Automated Extraction of Insurance Product Characteristics

Francisco Oliveira[1][0009−0006−3125−673X], Paulo Gandra de Sousa[1,2][0000−0001−9354−6608], Luiz Faria[1][0000−0002−1169−7785], Duarte Cardoso[2][0009−0005−9370−0085], and João Teixeira[1][0009−0005−5508−8446]

[1] Instituto Superior de Engenharia do Porto 1201545@isep.ipp.pt
[2] msg Life Iberia

Abstract. The increasing complexity and diversity of insurance products have highlighted the need for efficient methods to interpret and manage the detailed information present in regulatory documents. This project explores the application of Natural Language Processing (NLP) and Large Language Models (LLMs) in the automatic extraction of relevant characteristics of these products, addressing challenges such as structuring technical texts and accurately identifying rules, conditions, and variations.
The research focuses on analyzing the state of the art in technologies such as vector databases, LLMs, knowledge graphs, and agentic workflows, as well as evaluating NLP tools and methodologies. The text goes on to explore the primary challenges encountered during the interpretation of insurance documentation, as well as the transformation of unstructured data into organized formats that are compatible with modelling systems. The solution developed responded satisfactorily to the objectives established, enabling the structured and consistent extraction of product characteristics from regulatory documents. To this end, AI agent-based workflows were used, supported by LLMs and validation schemes, ensuring the quality and consistency of the results.

Keywords: · Document extraction · Named Entity Recognition · Large Language Models · AI Agentic Workflows · Insurance Product

1 Introduction

The insurance sector plays a crucial role in the economy, but the increasing complexity of its products, defined by extensive rules, conditions, and regulatory constraints, makes their development and maintenance challenging.

These specifications are detailed in regulatory documents approved by supervisory authorities, often including contractual terms, actuarial behavior, and business rules. Their unstructured and technical nature makes the interpretation and extraction of information a demanding and time-consuming task. Additionally, frequent regulatory updates require insurers to adapt their products constantly, reinforcing the need for efficiency, precision, and compliance.

Currently, msg Life Iberia consultants manually interpret and translate regulatory specification documents into product modeling requirements. This manual process is inefficient, error-prone, and difficult to scale. Recent advances in AI present an opportunity to automate the extraction of relevant product information from textual sources. However, applying these technologies in the insurance domain remains complex due to the heterogeneity and lack of standard structure across regulatory documents, the difficulty in generalizing extraction rules for different products and insurers, and the frequent incompatibilities between the extracted data and existing modeling platforms.

This project aims to develop a tool capable of automatically interpreting insurance specification documents, extracting the main product characteristics, and converting them into a structured representation compatible with msg Life Iberia's modeling platform.

The main goals are:

- Define a standardized output structure for product information.
- Implement an extraction tool to convert unstructured text into structured data.
- Integrate the extracted data into the company's modeling system to generate initial insurance product models automatically.

This automation seeks to improve accuracy, reduce manual effort, and increase consistency in the transformation of regulatory documents into software-ready models.

The remainder of this document is organised into five sections. Section 2 reviews related work. Section 3 outlines the document analysis and conceptual foundations of the extraction model. Section 4 details the system's development, from document processing to testing. Section 5 presents the results of the proposed approach. Section 6 concludes the report with final considerations and improvement opportunities.

2 Related Work

Research on document understanding and information extraction has evolved significantly with the adoption of NLP and LLMs. Several studies have explored different strategies to transform unstructured insurance or regulatory documents into structured and machine-interpretable data.

Authors in [1] investigated the application of two widely used NLP libraries, NLTK and SpaCy, for automatic text summarization. Their study proposed a hybrid approach that integrates the strengths of both libraries to enhance precision and adaptability in document processing. By employing evaluation metrics such as ROUGE and BLEU, the authors provided practical insights into how these tools can be integrated into systems requiring automated analysis of extensive and unstructured textual data, laying foundational work for structured information extraction.

In the context of robustness in data extraction, work in [2] evaluated multi-modal LLMs capable of processing both textual and visual information. Their experiments demonstrated that integrating visual inputs (e.g., rotated or misaligned documents) significantly improves the extraction of structured data, especially when dealing with scanned or low-quality insurance documents. This work shows how multi-modal models can overcome limitations of traditional OCR-based methods.

Focusing on the insurance sector, the study in [3] explored the use of LLMs such as GPT for actuarial and insurance tasks, including risk modeling, predictive claims analysis, and policy simulation. The authors emphasized the transformative potential of these models in automating analytical processes, while noting challenges related to data privacy, regulatory compliance, and the need for domain-specific fine-tuning.

