encouraged to reproduce expressions as they appear in the source text, with minimal or no modification. This constraint was intended to reduce variability in the answers generated and improve alignment with the ground-truth labels so it could be evaluated.

A fallback behaviour was also defined for cases where the requested information was not present in the context. In such situations, the model was instructed to return an empty value within the expected structure. This ensured that all responses remained consistent and comparable, even when information was missing or could not be found (Listing 3).

```
3. Se a resposta não estiver no contexto, retorne uma lista com uma string
vazia:
    {{"<entidade>": [""]}}
```

Listing 3: Fallback rule for empty responses

To reinforce the expected behaviour, the system prompt included a set of question-and-answer examples, covering both single-value and multi-value fields. These examples illustrated how natural language questions should be mapped to structured responses and helped guide the model towards consistent outputs (Listing 4).

```
**Exemplos de Perguntas e Respostas:**
1. Pergunta:
Pergunta: "Quais as Recomendações de prudência (Códigos P***)?
Dica: "Podem ser múltiplos itens.

1. Resposta:
{{
    "recomendacoes_de_prudencia": [
       "P101 - Se for necessário consultar um médico, mostre-lhe a embalagem
ou o rótulo.",
        "P102 - Manter fora do alcance das crianças."
    ]
}}

2. Pergunta:
Pergunta: Qual a palavra sinal?
Dica: Apenas 0 ou 1 valores.

2. Resposta:
{{"palavra_sinal": ["Atenção"]}}
```

Listing 4: Example question–answer pairs and expected input/output

The relevant SDS section text was embedded directly into the system prompt as contextual information. The model was explicitly instructed not to generate any output immediately, but instead to wait for a question. This allowed the same context to be reused across multiple queries, enabling the sequential extraction of different fields from the same document section (Listing 5).

```
**CONTEXTO:**
{context}

**Atenção:** Não responda nada agora. Apenas aguarde a próxima pergunta.
```

Listing 5: Context injection and response delay instruction

Finally, the model was instructed to respond exclusively in JSON format, following a predefined structure where each field is associated with a list of values. This constraint ensured that the generated outputs could be consistently parsed and integrated into the evaluation pipeline. The full system prompt is on Appendix A.

7.4.2. Question prompt

The field-specific queries were created using natural language questions, which were already defined during the dataset preparation phase. As described in Chapter 5, each field in the dataset is associated with a corresponding question, stored within the JSON structure alongside its metadata and ground-truth value.

For each document, the model was queried sequentially using these predefined questions. Each question was presented in a structured format, consisting of the question itself and an additional short hint indicating the expected type of answer and how many answer items are expected. An example of this structure is shown on Listing 6.

```
Pergunta: A substância é considerada perigosa (Sim/Não)?
Dica: Resposta Sim/Não
```

Listing 6: Example of a field-specific (dangerous) question prompt used for LLM queries

These hints are intended to guide the model towards producing answers consistent with the expected output format. Depending on the field type, the hints indicated whether the answer should be a boolean value, a single value, or multiple values.

Questions were always asked one at a time, following the initialization of the system prompt for each document section. This approach allowed the model to focus on a single field per interaction, simplifying both the response generation and the subsequent parsing of results.

7.4.3. Output Conditioning

To ensure consistent and machine-readable outputs, the model was instructed to respond exclusively in JSON format. Each answer followed a predefined structure, where the field name is associated with its corresponding value. A simplified example of this format is shown in Listing 7:

```
{"property_name": ["value"]}
```

Listing 7: Example of the JSON output format enforced for LLM responses

By enforcing a structured format, the responses could be stored and compared with the ground-truth values without requiring additional post-processing to select the actual answer. Although LMMs can generate free-form natural language responses, constraining the output format can help to reduce variability and improve consistency across different queries and documents.

The model responses were then stored within the same JSON structure used throughout this work, alongside the corresponding metadata, ground-truth values, and questions.

7.4.4. Prompting Procedure

The prompting procedure (Figure 15) followed a structured and repeatable process applied to each document in the dataset. For each SDS document, the relevant section text was first used to construct the system prompt, initializing a new interaction session with the model.

Within this session, the model was queried sequentially using the set of natural language questions associated with the target fields. Each question was submitted individually, allowing the model to focus on a single field at a time while reusing the same contextual information provided in the system prompt.

