DATA EXTRACTION FROM HETEROGENEOUS INSURANCE POLICIES USING LLMS

Simeon Monov, Nikolay Pavlov, Miglena Dodleva

Abstract. *In this paper we explore the potential of large language models (LLMs) to automate data extraction from insurance policies of various formats, layouts, and structures. The motivation arises from the manual processing of such documents, which is time-consuming and error-prone due to the diversity of sources – Excel files, PDF documents, and text-based tables differing between insurers. This paper presents an experimental approach for data extraction from such policies using LLMs. The proposed method applies multiple models to identify and extract key information fields and represent them in a unified JSON structure. The extracted data from different models are analyzed, validated, and assessed in comparison with human interpretation of the content to obtain a realistic evaluation of precision, consistency, and completeness. The results indicate that the approach can also be applied to the processing of financial or legal documentation.*

Key words: Data Extraction, Large Language Models, LLMs, Insurance Policies, Heterogeneous Data, Automation, JSON, Financial Documentation, Legal Documentation.

Introduction

Operational workflows in insurance rely on structured data about insured objects, coverage limits, premiums and deductibles. In many companies this data is still extracted from policy documents manually, which is time-consuming and labor-intensive, and limits the degree of automation that can be achieved. A related approach to context retrieval for question generation systems is described in [1].

Problem Overview

In the insurance industry, companies receive large volumes of policy documents every day from multiple insurers and brokers. These policies arrive in heterogeneous formats and layouts, with varying use of abbreviations and domain-specific terminology. Critically, there is no guarantee of consistency over time: the same insurer may send a policy table in one structure today

and a completely different layout tomorrow, with reordered columns, renamed fields, or newly added attributes.

This structural variability makes automation extremely challenging. Traditional rule-based extraction pipelines depend on fixed templates, stable table structures or predefined column headers, and therefore break as soon as the layout changes. Consequently, many organizations still rely on manual data entry, where operators read each policy and retype key information into internal systems. Given the volume of incoming documents, this process is slow, labor-intensive and it does not scale with business growth. Addressing this heterogeneity is thus a central prerequisite for any robust automation of insurance policy processing. These challenges are increasingly recognised in the insurance industry, where LLMs are discussed as a potential game changer for core processes [2].

Task Definition

This work investigates the feasibility of using large language models to support the extraction of structured data from insurance policy documents and to represent it in a unified, machine-readable format [3, 4]. The input consists of real-world policies delivered by insurers and brokers in heterogeneous Excel and PDF formats, often with multiple tables and varying layouts. Given such a document, the system is expected to locate the relevant tables, interpret their structure, and propose a normalized JSON representation that can be integrated into downstream analytical and operational systems.

More specifically, the study aims to explore the potential of LLMs for automating parts of the data extraction process from insurance policies and to assess how well different models can handle diverse document types and layouts. The work provides an experimental extraction pipeline that maps policy content into a unified JSON schema and evaluates its performance in terms of precision, consistency, and overall readiness for use in real-world automation workflows.

Extraction Scope

The extraction pipeline focuses on insured objects belonging to three main categories: vessels, vehicles, and land-based equipment. For every detected object, the system aims to capture a consistent set of fields, including identifiers (such as vessel name, IMO number, vehicle registration or VIN), currency, coverage types, coverage limits, premiums, and deductibles. All extracted fields are mapped to a common JSON schema, independent of the original column names, ordering, or grouping used in the source document.

The study is limited to structured and semi-structured content available in digital Excel and PDF files. The evaluation concentrates on how reliably LLMs can populate the target JSON schema from tabular policy content, with emphasis on precision, consistency across models and documents, and practical suitability for integration into downstream automation workflows.

Experiment Setup

We conduct our experiments on four real insurance policy files provided by industry partners: two Excel files and two multi-page PDF reports. The files differ in layout, table structure and use of financial fields, and were selected to cover both relatively regular spreadsheets and complex, irregular PDF documents.

For each file we manually constructed a gold-standard JSON representation of all insured objects and their attributes. This gold standard serves as the reference against which all model outputs are evaluated.

We compare the performance of four generally available LLMs:

- **Gpt-4.1-mini**.
- **Gpt-5-mini**.
- **Google/gemini-2.5-flash**.
- **Mistralai/Mistral-Small-3.2-24B-Instruct-2506**.

For every model–document pair we evaluate how well the LLM reconstructs the set of insured objects together with their individual fields. Each (object, field) pair in the reference JSON is treated as a target item. Based on this, we compute:

- Total predicted items: all (object, field) pairs extracted by the LLM.
- True positives: predicted (object, field) pairs that match the reference JSON.
- False positives: predicted (object, field) pairs that do not exist in the reference (including hallucinated objects or incorrect field values).
- False negatives: reference (object, field) pairs that the model failed to extract.