Earlier research by [4] introduced a semantic, ontology-based framework for processing cyber insurance policies. The proposed system used AI and logic-based reasoning to populate knowledge graphs from policy text, enabling automated identification of inclusions and exclusions. Although effective in capturing semantic relationships, such symbolic approaches are less scalable compared to modern LLM-based techniques.

Overall, the reviewed studies demonstrate a clear evolution from traditional NLP and semantic approaches toward hybrid and LLM-driven systems capable of performing complex reasoning, visual-text integration, and structured data generation. This progression highlights the growing feasibility of applying agentic and workflow-based architectures to automate the extraction of insurance product characteristics from regulatory documents.

3 Analysis

The analysis focused on two main aspects: the structure and characteristics of insurance documents, and the identification of core business concepts that define insurance products. This foundation was essential to address the challenges of processing unstructured information.

Insurance specification documents are the foundation of the entire modeling and automation process. They contain the rules, conditions, and characteristics that define how insurance products operate, and are the primary source from which structured representations must be derived. These documents combine various content types - such as contractual clauses, exclusions, eligibility conditions, and free-form text - which significantly increases their complexity.

Moreover, document structures vary widely between insurers, reinforcing the need for a flexible extraction approach capable of adapting to different layouts and writing styles. Despite these differences, most documents share a common conceptual base, including recurring elements such as products, coverages, conditions, and exclusions.

Based on the analysis of documents and information gathered through meetings with domain experts, a conceptual map of the insurance product model was

defined (cf. Fig. 1). This model identifies the core entities and their relationships, which form the semantic structure of an insurance product.

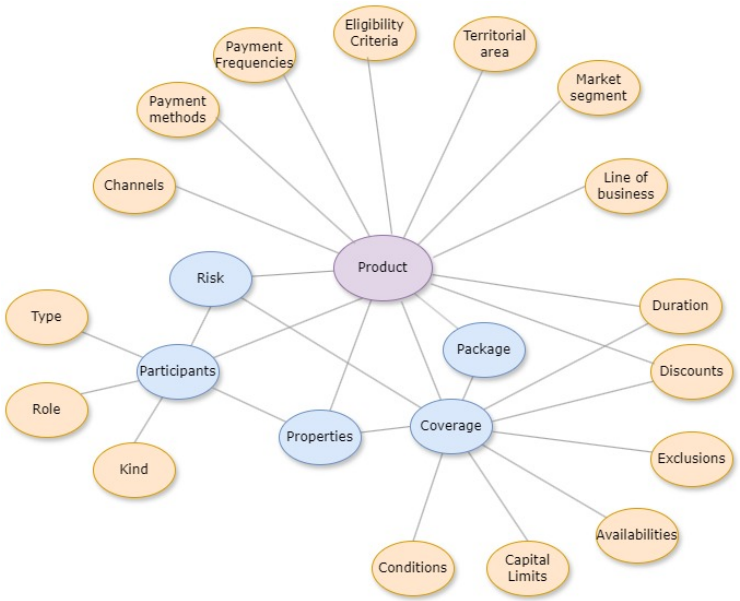

Fig. 1. Concept map of the insurance domain.

The central entity, *Product*, represents the core unit of the insurance offering. Each *Product* may include one or more *Packages*, corresponding to commercial variants or plans. A *Package* groups several *Coverages*, which describe the actual risks insured. Coverages include key elements such as exclusions, applicability rules, duration, limits, and conditions. The entity *Properties* defines variable attributes - such as effective date, birth date, or species - that influence underwriting and risk management. These properties may apply to participants, coverages, or the product itself. *Participants* represent the actors involved in the policy, such as the policyholder, insured person, or beneficiary, whose characteristics may determine eligibility and benefits. Finally, the *Risk* entity encapsulates the insurable exposure associated with the product and serves as a reference for underwriting and pricing rules. Together, these entities - *Product*, *Packages*, *Coverages*, *Properties*, *Participants*, and *Risk* - form a conceptual graph that enables a comprehensive and flexible representation of insurance products.

4 Development

This section details the development of the extraction system, covering document processing strategies, workflow orchestration, schema design, prompting

techniques, and testing methodology. Each subsection describes a specific stage of the implementation, from handling heterogeneous document formats to ensuring structured, validated output.