The responses returned by the model were expected to follow the predefined JSON structure. These responses were parsed and stored as the *model_answer* corresponding to each field within the dataset.

The procedure was repeated for all documents, resulting in a complete dataset containing, for each field, the original context, the natural language question, the ground-truth value, and the model-generated answer.

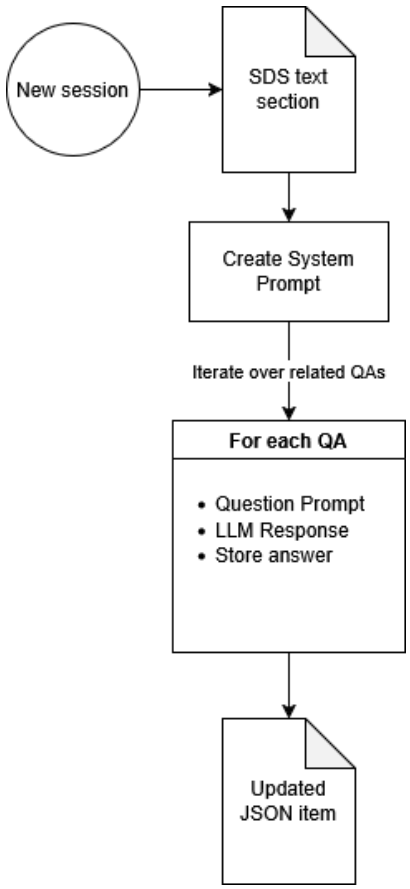

Figure 15: Workflow of the prompting procedure for sequential LLM-based information extraction

7.5. Evaluation and Results

The performance of the LLM-based extraction approach was evaluated using the same metrics applied in the previous chapter: accuracy, precision, recall, and F1-score. This ensured consistency across experiments and enabled direct comparison with the BERT-based and rule-based approaches.

Since the model was used in a zero-shot setting, no training or validation phases were involved. This allowed the model to be evaluated in the entire dataset of 519 documents, providing an overview of the model's performance across all available samples.

However, to enable a direct comparison with the models presented in previous chapter, the results were also computed on the previously defined test subset of 52 documents. This ensured that all approaches could be evaluated under the same conditions using a common reference dataset.

7.5.1. Binary – Dangerous

The performance of the LLM-based approach for the dangerous field is presented in Table 17, alongside the results obtained with the BERT-based models.

Table 17: Performance comparison of LLM and BERT models for the "dangerous" field

Metric	LLaMA 3.3 (total)	LLaMA 3.3 (test)	BERT base	BERT multilingual
Accuracy	0.94	0.98	1.0	1.0
Precision	0.99	1.00	1.0	1.0
Recall	0.92	0.97	1.0	1.0
F1-Score	0.96	0.99	1.0	1.0

The LLM achieved strong performance for the dangerous field, with results close to those obtained by the BERT models. On the test dataset, the model reached near-perfect scores across all metrics, indicating that it is capable of correctly identifying whether a product is classified as dangerous in most cases.

When compared with the BERT-based approaches, the LLM shows slightly lower performance, particularly in terms of recall on the full dataset. However, the differences are small, and the results indicate that binary classification tasks with clear and explicit cues can be effectively handled by a zero-shot LLM without the need for task-specific training.

7.5.2. Multiclass Single-label – Signal Word

The performance of the LLM-based approach for the signal word field is presented in Table 18, alongside the results obtained with the BERT-based models.

Table 18: Performance comparison of LLM and BERT models for the "signal_word" field

Metric	LLaMA 3.3 (total)	LLaMA 3.3 (test)	BERT base	BERT multilingual
Accuracy	0.92	0.90	0.67	0.87
Precision	0.90	0.87	0.75	0.89
Recall	0.91	0.89	0.67	0.87
F1-Score	0.91	0.88	0.66	0.86

The LLM achieved strong performance for this task. Although not perfect, it outperformed the monolingual BERT model by a significant margin and achieved results slightly better than the multilingual BERT model. This indicates that the LLM can capture the distinctions between the signal word classes of each product effectively, even without task-specific training.

