

Master's Programme in ICT Innovation (EIT Digital Master School)

# Hybrid Prompt Optimization Without Fine-Tuning: Enhancing Information Extraction and Translation in Vehicle Maintenance Records

A Prompt-Centric Approach to Multilingual Structured Data Extraction

---

**Zhiyuan Wu**

© 2025

This work is licensed under a [Creative Commons](https://creativecommons.org/licenses/by-nc-sa/4.0/) “Attribution-NonCommercial-ShareAlike 4.0 International” license.



---

**Author** Zhiyuan Wu

---

**Title** Hybrid Prompt Optimization Without Fine-Tuning: Enhancing Information Extraction and Translation in Vehicle Maintenance Records — A Prompt-Centric Approach to Multilingual Structured Data Extraction

---

**Degree programme** ICT Innovation (EIT Digital Master School)

---

**Major** Autonomous Systems

---

**Supervisor** Prof. Quan Zhou

---

**Advisors** Prof. Anders Hedman, Prof. Haibo Li

---

**Collaborative partner** Volvo Group

---

**Date** 29 May 2025

**Number of pages** 76+2

**Language** English

---

**Abstract**

Unstructured, multilingual maintenance logs hamper automated fault analysis, yet fully fine-tuning large language models (LLMs) is prohibitively costly. We therefore adapt a 70 B-parameter LLaMA-33 model without gradient updates using a two-step Hybrid Prompt Optimization (HPO) scheme that merges expert-written instructions with DSPy’s automatic prompt search. The approach is trained and tested on 1000 real, Volvo Trucks records while running only on CPU.

For evaluation we draw 10 disjoint test sets of 50 records each; scores reported here are the mean of those ten runs. Relative to a zero-shot baseline, manual prompts raise structured-field extraction accuracy by 23 %; DSPy adds 27 %; and HPO supplies a further 5 %, yielding a cumulative gain of  $\approx 60\%$ . For full-text translation the same steps deliver 55 %, +2 %, and +2 %, respectively. Paired two-tailed t-tests across the  $10 \times 50$  predictions confirm that all improvements remain significant at  $\alpha = 0.05$ . Under the best prompt, the lightweight LLaMA-33-70B reaches 92 % of GPT-4o’s extraction quality and slightly surpasses it in translation—yet consumes only a fraction of the compute budget.

These findings show that prompt engineering alone can unlock near-state-of-the-art multilingual extraction and translation, delivering a low-cost alternative to full model fine-tuning for industrial after-sales data.

---

**Keywords** Prompt Engineering, Large Language Models, DSPy, Hybrid Optimization, Information Extraction, Multilingual Translation, Automotive Industry

---

---

**Tekijä** Zhiyuan Wu

---

**Työn nimi** Hybridi-kehotteiden optimointi ilman hienosäätöä: Tiedon poiminnan ja käännöksen tehostaminen ajoneuvojen huoltolokeista —  
Kehotepohjainen lähestymistapa monikieliseen rakenteisen datan poimintaan

---

**Koulutusohjelma** ICT-innovaatio (EIT Digital -maisteriohjelma)

---

**Pääaine** Autonomiset järjestelmät

---

**Työn valvoja** Prof. Quan Zhou

---

**Työn ohjaajat** Prof. Anders Hedman, Prof. Haibo Li

---

**Yhteistyötaho** Volvo Group

---

**Päivämäärä** 29.05.2025

**Sivumäärä** 76+2

**Kieli** englanti

---

### Tiivistelmä

Rakenteettomat ja monikieliset huoltolokit vaikeuttavat automaattista vikojen analysointia, mutta suurten kielimallien (LLM) täysimääräinen hienosäätö on kustannuksiltaan erittäin raskasta. Sovellamme siksi 70 miljardin parametrin LLaMA-33-mallia ilman gradienttipäivityksiä käyttämällä kaksivaiheista hybridikehotteiden optimointimenetelmää (HPO), joka yhdistää asiantuntijan kirjoittamat ohjeet ja DSPy:n automaattisen kehotteiden haun. Menetelmää koulutetaan ja testataan 1000 aidolla Volvo Trucks -huoltokirjauksella käyttäen ainoastaan suorittimen laskentatehoa (CPU).

Arviointia varten muodostetaan 10 erillistä testijoukkoa, joissa kussakin on 50 huoltotietuetta; raportoidut tulokset ovat näiden kymmenen ajon keskiarvoja. Verrattuna nollakehotteisiin, manuaaliset kehotteet parantavat rakenteisten kenttien poimintatarkkuutta 23 %; DSPy lisää 27 %; ja HPO tuo vielä 5 % lisää, mikä johtaa kokonaisparannukseen, joka on  $\approx 60\%$ . Kokotekstin käännöksessä samat vaiheet tuottavat parannuksia 55 %, +2 % ja +2 %. Kahdennäytteiset parilliset t-testit  $10 \times 50$  ennusteen joukossa vahvistavat, että kaikki parannukset ovat tilastollisesti merkitseviä tasolla  $\alpha = 0.05$ . Parhaalla kehotteella kevyt LLaMA-33-70B saavuttaa 92 % GPT-4o:n poimintalaadusta ja ylittää sen hieman käännöstehtävässä — mutta vaatii vain murto-osan laskentatehosta.

Tulokset osoittavat, että pelkällä kehotesuunnittelulla voidaan saavuttaa lähes huipputason monikielinen poiminta ja käännös, mikä tarjoaa kustannustehokkaan vaihtoehdon mallien täyshienosäädölle teollisessa jälkimarkkinadatan käsittelyssä.

---

**Avainsanat** Kehotesuunnittelu, Suuret kielimallit, DSPy, Hybridioptimointi, Tiedon poiminta, Monikielinen käännös, Autoteollisuus

---

## **Preface**

First and foremost, I would like to express my deepest gratitude to my parents, as well as my aunt and uncle, for their unwavering love, encouragement, and support throughout my master's studies. Without their understanding and constant support, this journey would not have been possible.

Secondly, I would like to sincerely thank my supervisors for their invaluable guidance, patience, and my teammate support during the course of my thesis. I am also deeply grateful to Volvo Group for providing access to real-world data and all the necessary computing resources that made this research possible.

Gothenburg, 9 June 2025

Zhiyuan Wu

# Contents

<b>Abstract</b>	<b>3</b>
<b>Abstract (in Finnish)</b>	<b>4</b>
<b>Preface</b>	<b>5</b>
<b>Contents</b>	<b>6</b>
<b>Symbols and abbreviations</b>	<b>9</b>
<b>1 Introduction</b>	<b>11</b>
1.1 Background	12
1.2 Motivation	12
1.3 Problem Statement	13
1.4 Research Objectives	14
<b>2 Literature Review</b>	<b>16</b>
2.1 Large Language Models (LLMs)	16
2.1.1 Large Language Model Meta AI Model(LLaMA Model)	16
2.1.2 Generative Pre-trained Transformer (GPT) Model	18
2.2 Embedding Techniques for Semantic Representation	19
2.2.1 Instruction-Tuned Embeddings	20
2.2.2 Mathematical Formulation	20
2.3 Fine-Tuning Costs and Limitations	21
2.3.1 Fine-Tuning Costs	22
2.3.2 Operational Challenges	22
2.4 Current Prompt Engineering Work in Information Extraction and Multilingual Applications	23
2.4.1 Prompt Engineering for Information Extraction	23
2.4.2 Prompt Engineering in Multilingual Applications	24
2.5 Manual Prompt Engineering in NLP	24
2.5.1 Zero-shot and Few-shot Prompting	24
2.5.2 Chain-of-Thought(CoT) Prompting	25
2.5.3 Role-based Prompting	26
2.5.4 Manual Iterative Prompt Design	27
2.6 Automatic Prompt Engineering in NLP	27
2.6.1 DSPy-based Prompt Optimization	27
2.6.2 Reinforcement Learning-based Prompting	28
2.7 Evaluation Metrics for Extraction and Translation	28
2.7.1 ROUGE Scores for Lexical Overlap	28
2.7.2 BERTScore for Semantic Similarity	30
2.7.3 Cosine Similarity of Sentence Embeddings	31
2.8 Research Gaps and Opportunities	32