From these counts we derive precision, recall and accuracy for the overall extraction of insured objects with their fields.

Methodology

Our extraction pipeline follows a fixed sequence of steps applied to each input policy document.

1. **Document type detection.** We first detect whether the input file is an Excel workbook or a PDF report. This determines which converter is used and how tables are located in the document.
2. **Conversion to Markdown.** The document is then converted into a normalized Markdown representation. For Excel files, each worksheet is rendered as a Markdown table. For PDF files, we apply a layout parser that reconstructs tables and exports them in the same Markdown format. This step reduces layout variability while preserving the essential tabular structure [5].
3. **Table extraction.** From the Markdown document we select the tables that potentially contain insured objects. Heuristics based on header keywords (e.g., vessel name, registration, premium) and column patterns are used to filter out irrelevant tables such as summaries or notes.
4. **LLM prompting.** The selected table content, together with brief contextual information about the policy, is passed to the LLM. The prompt specifies the target JSON schema (object categories and required fields) and instructs the model to output strictly valid JSON describing all detected objects.
5. **JSON parsing and validation.** The JSON output is parsed and checked against the schema; malformed responses are corrected when possible or discarded otherwise. The resulting structured data form the model's prediction for that document.
6. **Evaluation.** Finally, the predicted JSON is aligned with the gold-standard JSON for the same document, and we compute true positives, false positives and false negatives over all (object, field) pairs, from which precision, recall and accuracy are derived.

Engineering the Input Prompt

To obtain consistent and schema-compliant outputs across models, we use a single, shared system prompt and two user prompts depending on the document type (Excel or PDF).

The system prompt defines the model as an expert in Dutch insurance policies for vessels, vehicles and land-based equipment and explicitly fixes the target JSON schema with three top-level arrays: vessels, vehicles and lbe. It

provides detailed instructions on how to interpret Markdown tables, normalize column headers to English concepts, detect the currency, and handle numeric values (no symbols, no thousand separators, fixed two-decimal format). The prompt also specifies how to structure risks and deductibles as lists of covers, and how to treat unknown abbreviations by using their context while keeping the original short forms. The model is required to output only raw JSON that strictly follows the schema, with no additional text [6, 7].

The user prompt injects the actual policy content. For Excel files, we supply the policy tables rendered as Markdown and ask the model to extract all data into the predefined JSON schema. For PDF files, we provide two Markdown blocks:

- policy text: non-tabular narrative description of the insurance conditions;
- policy tables: one or more reconstructed tables containing the insured objects, limits, premiums and deductibles.

The prompt explicitly instructs the model to treat both blocks as parts of the same policy and to use the narrative text to interpret abbreviations, coverage names, limits and deductibles appearing in the tables. The same prompt templates are used for all evaluated LLMs.

Results

Table 1. Excel Sample A

Model	Gold Standard	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	Accuracy
Gpt-4.1-mini	20	20	0	0	100%	100%	100%
Gpt-5-mini	20	20	0	2	100%	91%	91%
Gemini 2.5 Flash	20	10	10	0	50%	100%	50%
Mistral-Small	20	14	4	6	88%	64%	64%

Table 2. Excel Sample B

Model	Gold Standard	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	Accuracy
Gpt-4.1-mini	12	12	0	1	100%	92.3%	92.3%
Gpt-5-mini	12	12	2	0	86%	100%	86%
Gemini 2.5 Flash	12	12	2	0	86%	100%	86%
Mistral-Small	12	12	0	0	100%	100%	100%

Tables 1 and 2 summarize the results on the two Excel policies. All models recover most insured objects and fields, with Gpt-4.1-mini achieving

perfect scores on Sample A and near-perfect scores on Sample B. Mistral-Small is also perfect on Sample B. In contrast, Gpt-5-mini and Gemini 2.5 Flash occasionally hallucinate extra deductibles or misplace premiums, which reduces their precision compared to Gpt-4.1-mini and Mistral.

Table 3. PDF Sample A

Model	Gold Standard	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	Accuracy
Gpt-4.1-mini	26	N/A	N/A	missing fields	N/A	N/A	N/A
Gpt-5-mini	26	N/A	N/A	missing fields	N/A	N/A	N/A
Gemini 2.5 Flash	26	19	3	0	86.4%	100%	86.4%
Mistral-Small	26	16-23	3-10	0	60-88%	100%	60-80%

For this policy, both Gpt models produced only a truncated extraction, covering roughly the first six objects, and are therefore excluded from the metric-based comparison.