These results suggest that the LLM is well-suited for this type of classification task, where the number of possible outputs is limited and the distinctions between classes are relatively clear. In this context, the model was able to exceed the performance of specialized classifiers, while relying only on the prompt context.

7.5.3. Multiclass Multi-label – Pictograms

The performance of the LLM-based approach for the pictograms field is presented in Table 19, alongside the results obtained with the BERT-based model.

Table 19: Performance comparison of LLM and BERT models for the "pictograms" field

Metric	LLaMA 3.3 (total)	LLaMA 3.3 (test)	BERT base	BERT multilingual
Accuracy	0.50	0.52	0.93	0.94
Precision	0.72	0.76	0.75	0.79
Recall	0.77	0.80	0.85	0.91
F1-Score	0.72	0.77	0.79	0.84

The LLM achieved moderate performance for this task, reaching an F1-score of 0.77 on the test dataset. However, these results are lower than those obtained with the BERT-based models, which achieved up to 0.84.

A key factor that may be influencing these results is the nature of pictogram information in SDS documents. In many cases, pictograms are presented as graphical elements, with limited or no explicit textual references. Since the pipeline relies exclusively on text extraction, this information is often not directly available to the model. As a result, the LLM must rely on indirect textual cues, which may be insufficient, especially in a zero-shot setting, whereas the BERT-based models benefit from supervised training to learn implicit associations.

Despite the relatively low accuracy, the LLM achieved higher precision and recall scores, indicating that it often identified relevant pictograms correctly but struggled to produce exact matches across all labels in this multi-label setting.

These results indicate that pictogram extraction is a more challenging task under the current setup and highlight the limitations of relying solely on text. Improvements could be achieved by refining the extraction pipeline or incorporating additional sources of information.

7.5.4. Multiclass Multi-label – Hazard Codes

The performance of the LLM-based approach for the hazard codes field is presented in Table 20, alongside the results obtained with the BERT multilingual and rule-based (RegEx) methods.

Table 20: Performance comparison of LLM and Rule-based methods for the "hazard codes" field

Metric	LLaMA 3.3 (total)	LLaMA 3.3 (test)	BERT multilingual	RegEx
Accuracy	0.59	0.63	0.94	0.98
Precision	0.90	0.92	0.56	0.90
Recall	0.92	0.98	0.41	1.00
F1-Score	0.91	0.95	0.45	0.94

The LLM achieved high precision, recall, and F1-score, indicating that it is generally effective at identifying the hazard codes present in the text. However, the overall accuracy is significantly lower than the other methods. This difference may be partly due to the stricter nature of the accuracy metric in multi-label settings, which requires an exact match between the predicted and ground-truth sets. Additionally, the evaluation process relies on parsing the model-generated outputs into a structured format, and small deviations from the expected format may negatively impact the evaluation and thus the accuracy.

When compared with the other methods, the rule-based approach remains the most reliable solution for this task, achieving near-perfect results due to the structured nature of hazard codes. The BERT-based classifier performs significantly much worse, reflecting the limitations of applying a classification approach to a task with a large and potentially open set of labels. The LLM provides a flexible alternative, achieving strong recall without requiring explicit rules, although with lower exact-match accuracy.

7.6. Summary

Unlike the previous methods, which relied on dedicated models or rules for each individual field, the LLM-based approach enabled the extraction of multiple fields using a single model through prompt-based interaction.

The results demonstrate that LLMs can achieve competitive performance across different types of tasks, particularly for simpler classification problems and fields with clear linguistic patterns. Even within this experimental setup, performance was close to, and in some cases better than, specialized models that require dedicated training to achieve results.

However, the results also highlight that LLM performance is strongly dependent on the availability of relevant textual information. In tasks such as pictogram extraction, where key information is often presented as graphical elements with limited textual representation, the model achieved lower performance compared to BERT-based approaches. This suggests that, under a text-only pipeline, certain types of information will be more challenging to extract.

It is also important to note that the experiments were conducted on Portuguese Safety Data Sheets. While the LLaMA 3.3 model supports multiple languages, including Portuguese, a significant portion of its pre-training data is predominantly in English. As such, this setup represents a more challenging scenario for the model. A similar limitation applies to the BERT-based models used in previous experiments. Despite this, the results obtained remain competitive, suggesting that the LLM can generalize effectively to this multilingual context.