<b>3</b>	<b>Methodology</b>	<b>32</b>
3.1	Data Collection and Preprocessing	33
3.1.1	Claim Table as Ground Truth	33
3.1.2	TMR Table as Target Corpus	33
3.1.3	Preprocessing the Claim Table	34
3.1.4	Dataset Splitting for Model Training and Evaluation	35
3.1.5	TMR Subset Sampling	36
3.2	Model Selection	36
3.3	Rationale for Choosing Manual and DSPy-Based Prompt Optimization	37
3.4	Prompt Optimization	38
3.4.1	Baseline Prompt Design	38
3.4.2	Manual Prompt Optimization — Relative Changes from Baseline	38
3.4.3	Automatic Prompt Optimization with DSPy	42
3.4.4	Hybrid Optimization: Combining Manual and DSPy-Based Methods	44
3.5	Rationale for Embedding Model Selection	45
3.6	Rationale for Metric Selection	45
3.7	Evaluation Framework	46
3.8	Test Framework Construction	47
<b>4</b>	<b>Results and Analysis</b>	<b>48</b>
4.1	Environment and Tools	48
4.2	Results	48
4.2.1	Complaint Field Results(LLaMA-33-70B)	49
4.2.2	Cause Field Results(LLaMA-33-70B)	49
4.2.3	Correction Field Results(LLaMA-33-70B)	49
4.2.4	Translation Field Results(LLaMA-33-70B)	49
4.2.5	Overall Performance Summary(LLaMA-33-70B)	49
4.2.6	Cross-Model Comparison Results: LLaMA-33-70B vs. GPT-4o	50
4.3	Analysis	57
4.3.1	Manual Prompt Engineering(LLaMA-33-70B)	57
4.3.2	DSPy Optimization(LLaMA-33-70B)	58
4.3.3	Hybrid Prompt Strategies(LLaMA-33-70B)	59
4.3.4	Summary of Trends(LLaMA-33-70B)	59
4.3.5	Cross-Model Comparison: LLaMA-33-70B vs. GPT-4o	60
<b>5</b>	<b>Discussion</b>	<b>60</b>
5.1	Introduction and Overview of Findings	60
5.2	Interpretation of Results(LLaMA-33-70B)	61
5.2.1	Comparison of Manual and Automatic Methods	61
5.2.2	Performance Drop from Baseline (M0) to Manual-1 (M1)	61
5.2.3	Performance Drop from Manual-11 (M11) to Manual-12 (M12)	62
5.2.4	Performance Drop from DSPy-Predict-FewShot (D1) to DSPy-CoT-FewShot (D3)	62
5.2.5	Significance of Manual Prompt Optimization	62

5.2.6	Significance of DSPy Automatic Prompt Optimization . . . .	62
5.2.7	Significance of the Hybrid Approach . . . . .	63
5.3	Cross-Model Generalization and Efficiency(LLaMA-33-70B vs. GPT-4o) . . . . .	63
5.4	Theoretical, Practical, and Ethical Implications . . . . .	64
5.4.1	Theoretical Contributions . . . . .	64
5.4.2	Practical Significance . . . . .	64
5.4.3	Trust and Ethical Considerations . . . . .	64
5.5	Limitations . . . . .	65
5.5.1	Company Domain Generalizability . . . . .	65
5.5.2	Scalability to Other Datasets, Domains, and Smaller Models . . . . .	65
5.5.3	Infrastructure and API Limitations . . . . .	66
5.6	Sustainability Considerations . . . . .	66
<b>6</b>	<b>Conclusion and future work</b>	<b>67</b>
6.1	Conclusion . . . . .	67
6.2	Future Work . . . . .	68
6.2.1	Advanced Prompt Optimization Techniques . . . . .	68
6.2.2	Cross-OEM Generalization and Domain Expansion . . . . .	68
6.2.3	Dynamic Prompting in Real-Time Applications . . . . .	68
6.2.4	Improved Evaluation and Feedback Integration . . . . .	68
6.2.5	Scalable Inference and Error Recovery Mechanisms . . . . .	68
<b>A</b>	<b>Appendix: Sample Records from Datasets</b>	<b>77</b>
A.1	Sample from Claim Table (Ground Truth) . . . . .	77
A.2	Sample from TMR Table (Target Corpus) . . . . .	77
<b>B</b>	<b>Appendix: pseudocode from prompt optimization</b>	<b>78</b>
B.1	Appendix: pseudocode from Dspy method . . . . .	78
B.2	Appendix: pseudocode from Hybrid method . . . . .	78

## Symbols and abbreviations

### Symbols

$\mathbf{f}(x)$	Function or model output given input $x$
$\nabla$	Gradient operator
$\ \cdot\ $	L2 norm (Euclidean norm)
$\cdot$	Dot product between vectors
$\Sigma$	Summation operator
$\alpha$	Significance level in statistical testing
$n$	Number of samples or tokens
$\mathbf{E}_x$	Embedding vector of token $x$
$\cos(\cdot)$	Cosine similarity function

### Operators

$\nabla f(x)$	Gradient of function $f$ with respect to $x$
$A \cdot B$	Dot product of vectors $A$ and $B$
$\ x\ $	L2 norm (Euclidean length) of vector $x$
$\sum_i$	Summation over index $i$
$\max(\cdot)$	Maximum function (used in BERTScore)
$\cos(A, B)$	Cosine similarity between vectors $A$ and $B$
$\text{Embed}(S)$	Sentence embedding computed from token vectors
$\text{Tokenize}(S)$	Tokenization of input string $S$
$\frac{d}{dt}$	Derivative with respect to variable $t$
$\frac{\partial}{\partial t}$	Partial derivative with respect to variable $t$

## Abbreviations

LLM	Large Language Model
HPO	Hybrid Prompt Optimization
CoT	Chain-of-Thought
DSPy	Declarative Self-Improving Prompting
RAG	Retrieval-Augmented Generation
NER	Named Entity Recognition
RE	Relation Extraction
RoPE	Rotary Positional Embedding
RMSNorm	Root Mean Square Normalization
SVO	Subject-Verb-Object
BERT	Bidirectional Encoder Representations from Transformers
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
MTEB	Massive Text Embedding Benchmark
CPU	Central Processing Unit
TPU	Tensor Processing Unit
API	Application Programming Interface
SOTA	State of the Art
NLP	Natural Language Processing
GPT	Generative Pre-trained Transformer
LLaMA	Large Language Model Meta AI
Ada	OpenAI's Embedding Model (e.g., <code>text-embedding-ada-003</code> )
BGE-M3	Beijing General Embedding Model 3
RL	Reinforcement Learning

# 1 Introduction

Artificial Intelligence (AI) and Natural Language Processing (NLP) technologies have updated rapidly these years. They have already been introduced in various industries to handle the task automation and improved the workflow efficiency. In the automotive industry, after-sales services have benefited from these technologies. An important area is the management and analysis of vehicle maintenance records which includes many manual records combined with automatic records. It is crucial for the engineer to get accurate diagnostics and effective repair methods from these maintenance records. However, these records often contain multilingual, noisy, and unstructured, error textual data, which complicates the use of these data.

With the rise of large language models (LLMs) technique, the automotive industry is presented with novel opportunities to tackle complex challenges in vehicle maintenance and data management. Automotive companies increasingly seek digital innovative, practical, and cost-effective digital solutions to manage this complexity. According to Federico [1] a post-doctoral researcher at Stanford University, there are three main cornerstones in AI application fields, which are LLM base capacity, fine-tuned methods, and prompt engineering.

The base capacity of LLMs, referring to their intrinsic ability to understand and generate natural language. It provides a powerful foundation for various downstream tasks without additional specialized training. The extensive pre-training of these models allows them to capture nuanced language patterns and context, enabling effective generalization even in complex scenarios. However, for traditional manufacturing companies like Volvo Group, developing a completely new LLM can be prohibitively expensive. According to Ohiri et al. [2], the compute cost of training state-of-the-art large-scale models can reach millions of dollars due to substantial computational resources required. Besides, the company needs to hire many experts in LLM fields to develop their own LLM. Therefore, it is not practical for Volvo Group to invest a huge sum of money in such projects.