4.1 Document Processing

Document processing plays a crucial role in ensuring that the textual content extracted from insurance-related documents maintains the integrity and structure of the original source. Given the heterogeneity of file formats, layouts, and levels of complexity, distinct strategies and tools were adopted to handle different types of documents, including those containing tables, long documents, and scanned files.

Before structured information can be extracted, it is essential to correctly load and preprocess the input documents. For this purpose, the **PyMuPDF** library was employed as the main loader. It provides a low-level Python API that grants direct access to document structures - such as text blocks, positions, and metadata - while ensuring high performance even for large files [5]. Furthermore, it allows text extraction in Markdown format, facilitating interpretation by LLMs.

However, when processing documents containing complex tables, PyMuPDF proved insufficient for preserving the original tabular layout. In such cases, the **Docling** loader was used, offering superior preservation of table structures through native conversion to Markdown [6]. This approach ensures that key insurance data, such as coverage limits and package details, are accurately associated with their corresponding headers. Although more computationally demanding, Docling consistently provided better accuracy for complex tables, while PyMuPDF remained faster and effective for simpler cases.

Processing **long documents** introduced another challenge: maintaining sufficient context for the LLM without exceeding token limits. To address this, a batching strategy was initially adopted, splitting documents into multiple segments processed in parallel. Despite improving execution time, this method often produced duplicated or inconsistent outputs. Consequently, a sequential and incremental approach was implemented, preserving contextual continuity between batches and leading to more coherent and reliable extractions.

Finally, for **scanned or image-based documents**, traditional loaders failed to detect textual content due to the absence of machine-readable text. In these cases, an LLM-based loader was employed to perform Optical Character Recognition (OCR) directly through the model, converting the base64-encoded image of the document into structured text. This approach proved to be the only viable solution for such documents, successfully recovering the complete textual content with high fidelity to the original layout.

4.2 System Flow

The current system provides the LLM with the complete product schema, guiding the extraction process and reducing ambiguity. Fig. 2 illustrates the system

workflow. Once a document is uploaded, a central router determines the appropriate extraction path, invoking specialized agents such as the *Product Info Extractor*. This agent includes sub-agents responsible for specific extraction tasks and supports two main modes: document analysis or structured information extraction.

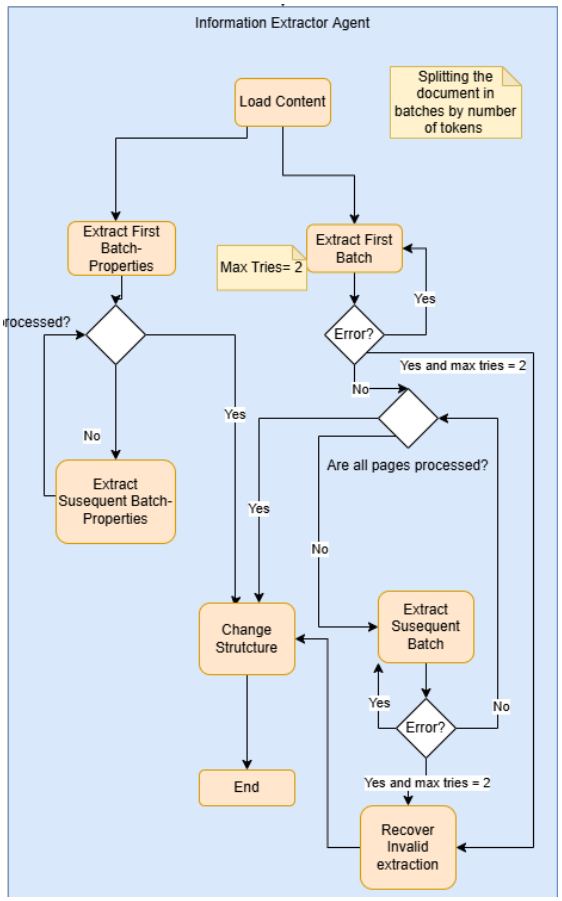

Fig. 2. Fluxo do Sistema na terceira abordagem

During the extraction phase, the document content is first divided into batches based on token count. This segmentation is necessary to manage context size and ensure efficient interaction with the LLM. Two agents operate in parallel on the initial batch: *Extract First Batch Properties* and *Extract First Batch*, which focus respectively on product properties and product characteristics. For larger documents, subsequent batches are processed iteratively by the *Extract Subse-*

quent Batch agent, which integrates newly extracted information with previously processed data to maintain contextual consistency.