Table 4. PDF Sample B

Model	Gold Standard	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	Accuracy
Gpt-4.1-mini	26	N/A	N/A	missing fields	N/A	N/A	N/A
Gpt-5-mini	26	N/A	N/A	missing fields	N/A	N/A	N/A
Gemini 2.5 Flash	26	19	3	0	86.4%	100%	86.4%
Mistral-Small	26	16-23	3-10	0	60-88%	100%	60-80%

Tables 3 and 4 show the results on the two PDF policies. For PDF Sample A, both GPT models truncate the extraction after the first part of the table and therefore do not yield reliable metrics. Gemini and Mistral process the full table but exhibit more structural and field-level errors. PDF Sample B is easier: all models reach high recall and precision around 0.86–0.91, with remaining errors mainly due to confusions between 'vehicles' and 'lbe' and a few missing fields.

Overall, Excel policies are close to fully automatable, while complex PDFs remain significantly more challenging.

Qualitative error analysis

We observed two dominant error classes: (i) schema/structural violations and (ii) value-level inaccuracies.

On Excel Sample A, Gpt-5-mini often omitted required keys (e.g., imo_number, class) when the source cell was empty, instead of including the keys with an explicit null value, which constitutes a schema completeness violation

and reduces recall. In the same file, Gemini 2.5 Flash preserved the object structure but showed premium deviations of ±0.01, likely due to rounding/formatting and numeric normalization. On Excel Sample B, Gpt-4.1-mini drops the class key when it is blank in the table, showing the same schema-completeness pattern observed in Sample A (missing required keys instead of outputting explicit null). In contrast, both Gpt-5-mini and Gemini 2.5 Flash occasionally hallucinated an extra deductible (e.g., "Machinery"), lowering precision.

For PDFs, the main failure mode is incomplete extraction: on PDF Sample A both GPT models produced truncated outputs covering only $\sim 20\%$ of the document (the initial portion of the table) and then stopped, which also prevents a meaningful assessment of value-level errors beyond the extracted segment. When full-table extraction succeeds (PDF Sample B), residual errors are mostly category confusions (vehicles vs lbe) and sporadic hallucinated fields (e.g., premium_rate) or missing keys. These patterns motivate a stricter schema validator (key completion with explicit null), fixed-decimal parsing and rounding, and PDF-specific chunking by object rows to prevent truncation.

Conclusion

This work evaluated the use of large language models for extracting structured data from heterogeneous insurance policies into a unified JSON schema. Across two Excel and two PDF policies, the models showed clearly different strengths.

For Excel files, Gpt-4.1-mini consistently delivered the best results, with high precision and recall and no hallucinated objects or extra fields. For difficult PDF policies with fragmented tables and mixed narrative text, Mistral-Small proved more robust, recovering more objects and fields when the layout was irregular.

These findings suggest that no single model is optimal for all document types. The most effective strategy is a combined one: use Gpt-4.1-mini as the primary engine for structured Excel policies, and rely on Mistral-Small for complex PDF cases where the tables are fragmented and the layout is unstable.

PDF parsing currently remains a clear bottleneck in our pipeline. In future work, we plan to experiment with more advanced PDF parsing techniques, for example by utilizing multimodal LLMs.

Acknowledgments

This study is supported by the project SP25-FMI-008 "Research and implementation of modern language models and artificial neural networks for au-

tomated processing, forecasting, and structuring of data and texts in specific application domains" at the Paisii Hilendarski University of Plovdiv.

References

[1] S. Monov, E. Myumyun, N. Pavlov, A. Nikolov, Question Context Retrieval Based on TOC Lookup in Question Generation and Testing System, *Proc. IMEA 2024*, https://fmi-plovdiv.org/GetResource?id=4861

[2] J. Cottongim, LLMs, Why the insurance industry needs its own large language model, Digital Insurance, 2023, https://www.dig-in.com/opinion/why-the-insurance-industry-needs-its-own-large-language-model

[3] IBM, What are large language models (LLMs)?, 2024, https://www.ibm.com/think/topics/large-language-models

[4] R. Bommasani et al., On the Opportunities and Risks of Foundation Models, arXiv:2108.07258, 2021, https://arxiv.org/pdf/2108.07258

[5] S. Hegselmann, TabLLM: Few-shot Classification of Tabular Data with Large Language Models, 2023, https://arxiv.org/abs/2210.10723

[6] Microsoft, Getting started with customizing a large language model (LLM), 2025, https://learn.microsoft.com/en-us/azure/ai-foundry/openai/concepts/customizing-llms?view=foundry-classic

[7] X. Amatriain, Prompt Design and Engineering: Introduction and Advanced Methods, arXiv:2401.14423, 2024, https://arxiv.org/abs/2401.14423

Simeon Monov[1], Nikolay Pavlov[1], Miglena Dodleva[1]
[1] Paisii Hilendarski University of Plovdiv,
Faculty of Mathematics and Informatics,
236 Bulgaria Blvd., 4027 Plovdiv, Bulgaria
Corresponding author: smonov@gmail.com