Fine-tuning a LLM for this specific area is expensive. According to Xia's research [3], it indicated that Mixtral model with data size of 2 million queries fine-tuned by NVIDIA H100 GPU which will cost \$3460 approximately. More examples are discussed in Section 2.3. Hence, fine-tuning LLMs can be computationally expensive and time-consuming.

Prompt Engineering offers a cost- and time-efficient method to guide LLMs compared with traditional fine-tuning. Nananukul's team [4] who are from University of Southern California indicated that effective prompt optimization can be charged around 0.7\$ to 0.9\$ on WDC prompt pattern and 2\$ to 3\$ on Amazon-Google prompt pattern. Another study from Sabbatella [5] shows that it only took 300-800 seconds to optimize an effective prompt by using Prompt-BO method to get a similar level of accuracy by using the fine-tuned method on LLM.

Building on this cost-efficient strategy, this thesis explores Prompt Engineering to improve the processing of Volvo's multilingual truck maintenance records within automotive after-sales contexts by using a relatively small LLM to reach an acceptable result.

## 1.1 Background

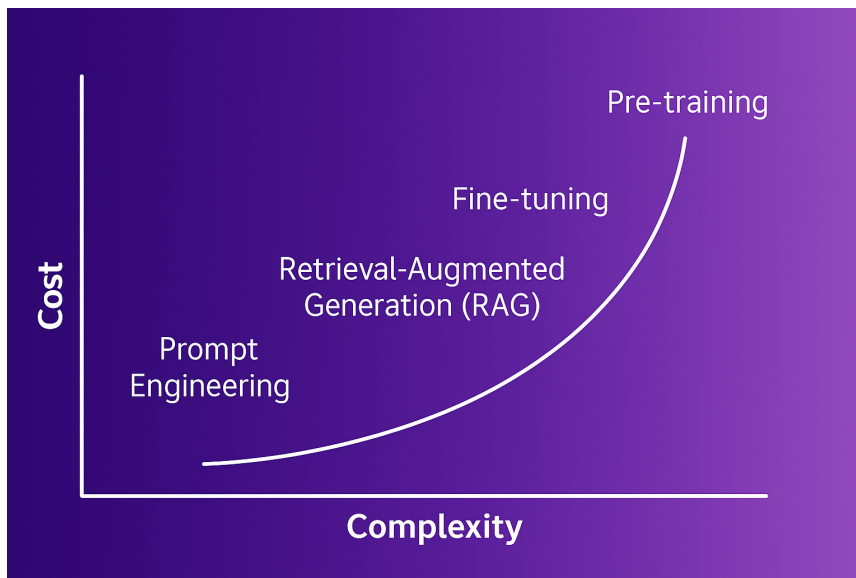
Truck maintenance records typically involve multilingual and unstructured textual data, which let efficient information extraction and translation difficult. Currently, automotive after-sales rely primarily on regular expression searches, simple sentence splitting and NLP techniques by data analysts to extract the key information. Such methods are often inefficient, error-prone, and time-consuming, which mentioned in Giordano's [6] study. Another problem is on translation, as our records are multilingual, company's like Volvo Group Truck Technology have its own standard for translation. For example in our case, we need to translate a large amount of Spanish texts into English in a pattern in English's subject-verb-object(SVO)structure, but according to a study by lozano [7], the Spanish often places adjectives after nouns, uses different question structures, and places object pronouns before or attached to verbs. Also, negation is done differently, often preceding the verb with "no" instead of using auxiliary constructions as in English. These syntactic and structural differences pose challenges for direct word-by-word translation, potentially leading to grammatically incorrect English outputs. If we simply use some translators like Google and DeepL, it will not satisfy Volvo's translation standard. These issues may misunderstand and mislead technicians' repairing plan and delay repairs, increase vehicle downtime, and negatively impact customer satisfaction. These issues are not only in the Volvo Group Trucks Technology, but are common issues in all automotive producers around the world. Therefore, these companies are trying to find effective solutions to fix these problems.

Recent developments in LLMs, such as GPT, Deepseek and LLaMA, have already made progress in improving automation in information extraction and translation tasks. However, adapting these models to specific industry requirements typically involves costly fine-tuning and long adapt time. According to a group of researchers from Tsinghua University [8], this process demands significant computational resources, extensive labeled datasets, and specialized expertise, making widespread industrial adoption difficult. Another aspect is that companies hope to save money and resources, they don't want to spend a large amount of funding on different LLMs, so the management hopes to reuse some LLMs already in company's LLM resource platform.

## 1.2 Motivation

Prompt Engineering offers an alternative option resource-efficient option compared to the traditional method of fine-tuning an LLM in a specific area. As prompt engineering, it enables the effective adaptation of LLMs without a substantial resource needed. With this kind of technique, it can help to reuse some pre-trained LLMs that aimed at other jobs in companies' LLM resource base. Hence, it is the new trend to find an effective way to optimize prompts, especially in the automotive industry. In addition, according to Lin [9], an expert from Amazon, prompt engineering offers the lowest cost and complexity compared to other LLM tuning methods, as illustrated in Figure 1.

Many methods on prompt engineering have come out in other industry. For example, a manual optimization methods called Chain-of-Thought Reasoning was



**Figure 1:** Overview of large model optimization approaches by AWS, illustrating the trade-off between cost and complexity among prompt engineering, fine-tuning, RAG, and pre-training. Source: AWS official documentation. (<https://aws.amazon.com/cn/blogs/china/sixteen-ways-of-prompt-engineering/>)

published out by Wei [10], it showed an improvement on solving math problem area. Another research by Kim [11], provides a method to optimize multilingual adaptation, and role-based prompting, have shown notable performance improvements. Similarly, Juan [12] found automated methods like DSPy-based optimization have demonstrated effectiveness in systematic prompt refinement on developing a multi-agent in generating smart contracts field. In addition, automatic prompt optimization tools like VolcEngine Prompt Optimization provide an automatic and displayable way to improve the prompt instructions for LLMs to better understand what their tasks are. In Figure2, it shows the Volcengine Prompt Optimization toolkit. It provides multiple LLMs to generate and optimize the prompt which user needs to describe their tasks.

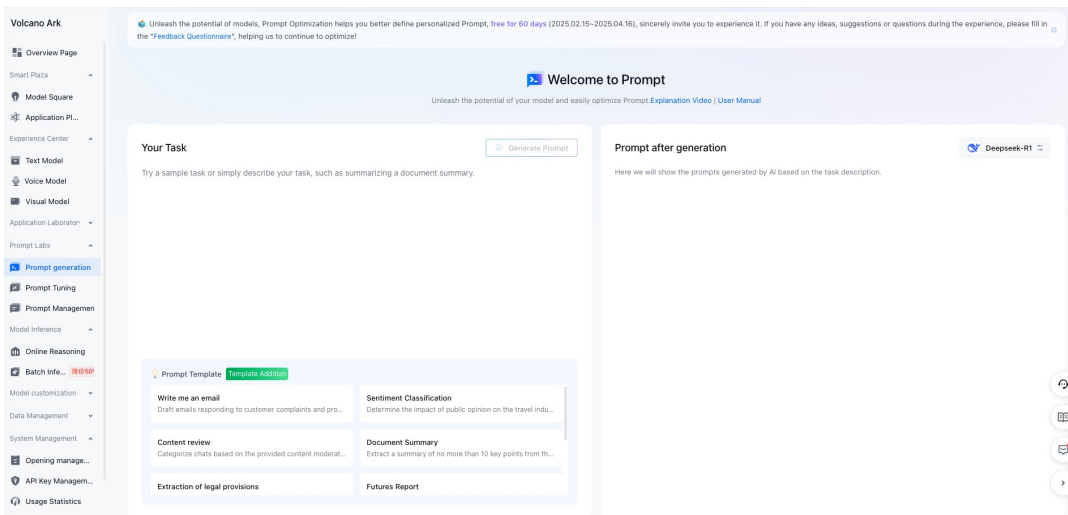
However, research combining manual and automatic optimization methods within the automotive after-sales context remains limited. This gap highlights the necessity to explore and maximize the benefits of integrated prompt optimization approaches specifically tailored to automotive industry applications.