The system also incorporates reliability mechanisms to handle potential failures during LLM interaction. If the extraction fails due to schema complexity or data quality issues, the process can automatically restart, with up to two attempts. Additionally, when partial results contain invalid or incomplete information, the *Recover Invalid Information* agent filters and restores valid data, ensuring maximum retention of useful content.

Finally, once all product details are extracted, the information is consolidated in the *Change Structure* module, responsible for transforming and formatting the results into the structure required by the *Product Machine*. This guarantees full compatibility with downstream systems for product import or update operations. The overall workflow thus ensures a robust, adaptive, and context-aware extraction process that balances efficiency, reliability, and data integrity.

4.3 Schemas and Structured Output

The *Pydantic* schemas constitutes the foundation of the system's structured output mechanism. It defines how extracted data is organized, validated, and represented throughout the pipeline. Acting simultaneously as a *data schema* and a *validation layer*, Pydantic ensures that all information produced by the LLM adheres to the predefined structure, maintaining both semantic integrity and technical consistency [7]. In practice, Pydantic serves as the formal contract between the extraction layer and the Product Machine, guaranteeing that outputs are immediately compatible with downstream components.

Each schema encapsulates detailed field descriptions and representative examples, providing the LLM with explicit guidance on the expected format and meaning of each attribute. In the insurance domain, where entities are highly interrelated, this structured definition is crucial. A product may include multiple coverages, participants, and modalities—each governed by its own Pydantic schema. This modular design mirrors the hierarchical organization of insurance products and enables consistent validation across heterogeneous documents.

In parallel, independent property schemas capture variable parameters—such as insured age, coverage limits, and contract duration—that influence product definitions. These schemas are later merged into the unified product model, allowing flexible yet controlled integration of all extracted elements within a single, validated structure. This structured representation allows the system to model complex relationships with precision and supports accurate, schema-driven extraction.

The extraction process itself is tightly coupled with these schemas. Using the *TrustCall* library, the system enforces schema-guided responses through Pydantic models. This ensures that every output generated by the LLM is automatically validated against the expected data structure. The process starts with *with_ structured_ output*, which applies Pydantic validation to the LLM's response, and then iteratively refines invalid or incomplete fields via targeted

patch operations. This approach allows for progressive, lossless refinement of the extracted data object while preserving all previously validated content [8].

At runtime, an extractor is initialized and configured with the appropriate Pydantic schema, guiding the LLM toward structured and compliant outputs. As documents are processed in batches, the system incrementally updates the existing structured object through controlled *patches*, maintaining alignment with the schema and ensuring consistent structured output across multiple extraction iterations.

4.4 Prompts

Prompts play a central role in guiding the LLM during the extraction process. They serve as the main communication interface between the user or system and the model, providing clear and precise instructions that define the extraction objectives. The quality and clarity of a prompt directly affect the coherence and accuracy of the generated output, especially in structured data extraction tasks.

Following OpenAI's official prompt design guidelines [9], three main prompt types were considered: *zero-shot* (instructions without examples), *few-shot* (with illustrative examples), and *chain-of-thought* (step-by-step reasoning). In this project, carefully crafted prompts ensured that the LLM adhered to the predefined data schema and minimized formatting or validation errors.

Different agent tasks required specific prompts, though common components were often reused to maintain consistency, particularly when processing documents in batches. For initial batches, prompts focused on extracting data according to the schema, while subsequent batches were instructed to update existing structures with new information. All prompts were written in Markdown format, clearly organizing roles, objectives, detailed instructions, and document content to optimize the LLM's understanding and output quality.

4.5 Testing and Evaluation

The entire development process followed an *evals*-oriented approach, where testing played a central role in ensuring reliability and performance. The main goal of these tests was to detect and minimize anomalies by continuously evaluating the extraction results against a predefined *Golden Standard Output*. This benchmark allowed for fast, objective validation without manual inspection after each extraction [10].

A similarity-based scoring method was used to compare each extracted output with its reference file, using the *Non-LLM String Similarity* metric. A similarity threshold of 0.70 (on a 0–1 scale) was empirically defined to account for semantically equivalent but textually different responses. This metric objectively measured structural and semantic coherence between runs and helped track performance variations over time.

To evaluate robustness, tests were performed on documents of varying complexity - simple, semi-complex, and complex - reflecting different structural and

formatting challenges. Additionally, a complementary test was introduced based on counting extracted elements per field, enabling detection of omissions regardless of wording differences. Although this approach had limitations (e.g., merged or split fields), combining it with similarity-based tests provided a reliable overall evaluation of the extraction quality and consistency across multiple executions.