A key advantage of the LLM-based approach is its flexibility and scalability. By relying on natural language prompts rather than task-specific training, the same model can be applied to a wide range of fields with limited additional engineering effort. This represents a significant reduction in development complexity when compared to training and maintaining multiple classifiers or designing custom extraction rules for each field.

However, this flexibility comes with trade-offs. The approach requires significantly more computational resources at inference time, and the quality of the results is sensitive to prompt design and output formatting. The evaluation process, since it does not rely on encoders, introduces an extra layer of complexity, since the model outputs must be parsed and aligned with the expected structure before evaluation with ground-truth data.

Overall, this experiment suggests that LLM-based approaches offer a flexible and scalable solution for SDS information extraction, reducing the need for per-field model development when compared with specialized methods.

8. Conclusion

This work addressed the problem of automating information extraction from Safety Data Sheets, a process that is traditionally manual, time-consuming, and prone to error. The initial research phase played an important role in understanding the structure of SDS documents and the challenges associated with extracting information from semi-structured and domain-specific text, directly informing the design choices adopted throughout the project.

To tackle this challenge, a complete data processing pipeline was developed, transforming the original PDF documents into structured, model-ready datasets. Through text extraction, section segmentation, and automated JSON construction, this pipeline proved to be a key component of the work, enabling consistent dataset generation and supporting all subsequent experiments. It resulted in the creation of a reusable dataset that can support not only the approaches explored in this work, but also other NLP methods applied to SDS information extraction.

Multiple extraction approaches were explored, including rule-based methods, BERT-based classifiers, and a prompt-based Large Language Model. The results show that, within the scope of this work, no single method was optimal for all tasks.

Rule-based extraction proved highly effective for structured patterns, such as hazard codes, achieving strong performance (F1 = 0.94). However, this approach is limited to fields that follow strict and well-defined patterns and requires carefully designed rules for each target field.

BERT-based models, which constituted the main approach explored in this work, achieved very strong results in simpler classification tasks (e.g., F1 = 1.0 for the boolean dangerous field). These models were applied in a classification setting, requiring the definition of all possible output classes in advance and the use of encoding mechanisms, which adds complexity. While effective, this approach requires dedicated supervised training for each target field, making it impractical to scale to the full set of properties present in SDS documents.

It is also important to consider the evolution of NLP models over the duration of this work. At the time the project was initiated, BERT-based approaches represented the most practical and reliable solution, particularly for Portuguese text, as locally deployable LLMs offered limited performance. Early experiments with models such as LLaMA 2 showed poor results, especially when applied to Portuguese text and complex documents such as SDS, which motivated the focus on BERT-based methods as the primary approach.

More recent models, like the LLaMA 3.3 70B, represent a significant improvement in both general performance and multilingual capabilities. Although these models require more computational resources, they remain feasible to run on dedicated hardware and enable more effective LLM-based experiments, as demonstrated in this work. The LLM-based approach,

although explored in a limited and zero-shot setting, demonstrated competitive performance across all different tasks, with results close to or, in some cases, better than the BERT models (e.g., F1 = 0.91 for signal word classification). More importantly, it introduced a significantly higher degree of flexibility, enabling the extraction of multiple fields using a single model. While this approach is computationally more demanding and presents challenges related to prompt design, output consistency, and evaluation complexity, the results of this initial experiment indicate that it is the most promising direction for further development.

At the end, the results highlight a clear trade-off between performance, flexibility, and scalability. Rule-based and classification approaches offered strong performance but required significant per-field effort and lack adaptability, while LLM-based methods, even in this limited experimental setting, achieved comparable results with substantially lower per-field development effort. These findings suggest that, for scenarios involving a large number of target fields, LLM-based approaches provide a more practical and extensible solution.

The objectives defined for this work were achieved. Models capable of extracting and classifying selected SDS fields were developed using different approaches, including rule-based methods and machine learning techniques. These approaches were implemented, adapted to the problem context, and evaluated using appropriate metrics, allowing for a comparative analysis of their performance. This process also enabled a clear assessment of the feasibility and limitations of each method in the context of SDS information extraction.