### 1.3 Problem Statement

Comprehensive studies on Prompt Engineering underscore diverse manual and automatic optimization techniques to improve LLM performance. Nonetheless, a substantial research gap persists in the comparative analysis and integration of these optimization methodologies, especially within the automobile after-sales services sector. This disparity gives rise to our primary research question.

**Research Question:**

"How can a Hybrid Prompt Optimization approach, integrating manual prompt optimization and DSPy-based automatic optimization applied on a pre-trained LLM in other field, effectively enhance information extraction and translation performance from multilingual vehicle maintenance records?"



**Figure 2:** Volcengine Prompt Optimization Toolkit [Public domain], via official cite of Volcengine. (<https://console.volcengine.com/ark/region:ark+cn-beijing/aut/ope/startup>).

This study aims to evaluate and compare manual and automated prompt strategies, and assess the potential of a hybrid approach using real Volvo truck maintenance data from the Volvo's Claim table. Specifically, we focus on the detail column, which contains multilingual and unstructured descriptions of complaints, causes, and corrections.

Our hypothesis is that DSPy-based approaches give consistency and scalability but may need several iterations to get optimal prompts, while manual methods offer sophisticated contextual understanding but struggle with scalability. By balancing both advantages, a hybrid approach can effectively handle the resource-intensive nature of multilingual maintenance records.

## 1.4 Research Objectives

To systematically explore the stated research question, this study aims to establish a comprehensive baseline prompt explicitly describes for multilingual truck maintenance records.

Then, the research will investigate and evaluate various manual optimization methods, which will include the Chain of Thought reasoning, multilingual adaptations, role-based prompting, and provide good examples for LLM to learn its tasks. These strategies mainly enhance model accuracy and translation fidelity.

Concurrently, DSPy-based automatic prompt optimization methods like using different optimizers(e.g., BootstrapFewShot, BootstrapFewShotWithRandomSearch) and choose different DSPy modules(e.g., dspy.Predict, dspy.ChainOfThought). It specifically focuses on systematic refinement and improvement of precision.

The key interest is to propose and validate a hybrid optimization strategy that seamlessly integrates manual prompt refinement techniques with DSPy-driven automated enhancements. This integrated approach will be rigorously evaluated through comprehensive empirical evaluations, examining accuracy in information extraction, translation quality across languages, training times, and practical applicability in real-world automotive after-sales settings.

In conclusion, these assessments will provide useful insights for applying Prompt Engineering in real-world automotive after-sales industrial settings. The main contributions of this study are as follows:

- Develop a comprehensive baseline prompt for multilingual and unstructured maintenance records from real-world Volvo's Claim table.
- Explore and evaluate multiple manual prompt engineering methods, including Chain-of-Thought reasoning, multilingual adaptation, role-based prompting, and example-based learning, to enhance model accuracy and translation fidelity.
- Implement DSPy-based automatic prompt optimization strategies, experimenting with optimizers such as BootstrapFewShot and BootstrapFewShotWithRandomSearch, and comparing modules like `dspy.Predict` and `dspy.ChainOfThought` to identify the most effective configurations in automotive after-sales industry.
- Propose and verify a hybrid prompt optimization approach on real-world data. The hybrid method integrates the strengths of both manual and automated techniques for balanced performance in information extraction and multilingual translation tasks.
- Focusing on extraction accuracy, translation quality, training efficiency, and operational viability in industrial settings, provide thorough evaluation metrics of all methodologies using real-world Volvo truck maintenance data.
- Compare the performance of a relatively large model (GPT-4o) with a relatively smaller model (LLaMA-33-70B) after our prompt optimization strategy. Assesses whether smaller LLMs can achieve competitive performance through well-designed prompts.
- Provide practical guidance for applying prompt engineering techniques in automotive after-sales contexts. It will help to support the more scalable and effective use of LLMs in traditional automotive industry.

## 2 Literature Review

This chapter provides an extensive review of existing literature and some basic knowledge relevant to the current research. At beginning it introduces the LLM technology, which includes popular LLMs. Subsequently, it indicates the embedding techniques. Next, the related study has been moved to fine-tuning costs and limitation in recent years. Then it concentrates on Prompt Engineering, an important approach within NLP that facilitates the effective use of LLMs without the substantial computational resources, that are traditionally required for fine-tuning. The chapter then reviews some current and practical application of NLP in automotive after-sales service recent years. It mainly focus on challenges and current strategies employed to handle multilingual and unstructured maintenance records which is related to our thesis topic. Evaluation metrics such as ROUGE, BERTScore, cosine similarity and vector similarity are introduced in detail. It indicates their importance for evaluating extraction and translation effect in different aspects. Finally, the review identifies specific research gaps and opportunities, emphasizing the need and benefit for integrated hybrid optimization strategies for real-world automotive after-sales industry records.

### 2.1 Large Language Models (LLMs)

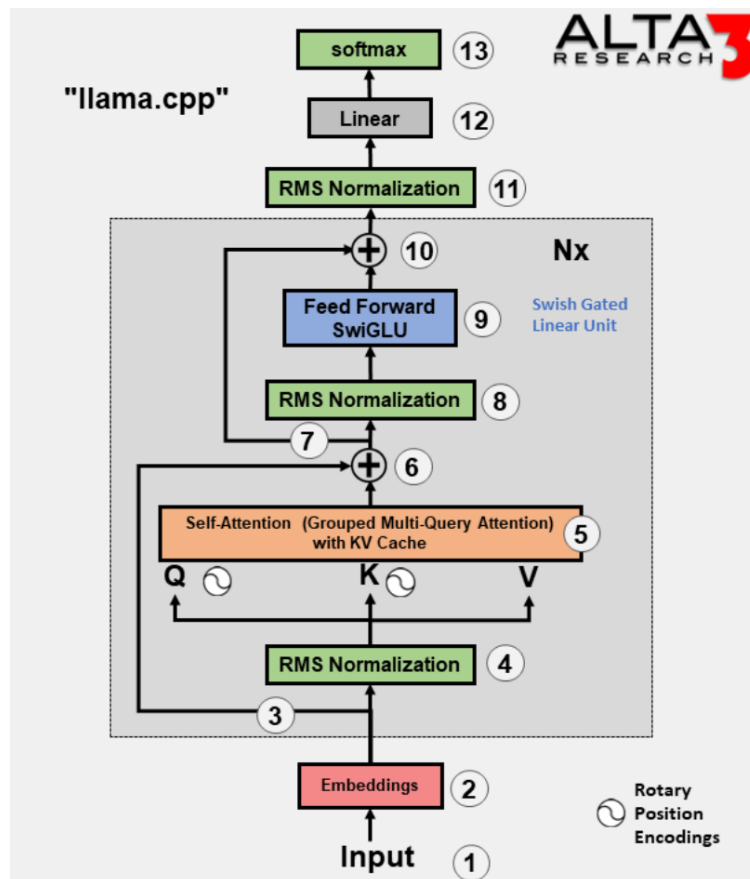
Large Language Models (LLMs) have become the foundation of modern NLP due to their capacity to learn generalizable linguistic and semantic knowledge from massive context. These models, typically based on the Transformer architecture, exhibit impressive capabilities in generation, reasoning, and multilingual understanding. Currently, in Volvo Group the LLM team provides Llama-31-405B, Llama-31-70B, Llama-33-70B, GPT4o, GPT3.5, to deploy. These LLMs have been fine-tuned on other project, which could not be directly used on our project.

#### 2.1.1 Large Language Model Meta AI Model(LLaMA Model)

LLaMA is a family of state-of-the-art open-source large language models (LLMs) developed by Meta AI, which released in February 2023, LLaMA is designed to advance research in natural language processing (NLP) by providing a more accessible and efficient alternative to proprietary models like GPT-3. [13] It provides strong performance with efficient parameter scaling. LLaMA models are particularly suited for fine-tuning and prompt-based adaptation in resource-constrained scenarios. The latest version is LLaMA 3.3, according to Olteanu [14], it shows the state-of-the-art results in multilingual and general-purpose NLP tasks.