5 Results

Overall, the final approach demonstrated more consistent and reliable results compared to previous iterations.

Table 1. Scores obtained over multiple runs (final approach)

Run	Simple	Semi_complex	Complex
1	0.8142	0.6251	0.7549
2	0.7544	0.6002	0.7379
3	0.7747	0.6600	0.7633
4	0.8388	0.6408	0.6174
5	0.8613	0.6408	0.7712
6	0.7497	0.7736	0.7595
7	0.8240	0.7681	0.5891
8	0.7624	0.7530	0.7675
9	0.7691	0.7210	0.5888
10	0.7875	0.6667	0.7718

The values in Table 1 represent similarity scores where higher values indicate closer alignment between the extracted output and the reference Golden Standard. Simpler documents achieved the highest and most stable scores, while semi-complex and complex ones showed slightly more variation, as expected due to their structural diversity. Importantly, integration tests revealed a significantly lower failure rate, confirming that most fields were accurately extracted. Manual validation further supported these findings, showing that the overall extraction quality achieved by the system is satisfactory and suitable for practical application.

6 Conclusion

Insurance product specifications are traditionally delivered as long, heterogeneous regulatory documents, making manual interpretation slow, costly, and error-prone. This project addressed the challenge of automating the extraction of structured product characteristics from these unstructured sources using LLM-based workflows. First, a unified output structure was defined and refined throughout the project, providing a consistent schema applicable to all document

types. Second, a functional prototype was developed to extract product characteristics from unstructured insurance documents, producing structured outputs aligned with the defined schema. Finally, the system was integrated with the PM, enabling the automatic generation of insurance product templates through an API-based connection.orgDespite the positive results, several limitations remain. LLMs can occasionally generate inaccurate or hallucinated responses, which affects data reliability. A possible mitigation would involve assigning confidence scores to each extracted field to support automatic validation. Performance efficiency is another concern, as extraction can be computationally expensive due to multiple LLM calls. Future improvements could include caching mechanisms, intelligent pre-filtering of content, and context compression using semantic embeddings to reduce redundant processing and optimize both runtime and cost.

Beyond the insurance sector, the proposed approach can be applied to other domains that rely on complex, heterogeneous, and technical documents. Its workflow, especially the separation of document loading, batch processing, and schema validation, supports adaptation to a wide range of file formats, including PDF, Word and Excel. In most cases, updating the schema to reflect the target domain is sufficient for the workflow to operate effectively.

References

1. Amade, D., Chandra, R., Sinha, V. K., Anand, D.: Automatic Text Summarization Using NLTK & SpaCy. *SSRN Electronic Journal* (2024). https://doi.org/10.2139/SSRN.4742012
2. Biswas, A., Talukdar, W.: Robustness of Structured Data Extraction from In-plane Rotated Documents using Multi-Modal LLMs. *arXiv preprint* arXiv:2402.07153 (2024).
3. Balona, C.: ActuaryGPT: Applications of Large Language Models to Insurance and Actuarial Work. *SSRN Electronic Journal* (2024). https://doi.org/10.2139/ssrn.4750341
4. Joshi, K., Pande Joshi, K., Mittal, S.: A Semantic Approach for Automating Knowledge in Policies of Cyber Insurance Services. In: *Proceedings of the 2020 IEEE International Conference on Web Services (ICWS)*, pp. 33–40. IEEE (2020). https://doi.org/10.1109/ICWS.2020.00018
5. PyMuPDF Developers: *PyMuPDF Documentation.* (2025). Available at: https://pymupdf.readthedocs.io/en/latest/
6. LangChain Documentation: *Docling Document Loader Integration.* (2025). Available at: https://python.langchain.com/docs/integrations/document_loaders/docling/
7. Pydantic Team: *Steering Large Language Models with Pydantic.* Pydantic Documentation (2024). Available at: https://pydantic.dev/articles/llm-intro
8. LangGraph Community: *TrustCall Integration with LangGraph.* LangChain Hub (2024). Available at: https://smith.langchain.com/hub/langchain/trustcall (Acedido em 7 set. 2025)
9. MacCallum, N., Lee, J.: *GPT-4.1 Prompting Guide.* OpenAI Cookbook (2025). Available at: https://cookbook.openai.com/examples/gpt4-1_prompting_guide
10. EvalOps: Evaluation-Driven Development: Building LLM Features with Confidence.
Available at: https://www.evalops.dev/blog/evaluation-driven-development