In conclusion, this work demonstrates that combining a robust and reusable data preparation pipeline with modern NLP techniques enables effective extraction of information from semi-structured documents. The pipeline itself represents a significant contribution, providing a consistent and extensible foundation for experimentation. While traditional methods remain valuable for specific tasks, LLM-based approaches offer a flexible and scalable framework that is well-suited for future developments in SDS information extraction.

8.1. Future work

The results of this work suggest that further development should primarily focus on Large Language Model-based approaches. Although explored in a limited and zero-shot setting, LLMs demonstrated strong potential for scalable information extraction. Future work should therefore explore improvements in prompt design, considering recent best practices, and the use of few-shot prompting or fine-tuning techniques could also significantly improve performance and consistency, enabling general-purpose LLMs to be tailored to specific tasks and application contexts.

In addition, it would also be valuable to evaluate other recent open-weight models from different providers, such as *DeepSeek*, Google's *Gemma*, *Mistral*, and newer versions of the *LLaMA* family. Given the rapid evolution of these models, such comparisons could provide

further insight into the trade-offs between performance, resource requirements, and multilingual capabilities of each.

From a system perspective, the data processing pipeline developed in this work could be extended into a complete end-to-end solution. This would involve adding a storage layer for the extracted information and integrating a simple user interface, for example using tools such as *Streamlit*, to allow interaction with the system. Given the structure already established, these additions would require relatively limited effort.

Finally, a hybrid approach could be explored, combining rule-based methods for highly structured fields with LLM-based extraction for more complex information. While this introduces additional system complexity, it may allow leveraging the strengths of each method in specific scenarios.

Bibliography

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 24, 2019, *arXiv*: arXiv:1810.04805. doi: https://doi.org/10.48550/arXiv.1810.04805.

[2] Vereinte Nationen, Ed., *Globally harmonized system of classification and labelling of chemicals (GHS)*, Eighth revised edition. in United Nations publication. New York Geneva: United Nations, 2019.

[3] "Understanding CLP," ECHA. Accessed: Feb. 15, 2026. [Online]. Available: https://echa.europa.eu/regulations/clp/understanding-clp

[4] "GHS (Rev.4) (2011) | UNECE." Accessed: Feb. 17, 2026. [Online]. Available: https://unece.org/ghs-rev4-2011

[5] "GHS implementation | UNECE." Accessed: Feb. 15, 2026. [Online]. Available: https://unece.org/ghs-implementation-0

[6] *Regulation (EC) No 1272/2008 of the European Parliament and of the Council of 16 December 2008 on classification, labelling and packaging of substances and mixtures, amending and repealing Directives 67/548/EEC and 1999/45/EC, and amending Regulation (EC) No 1907/2006 (Text with EEA relevance)*, vol. 353. 2008. Accessed: Apr. 15, 2026. [Online]. Available: http://data.europa.eu/eli/reg/2008/1272/oj

[7] "Hazard Communication - Overview | Occupational Safety and Health Administration." Accessed: Feb. 15, 2026. [Online]. Available: https://www.osha.gov/hazcom

[8] "What Is NLP (Natural Language Processing)? | IBM." Accessed: Feb. 17, 2026. [Online]. Available: https://www.ibm.com/think/topics/natural-language-processing

[9] "Natural Language Processing (NLP) Market Size, Share | Industry Report," MarketsandMarkets. Accessed: Feb. 17, 2026. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/natural-language-processing-nlp-825.html

[10] "Apple Intelligence and Siri," Apple. Accessed: Jun. 26, 2026. [Online]. Available: https://www.apple.com/apple-intelligence/

[11] "Google Gemini," Gemini. Accessed: Jun. 26, 2026. [Online]. Available: https://gemini.google.com

[12] "Amazon Alexa Voice AI | Alexa Developer Official Site," Amazon Alexa. Accessed: Jun. 26, 2026. [Online]. Available: https://developer.amazon.com/en-US/alexa.html

[13] J. Hutchins, "The first public demonstration of machine translation: the Georgetown-IBM system, 7th January 1954" https://api.semanticscholar.org/CorpusID:132677.

[14] N. Chomsky, *Syntactic structures*. in Syntactic structures. Oxford, England: Mouton, 1957, p. 116.

[15] J. Weizenbaum, "ELIZA—a computer program for the study of natural language communication between man and machine," *Commun ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966, doi: 10.1145/365153.365168.