The architecture of LLaMA model was opened for reviewing. According to Feeser, CEO of Alta3 Research, Inc [15], LLaMA model has 13 important components, which is illustrated in Figure 3

The LLaMA model starts its process by accepting a prompt made up of tokens extracted from raw text, which provides the foundation for all ensuing computations. Then the tokens are included into high-dimensional vectors with a trained embedding matrix. LLaMA employs positional encoding method, which name is Rotary Positional



**Figure 3:** LLaMA Architecture [Public domain], via Feeser’s blog. (<https://stuartfeeser.com/blogs/ai-engineers/llama-model/index.html>).

Embedding (RoPE). It could help to accurately differentiate between sequences such as “dog bites man” and “man bites dog” [16]. The initial token embeddings are cached promptly for subsequent residual connections, hence preventing unnecessary computation. LLaMA employs RMSNorm [17] in place of LayerNorm for input normalization, ensuring numerical stability and expediting convergence. LLaMA utilizes Grouped Multi-Query Attention (G-MQA), wherein several attention heads share key and value matrices while maintaining distinct queries, hence diminishing memory and computational demands while retaining contextual comprehension. Subsequent to attention, cached embeddings are reintegrated through residual connections, succeeded by an additional RMSNorm layer. The output subsequently through a feedforward network consisting of linear layers and a SwiGLU activation, which integrates Swish and Gated Linear Unit methods to augment non-linearity [18]. Next, Supplementary residuals and cached tokens are employed in a subsequent feedforward layer to enhance the representation further. Finally, A concluding RMSNorm readies the output for the softmax layer, which transforms hidden states into language probabilities, facilitating prediction tasks such as generation or classification.

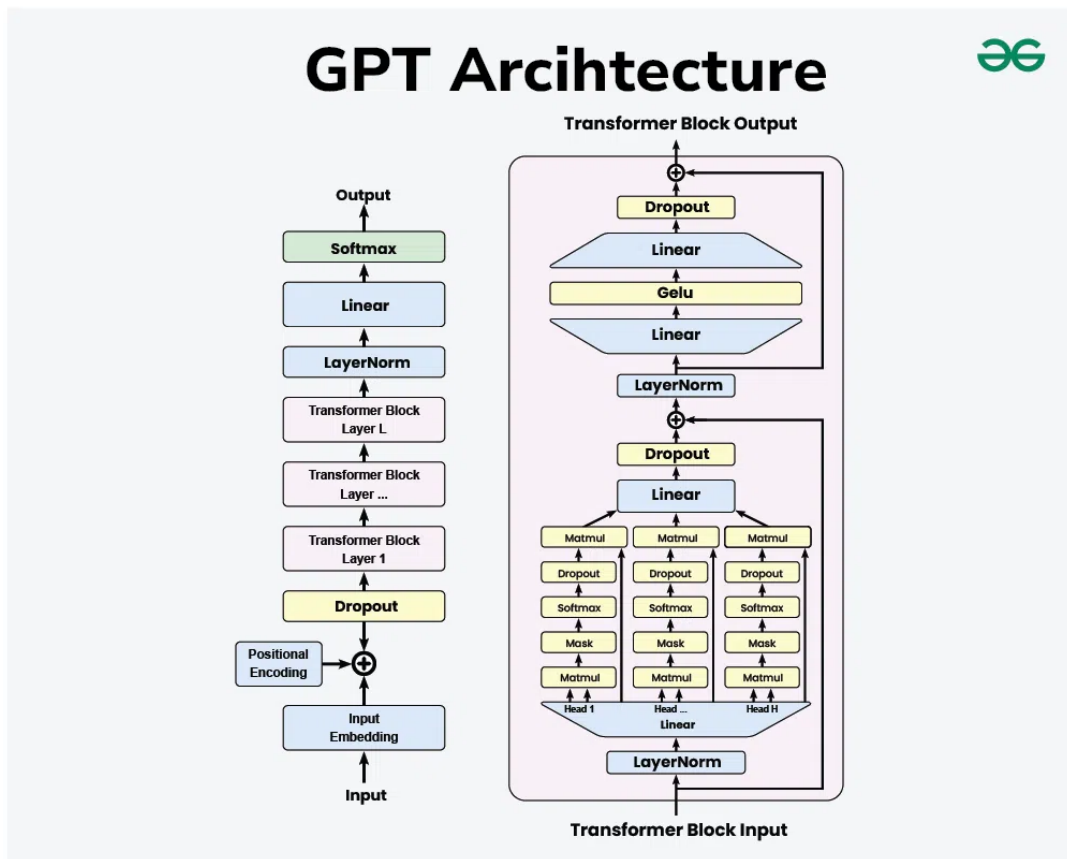
From this structure, LLaMA provides a strong foundation in some NLP tasks. Besides, according to Olteanu [14], LLaMA has been used in multilingual Chatbots,

Coding Support, Research, Knowledge-Based Applications and so on.

### 2.1.2 Generative Pre-trained Transformer (GPT) Model

GPT was carried out by OpenAI, a US-based artificial intelligence research company. Its development is based on a series of improvements in natural language processing (NLP). Based on the famous Transformer architecture introduced by Google in 2017 [19], the OpenAI technical report [20] indicates that the self-attention mechanism of the Transformer revolutionized the handling of sequential data by enabling more effective control of long-range text relationships. Employing this methodology, OpenAI launched the GPT (Generative Pre-trained Transformer) series, beginning with GPT-1 in 2018. Later versions—GPT-2 (2019), GPT-3 (2020), GPT-4 (2023), and GPT-4.5 (2025)—have shown increasingly exceptional text generation capabilities [21].

According to GeeksforGeeks [22], the following outlines the fundamental components of the GPT architecture. Many of the components are structurally similar to the LLaMA model, which we have described earlier. In Figure 4, it shows the detail structure of the GPT Model.



**Figure 4:** GPT Architecture [Public domain], via GeeksforGeeks website. (<https://www.geeksforgeeks.org/introduction-to-generative-pre-trained-transformer-gpt/>).

GPT initiates the process by tokenizing raw text and transforming each token into a high-dimensional vector through a learned embedding matrix, which employs a methodology similar to LLaMA. Then it integrates positional information with the addition of learned positional embeddings. It allows the model to differentiate between various token sequences, which is crucial for syntactic comprehension. To reduce overfitting problems, GPT employs a dropout layer on embeddings and intermediate activations, a regularization technique also prevalent in LLaMA. The primary architecture comprises layered Transformer blocks featuring masked multi-head self-attention. Compared with LLaMA, GPT employs a causal mask to guarantee that each token only attends to the preceding tokens during both training and inference. This mechanism is articulated in Eq. 1

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^\top}{\sqrt{d_k}} + M \right) V \quad (1)$$

where the mask  $M$  blocks future positions, distinguishing GPT from bidirectional models. Following the attention layer, GPT applies residual connections, layer normalization, and a feedforward network. It adopts the GeLU activation function, which differs from LLaMA's SwiGLU, and its equation is described in Eq. 2.

$$\text{FFN}(x) = \text{GeLU}(xW_1 + b_1)W_2 + b_2 \quad (2)$$

The model stacks multiple such layers to capture deep hierarchical representations. After the final Transformer block, a concluding LayerNorm is applied, and the output is projected into the vocabulary space using a learned weight matrix. A softmax function is then used to produce the probability distribution for the next token, a process structurally aligned with LLaMA's final step.

Together with LLaMA, the GPT model offers a flexible and scalable language modeling system. GPT sets itself apart by using causal masking, GeLU activations, and instruction tuning, even although both share similar architectural ideas including self-attention, residual connections, and layer stacking. By means of prompt-based learning, these design decisions help GPT to shine in autoregressive generation, conversational tasks, and few-shot adaption. In addition, according to Cotton [23], GPT has widely used in data analysis and coding tasks, roleplay, translation fields.

## 2.2 Embedding Techniques for Semantic Representation

Embedding methods convert discrete language units—such words, phrases, or sentences—into continuous vector representations in a high-dimensional space. Important latent semantic, syntactic, and contextual links captured by these deep embeddings support natural language understanding and creation.

Embeddings by encoding relational proximity in vector space enable similarity computations, grouping, and downstream reasoning unlike symbolic representations (e.g., one-hot vectors). Foundational in almost all neural NLP systems, including transformer-based models, embeddings are also fundamental input or intermediate representations.

### 2.2.1 Instruction-Tuned Embeddings

Recent advancements include instruction-tuned models such as OpenAI Ada Embedding (text-embedding-ada-3-large) and BAAI General Embedding (BGE-M3). These models are specifically designed for alignment with user tasks such as retrieval, classification, and question answering.

1. **Ada Embedding:** Developed by OpenAI, text-embedding-ada-3-large supports up to 8191 tokens and produces 3072-dimensional vectors, which was mentioned in Briggs article [24].

It is instruction-aligned, meaning it can encode task-specific intent and supports multilingual inputs. It is widely used in retrieval-augmented generation (RAG) and semantic search systems. According to Briggs [24], it showed an improvement assessed by the increase from 31.4% to 54.9% on the MIRACL benchmark. In terms of English-language performance, MTEB benchmark indicates a modest yet notable gain from 61% to 64.6%, which was compared with the text-embedding-ada-002.

2. **BGE-M3:** Released by Beijing Academy of Artificial Intelligence (BAAI), BGE-M3 is a multilingual, multitask, and multi-granularity embedding model. According to Chen’s team [25], it supports fine-grained query-document matching, works across 100+ languages, and is effective in document retrieval, reranking, and even zero-shot classification.

### 2.2.2 Mathematical Formulation

Let a sentence  $S = \{w_1, w_2, \dots, w_n\}$  consist of  $n$  tokens. A pre-trained model  $f$  maps each token  $w_i$  to a contextualized vector  $f(w_i) \in \mathbb{R}^d$ . The embedding representation of the entire sentence can be computed by pooling over the token vectors, commonly via mean pooling:

$$E_S = \{f(w_1), f(w_2), \dots, f(w_n)\} \quad (3)$$

$$\text{Embed}(S) = \frac{1}{n} \sum_{i=1}^n f(w_i) \quad (4)$$

where:

- $S$ : input sentence consisting of tokens  $\{w_1, w_2, \dots, w_n\}$ ,
- $f(w_i) \in \mathbb{R}^d$ : contextualized embedding of token  $w_i$ ,
- $d$ : embedding dimensionality,
- $E_S$ : set of token embeddings for sentence  $S$ ,
- $\text{Embed}(S)$ : final sentence embedding computed by averaging token vectors.

Pooling methods on Ada and BGE-M3 are different. Ada utilize max pooling principle, while BGE-M3 combines pooling with multi-task fine-tuning objectives.

### Ada Embedding Formula

$$\text{Embed}_{\text{Ada}}(S) = \text{TransformerEncoder}(\text{Tokenize}(S)) \in \mathbb{R}^{3072} \quad (5)$$

where:

- $S$ : input text (document, query, or title),
- $\text{Tokenize}(S)$ : OpenAI tokenizer output (up to 8191 tokens),
- $\text{TransformerEncoder}(\cdot)$ : internal transformer layers,
- Output vector is a 3072-dimensional dense embedding.

### BGE-M3 Embedding Formula

$$\text{Embed}_{\text{BGE-M3}}(S) = \frac{1}{n} \sum_{i=1}^n f(w_i) \in \mathbb{R}^{1024} \quad (6)$$

where:

- $S = \{w_1, \dots, w_n\}$ : input sentence tokens,
- $f(w_i) \in \mathbb{R}^{1024}$ : token-wise embedding generated by BGE-M3,
- Output is a 1024-dimensional embedding vector,
- Pooling strategy may include task-conditioned weighting.

## 2.3 Fine-Tuning Costs and Limitations

Fine-tuning is the process of adapting a pre-trained Large Language Model to a specific downstream task by updating all or part of its parameters on task-specific data. This approach allows the model to capture domain-specific knowledge, style, and task requirements beyond its general-purpose pretraining. In domains such as automotive after-sales, where the language used in maintenance records is highly specialized and multilingual, fine-tuning seems like a viable solution to improve performance on information extraction and translation tasks.

However, fine-tuning introduces several challenges that limit its feasibility in real-world industrial settings, particularly for companies lacking large-scale computational infrastructure.

### 2.3.1 Fine-Tuning Costs

Fine-tuning a LLM for this specific area is expensive. According to Xia’s research [3], it indicated that Mixtral model with data size of 2M queries fine-tuned by NVIDIA H100 GPU which will cost \$3460 approximately. Another thesis from Zheng [26], it shows the fine-tuning time for a LLM with 6758M parameters could be around 48.4 hours per epoch with NVIDIA H100 GPU. Another result given by Bybradbury [27], it cost about 37 hours to fine-tune the Google’s Flan-PaLM with 540B parameters on 512 TPU v4 chips. Dettmers’ team [28] found out even with an effective method to fine-tune an LLM with 65B parameters, which will still need 24 hours on a single 48 GB GPU. Hence, fine-tuning LLMs can be computationally expensive and time-consuming. These examples illustrate that the training cost grows rapidly with model size.

### 2.3.2 Operational Challenges

In addition to hardware needs, fine-tuning LLMs face major operational issues in real-world industrial sectors, particularly in the automobile sector. Large amounts of high-quality, domain-specific labeled data are necessary for fine-tuning. Currently in actual industrial settings, most of the time this resource is often limited and dispersed. For example, according to Chen’s [29] project, maintenance logs are usually private, unstructured, and unreliable, which makes them expensive to get and preprocess for purposes of fine-tuning.

Additional complexities are introduced by the annotating process itself. It needs to take precise labels that are in line with automobile taxonomies to fine-tune a model for fault classification or issue summary. Pavlopoulos [30] pointed out that Label consistency becomes more challenging to attain in multilingual truck maintenance records, as entries may be authored in a variety of languages, dialects, or with abbreviations. The fine-tuning overhead is further increased by these difficulties, which frequently necessitate the use of domain-expert annotators and comprehensive annotation rules. [31].

Furthermore, fine-tuned models frequently exhibit fragility when implemented in dynamic industrial settings. Any alteration in downstream tasks—such as the introduction of new car models, revised diagnostic codes, or regional nomenclature—can rapidly diminish the model’s efficacy. Consequently, fine-tuned LLMs necessitate regular re-tuning to accommodate fresh data distributions, rendering them costly to sustain and constraining their scalability and long-term reusability [32]. In a specific industrial-scale implementation, more than half million labeled data across 38 languages were essential for training a multilingual truck fault diagnostic model, necessitating re-finetuning anytime fault categories changed [30]. These facts underscore that although fine-tuning may yield immediate performance improvements, its operational requirements render it a less feasible long-term strategy in rapidly changing industrial environments.

## 2.4 Current Prompt Engineering Work in Information Extraction and Multilingual Applications

With the rapid development of LLMs, prompt engineering has emerged as a central technique for information extraction tasks. Compared to traditional supervised learning approaches, prompt-based methods allow flexible, zero-shot, or few-shot extraction of structured knowledge from raw text, without the need for task-specific model architectures or large-scale annotated datasets. This section reviews the recent advances in prompt engineering applied to information extraction tasks, as well as its emerging applications in the multilingual and industrial domains, particularly the automotive sector.

### 2.4.1 Prompt Engineering for Information Extraction

Recent studies have shown that LLMs, when guided by well-designed prompts, can effectively perform various information extraction tasks such as named entity recognition (NER), relation extraction (RE), and event extraction.

A method named Generative information extraction has gained popularity, with models like GenIE framing relation and entity extraction as text-to-text generation tasks. Josifoski’s team [33] carried out this method in 2022. GenIE demonstrated that LLMs can help to generate subject-relation-object triples directly from text with state-of-the-art results using autoregressive prompting. Specifically, GenIE achieved a 13 point improvement in micro-recall and an 11 point improvement in macro-recall on the FewRel dataset compared to baseline models. Similarly in Kan’s study [34], prompt-based methods allow the output of structured JSON data, which can reduce the complexity of separate components for each information extraction subtask.