[16] "ELIZAGEN." Accessed: Apr. 15, 2026. [Online]. Available: https://sites.google.com/view/elizagen-org/

[17] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The Mathematics of Statistical Machine Translation: Parameter Estimation," *Comput. Linguist.*, vol. 19, no. 2, pp. 263–311, 1993.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Sep. 07, 2013, *arXiv*: arXiv:1301.3781. doi: 10.48550/arXiv.1301.3781.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” Dec. 14, 2014, *arXiv*: arXiv:1409.3215. doi: 10.48550/arXiv.1409.3215.
[20] N. Malingan, “Word2Vec in NLP,” Scaler Topics. Accessed: Apr. 15, 2026. [Online]. Available: https://www.scaler.com/topics/nlp/word2vec/
[21] A. Vaswani *et al.*, “Attention Is All You Need,” Aug. 02, 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.
[22] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training”.
[23] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” Jul. 22, 2020, *arXiv*: arXiv:2005.14165. doi: 10.48550/arXiv.2005.14165.
[24] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” May 31, 2016, *arXiv*: arXiv:1605.08695. doi: 10.48550/arXiv.1605.08695.
[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016.
[26] K. Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” in *Document retrieval systems*, GBR: Taylor Graham Publishing, 1988, pp. 132–142.
[27] Y. Goldberg, “A Primer on Neural Network Models for Natural Language Processing,” Oct. 02, 2015, *arXiv*: arXiv:1510.00726. doi: 10.48550/arXiv.1510.00726.
[28] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[29] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.
[30] Y. Wu *et al.*, “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation,” Oct. 08, 2016, *arXiv*: arXiv:1609.08144. doi: 10.48550/arXiv.1609.08144.
[31] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, “Deep Learning Enabled Semantic Communication Systems,” 2020, doi: 10.48550/ARXIV.2006.10685.
[32] H. Touvron *et al.*, “LLaMA: Open and Efficient Foundation Language Models,” Feb. 27, 2023, *arXiv*: arXiv:2302.13971. doi: 10.48550/arXiv.2302.13971.
[33] G. Chen, Z. Cheng, Q. Lu, W. Weng, and W. Yang, “Named Entity Recognition of Hazardous Chemical Risk Information Based on Multihead Self-Attention Mechanism and BERT,” *Wirel. Commun. Mob. Comput.*, vol. 2022, no. 1, p. 8300672, 2022, doi: 10.1155/2022/8300672.
[34] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, “LayoutLM: Pre-training of Text and Layout for Document Image Understanding,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, in KDD ’20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 1192–1200. doi: 10.1145/3394486.3403172.
[35] Z. Gu *et al.*, “XYLayoutLM: Towards Layout-Aware Multimodal Networks for Visually-Rich Document Understanding,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4583–4592. Accessed: Feb. 17, 2026. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Gu_XYLayoutLM_Towards_Layout-Aware_Multimodal_Networks_for_Visually-Rich_Document_Understanding_CVPR_2022_paper.html
[36] A. Suman, M. Khan, V. Talreja, J. Penfield, and S. Crowell, “A machine learning driven automated system for safety data sheet indexing,” *Sci. Rep.*, vol. 14, no. 1, p. 4415, Feb. 2024, doi: 10.1038/s41598-024-55231-1.

[37] “Safety Data Sheet (SDS) Management | US.” Accessed: Feb. 17, 2026. [Online]. Available: https://sdsmanager.com/us/sds-management/
[38] A. Fishwick, “Launch of AI-powered SDS Smart Extraction | EcoOnline US,” EcoOnline. Accessed: Apr. 14, 2026. [Online]. Available: https://www.ecoonline.com/en-us/news/ecoonline-introduces-ai-powered-sds-smart-extraction/
[39] “H/P phrases - Samancta - European Union.” Accessed: Apr. 16, 2026. [Online]. Available: https://sampling-manual-customs-taxation.ec.europa.eu/health-and-safety/hazardous-substances/hazard-symbols/hp-phrases_en
[40] “google-bert/bert-base-uncased · Hugging Face.” Accessed: Mar. 29, 2026. [Online]. Available: https://huggingface.co/google-bert/bert-base-uncased
[41] “google-bert/bert-base-multilingual-cased · Hugging Face.” Accessed: Mar. 29, 2026. [Online]. Available: https://huggingface.co/google-bert/bert-base-multilingual-cased
[42] “Hugging Face – The AI community building the future.” Accessed: Mar. 29, 2026. [Online]. Available: https://huggingface.co/
[43] “Ollama.” Accessed: Mar. 21, 2026. [Online]. Available: https://ollama.com
[44] “Home | Open WebUI.” Accessed: Mar. 21, 2026. [Online]. Available: https://openwebui.com/
[45] “meta-llama/Llama-3.3-70B-Instruct · Hugging Face.” Accessed: Mar. 21, 2026. [Online]. Available: https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct

Appendix A

Full system prompt used for information extraction with the LLM

```
f"""Para todas as perguntas seguintes, forneça a resposta exclusivamente em formato
JSON.
As perguntas incluem dicas para orientar o formato e conteúdo esperado.

**Formato de Pergunta:**
Pergunta: Qual a <entidade>?
Dica:"<dica_da_pergunta>

**Formato de Resposta Esperado:**
{{"<entidade>": ["<valor1>", "<valor2>", ...]}}

**Regras:**
1. Aguarde uma pergunta antes de responder. Não forneça respostas até que uma pergunta
seja explicitamente feita.
2. Extraia informações diretamente do contexto fornecido no final desta mensagem,
usando expressões/frases exatamente como aparecem no contexto e com o mínimo de
modificações possível.
3. Se a resposta não estiver no contexto, retorne uma lista com uma string vazia:
    {{"<entidade>": [""]}}
4. Todas as respostas devem estar no formato JSON.
5. Não inclua informações externas ou interpretadas.

**Exemplos de Perguntas e Respostas:**
1. Pergunta:
Pergunta: "Quais as Recomendações de prudência (Códigos P***)?
Dica: "Podem ser múltiplos itens.

1. Resposta:
{{
    "recomendacoes_de_prudencia": [
        "P101 - Se for necessário consultar um médico, mostre-lhe a embalagem ou o
rótulo.",
        "P102 - Manter fora do alcance das crianças."
    ]
}}

2. Pergunta:
Pergunta: Qual a palavra sinal?
Dica: Apenas 0 ou 1 valores.

2. Resposta:
{{"palavra_sinal": ["Atenção"]}}

3. Pergunta:
Pergunta: O que fazer em caso de contato da substância com os olhos?
Dica: Apenas 0 ou 1 valores. Citação pode ser longa.

3. Resposta:
{{"contato_com_olhos": ["Lavar imediatamente com água em abundância por vários
minutos. Consultar um médico."]}}

**CONTEXTO:**
{context}

**Atenção:** Não responda nada agora. Apenas aguarde a próxima pergunta."""
```

Full system prompt used for information extraction with the LLM translated to English.

```
f"""For all following questions, provide the answer exclusively in JSON format.
The questions include hints to guide the expected format and content.

**Question Format:**
Question: What is the <entity>?
Hint: "<question_hint>"

**Expected Response Format:**
{{"<entity>": ["<value1>", "<value2>", ...]}}

**Rules:**
1. Wait for a question before answering. Do not provide answers until a question is explicitly asked.
2. Extract information directly from the context provided at the end of this message, using expressions/phrases exactly as they appear in the context and with as few modifications as possible.
3. If the answer is not in the context, return a list with an empty string:
  {{"<entity>": [""]}}
4. All answers must be in JSON format.
5. Do not include external or interpreted information.

**Examples of Questions and Answers:**
1. Question:
Question: "What are the Precautionary Statements (P*** codes)?
Hint: "There may be multiple items."

1. Answer:
{{
    "precautionary_statements": [
        "P101 - If medical advice is needed, have product container or label at hand.",
        "P102 - Keep out of reach of children."
    ]
}}

2. Question:
Question: What is the signal word?
Hint: Only 0 or 1 values.
2. Answer:
{{"signal_word": ["Warning"]}}

3. Question:
Question: What should be done in case of eye contact with the substance?
Hint: Only 0 or 1 values. Citation may be long.
3. Answer:
{{"eye_contact": ["Rinse immediately with plenty of water for several minutes. Consult a doctor."]}}

**CONTEXT:**
{context}

Attention: Do not answer anything now. Just wait for the next question."""
```