In the domain of Named Entity Recognition (NER), Shen’s team [35] proposed a method named PromptNER reimagines NER as a locating and typing problem handled via a single prompt template. It delivers significant gains in few-shot and low-resource settings. In their study, PromptNER made progress in absolute improvements in the F1 score of 4% in the CoNLL dataset, 9% in GENIA, 4% in FewNERD, 5% in FaBNER and 24% in TweetNER. Additionally, it performed better than previous methods in the cross-domain NER. They set new benchmarks on 3 out of 5 CrossNER target domains with an average F1 gain of 3%, despite using less than 2% of the available data.

In the field of Relation extraction (RE), it has also benefited from prompt engineering. A study carried out by Chia [36] demonstrated a method which is named RelationPrompt, it uses LLMs to generate synthetic training data for zero-shot RE tasks, enabling effective extraction of novel relations without manual annotation. This approach introduces the task setting of Zero-Shot Relation Triplet Extraction (ZeroRTE), allowing models to extract relation triplets where the relation labels are not seen during training, thus encouraging further research in low-resource relation extraction methods.

In the area of Event extraction, it has adopted techniques like prompt chaining which was brought out by Kwak [37]. The models first classify event types before extracting specific arguments. This method improves precision and scalability in

complex domains such as law and healthcare.

## 2.4.2 Prompt Engineering in Multilingual Applications

As language models evolve to include multiple languages, prompt engineering provides a language-neutral method for information extraction in various linguistic environments.

Prompt-XRE a method proposed by Hsu [38] in 2023, it outperforms baseline models on the ACE05 dataset by enabling multilingual relation extraction using both soft and hard prompt templates. In addition, Prompt-XRE achieves state-of-the-art performance in the cross-lingual relation extraction field. It outperforms vanilla multilingual PLMs and other models in experiments on the low-resource ACE05 benchmark across several languages.

Multilingual Named Entity Recognition (NER) has also been explored by Hsu through prompt engineering. Studies like MultiCoNER show that, in low-resource languages without language-specific training, prompting can help to enable correct entity recognition.

In industrial applications, especially the automotive domain, prompt-based LLMs have been applied to extract structured insights from large-scale unstructured multilingual data. For example, a study in 2024, it uses GPT models to extract up to 15 structured fields from road crash tweets. In Jaradat's study [39], it analyzed over 25000 traffic-related tweets from Australia. This study found that a prompt-tuned GPT-2 model achieved an average accuracy about 85% across six classification tasks. It has been proven that it exceeds the baseline GPT-4o mini model's 64% and XGBoost's 83.5% respectively. Other work focused on parsing civil aircraft accident reports using a prompt-driven pipeline, demonstrating the effectiveness of LLMs in extracting and analyzing textual information from aviation safety reports [40].

These examples show how prompt engineering allows flexible multilingual and scalable information extraction systems across fields and languages with limited labeled data.

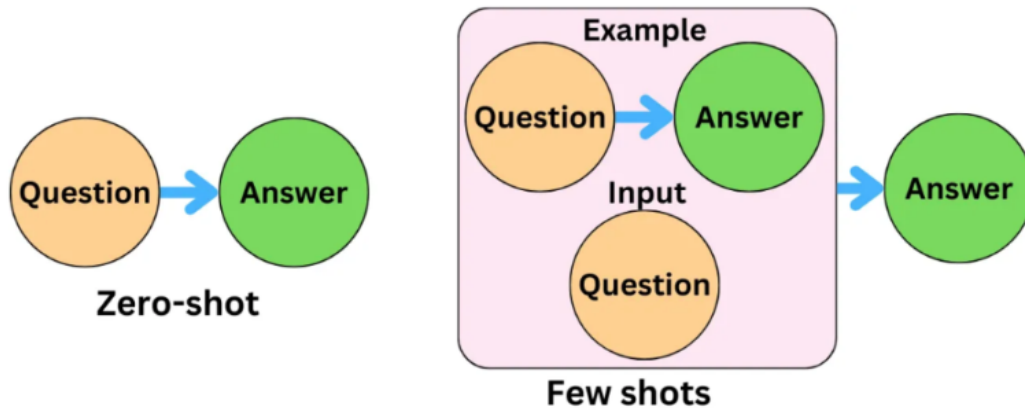
## 2.5 Manual Prompt Engineering in NLP

Manual prompt engineering is the human-driven process of designing, testing, and improving prompts explicitly to directly influence LLM effects on specific tasks. It builds good instructions or examples using domain knowledge, intuition, and iterative feedback. Although this process needs lots of time and repeated debugging, this strategy provides transparency, and task-specific customizing absent of autonomous approaches.

### 2.5.1 Zero-shot and Few-shot Prompting

According to Brown [41], zero-shot prompting involves instructing a language model to perform a task without giving any examples in the prompt. The model must generalize from the task description or question alone. In contrast, few-shot prompting provides a handful of example input-output pairs in the prompt, followed by a new

query of the same type. This technique leverages in-context learning, allowing large pretrained LLM models to infer the task and format from the examples. The difference of Zero-shot and Few-shot can be seen from Figure 5.



*Zero-Shot and Few-Shot Prompting*

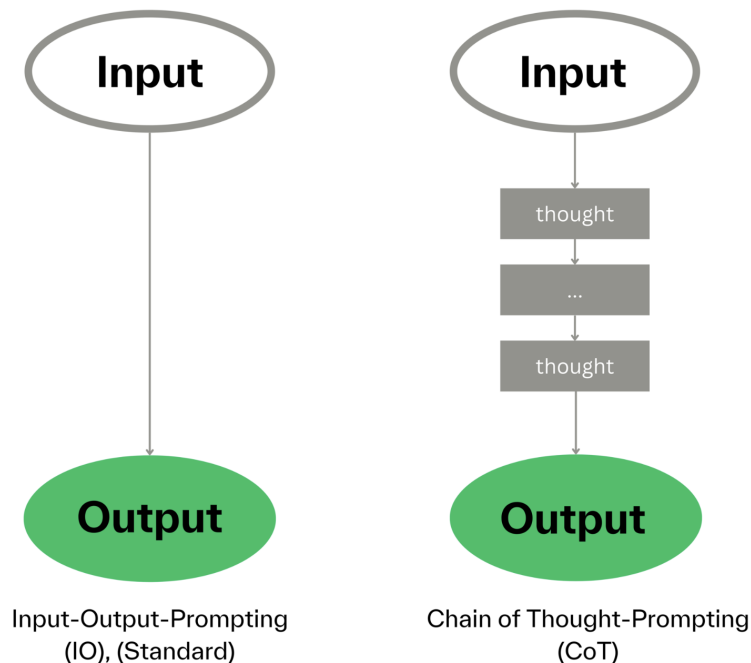
**Figure 5:** Zero-Shot and Few-Shot Prompting [Public domain], via weclouddata website. (<https://weclouddata.com/blog/few-shot-and-zero-shot-prompting/>).

Few-shot prompts often yield significantly better performance than zero-shot prompts on many NLP tasks, especially with very large models. Notably, Brown’s research [41] showed GPT-3 can be conditioned purely via text prompts with up to 100 examples and achieve near state-of-the-art results on translation, QA, and other tasks without any gradient updates. The benefit is that no task-specific fine-tuning is required; however, prompts can become long and the model may be sensitive to the choice of examples and wording. In practice, prompt designers must carefully select demonstrative examples that cover the task’s scope, as irrelevant or too few examples can lead to poor results. A limitation is that very large context windows are needed for many examples, and performance gains plateau beyond a certain number of shots.

### 2.5.2 Chain-of-Thought(CoT) Prompting

Chain-of-thought prompting is a prompt engineering method where the model is guided to produce explicit intermediate reasoning steps before giving the final answer, this idea was presented in Wei’s research [42]. Rather than replying immediately with an answer, the model is prompted to “think step by step.” For instance, a prompt might include a few exemplars where the solution is reached through a series of logic or arithmetic steps, encouraging the model to do the same for new queries, this structure can be seen in Figure 6. This method is especially useful for complex tasks like math word problems, logical reasoning, or commonsense questions that benefit from decomposing the problem. By externalizing a reasoning chain, LLMs can tackle problems that are otherwise too difficult with a single forward pass. Empirical results have shown significant performance gains: Wei [42] demonstrated that with CoT prompting, a 540B parameter model achieved state-of-the-art accuracy on the GSM8K

math word problem benchmark, even outperforming a fine-tuned verifier-enhanced model. The notable benefit is improved accuracy on multi-step reasoning tasks, as the model can correct itself or break down the problem.



**Figure 6:** Input output Prompting vs Chain of Thought Prompting[Public domain], via technische hochschule augsburg website. (<https://www.tha.de/en/Types-of-prompting.html>).

According to Zhang [43], a potential limitation is that the prompt becomes longer and more complex. Besides, not all tasks benefit from verbalized reasoning, and forcing a chain-of-thought can sometimes introduce errors if the model’s “thinking” drifts off-track. CoT prompting works best with sufficiently large models that have the capacity to perform reasoning when guided appropriately.

### 2.5.3 Role-based Prompting

Role-based prompting guides an AI model’s behavior by assigning it a specific persona or role through the prompt. The idea is to preface the query with a description such as ‘You are an expert in Volvo’s Truck Maintenance recorder’ or ‘Act as a professional multilingual translator,’ which establishes context and desired style for the response. By adopting a role, the model can tailor its vocabulary, tone, and the types of explanations it gives to better fit the expected persona.

This method is widely used in conversational AI (e.g., system messages in ChatGPT define the assistant’s role). Recent research suggests that role prompts can also improve reasoning performance. For example, instructing the model “Pretend you are a mathematician” can coax it into a more rigorous, step-by-step problem-solving mode. Kong [44] introduced a “role-play prompting” strategy and found it consistently

outperformed standard zero-shot prompting across a battery of reasoning benchmarks. In their zero-shot tests, simply assigning a role (e.g., “You are a knowledgeable assistant”) boosted accuracy significantly – e.g., on a math QA dataset, accuracy jumped from 53.5% to 63.8% with an appropriate role prompt.

The benefit of role-based prompting is better control over the style and potential strategies used by the model. A limitation is that an inappropriate persona might inject biases or distract the model. Careful selection of the role is therefore important, and it often requires experimentation to find a helpful one for a given task.

#### **2.5.4 Manual Iterative Prompt Design**

In many specific industrial cases, prompt engineering necessitates manual iteration beyond predefined templates. In an effort to elicit more effective responses, practitioners experiment with prompt phrasing, formatting, task structuring, and the ordering of examples. Although this process is time-consuming and challenging to generalize, it is a practicable solution for niche tasks in which domain knowledge is essential or data is scarce.

## **2.6 Automatic Prompt Engineering in NLP**

Automatic prompt engineering maximizes prompts with little human involvement by using algorithmic and machine learning-driven methods. These techniques are meant to improve scalability and efficiency especially in situations when manual tuning is impractical and time-consuming. Although these methods provide speed and automation, occasionally they lack the subtle adaptability of human-designed methods.

### **2.6.1 DSPy-based Prompt Optimization**

DSPy-based optimization, proposed by Khattab and his research group in 2024 [45]. It offers a structured and modular approach to automated prompt tuning. A key method within this framework, *BootstrapFewShotWithRandomSearch*, refines prompt templates by randomly sampling prompt candidates and selecting those that yield optimal model performance. In addition, user can choose whether they want to utilize the *Chain-of-Thought* mode. This allows users to balance accuracy, reducing the cost of inference while maintaining effectiveness.

With a few minutes of automatic compilation, Khattab’s research [45] shows DSPy can generate prompt pipelines that outperform human-engineered prompts for both big proprietary models (like GPT-3.5) and smaller open models.

Additionally, DSPy employs heuristic techniques to effectively locate suitable prompts as its search space might be huge; yet, there is still computational cost compared to hand crafting one suggestion. By "compiling" prompts from a higher-level design, optimizing them using data-driven feedback, DSPy generally shows a shift toward automated prompt engineering, hence enabling state-of-the-art solutions without exhaustive manual trial-and-error.

## 2.6.2 Reinforcement Learning-based Prompting

Reinforcement learning can be applied in prompt engineering by considering the prompt as the policy to be optimized, which was explained by Cleary [46]. In this setting, achieving a good effective output from the language model for a certain task is the key to receiving a significant reward. The agent will use an optimization algorithm to execute incremental alterations to the prompt and observes the output of the LLM alongside a corresponding reward signal (e.g., +1 for a correct response). The prompt is iteratively modified to maximize cumulative reward.

RLPrompt was come out by Deng who is from Carnegie Mellon University [47], this method learns discrete text prompts using policy gradient algorithms. Their approach determines the reward depending on the desired task performance (e.g., whether the output of the model is accurate) and starts a prompt—a series of tokens. Then, iteratively changing the prompt tokens, an RL algorithm—such as REINFORCE or proximal policy optimization—explores novel variants and uses those that produce more rewards. By means of training, the strategy converges to an optimal prompt. This has been demonstrated to provide prompts that can considerably increase even style transfer tasks and few-shot categorization accuracy. A curious finding is that the optimized prompts are often unintelligible strings (e.g., seemingly random) yet they consistently trigger better model behavior. This underscores that the RL process can uncover prompt patterns not obvious to humans.

Especially when the "output" could comprise a succession of interactions, the advantage of RL-based prompting is a strong search that can traverse challenging prompt areas and maximize for long-term or multi-step targets. It also fits well when a differentiable aim is not accessible but a reward can be obtained—even from human feedback or a learned critic model.

## 2.7 Evaluation Metrics for Extraction and Translation

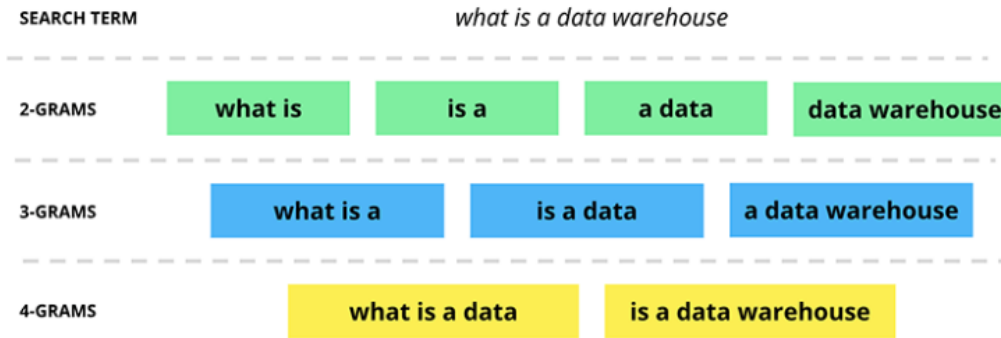
A range of evaluation criteria is used to methodically evaluate the quality of information extraction and translation generated via quick engineering. These comprise semantic similarity measurements like BERTScore and Cosine Similarity, lexical overlap metrics like ROUGE, and embedding-based representations built from Sentence-BERT or related models. These measures are thoroughly discussed in the following sections together with their mathematical formulations, theoretical underpinnings, application settings, and restrictions.

### 2.7.1 ROUGE Scores for Lexical Overlap

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a family of metrics commonly used for evaluating text summarization and machine translation, which was first introduced by Lin [48] in 2004. It measures the similarity between a generated text and one or more reference texts by tallying overlapping units such as n-grams, word sequences, or longest common subsequences. There are typically three types of ROUGE versions.

## 1. ROUGE-N

ROUGE-N evaluates the overlap of contiguous sequences of  $n$  words (n-grams) between candidate and reference texts. It reflects the precision, recall, and F1 score of the n-gram matches. The n-grams are the basic building blocks which are often used in the NLP field. The Figure 7 illustrates the example of n-grams.



**Figure 7:** What is an n-gram [Public domain], via funnel website. (<https://funnel.io/blog/n-gram-analysis>).

$$P_n = \frac{\text{Number of overlapping n-grams}}{\text{Number of n-grams in candidate}}, \quad R_n = \frac{\text{Number of overlapping n-grams}}{\text{Number of n-grams in reference}} \quad (7)$$

$$F_n = \frac{(1 + \beta^2) \cdot P_n \cdot R_n}{\beta^2 P_n + R_n} \quad (8)$$

where:

- $P_n$ : precision of n-gram overlap,
- $R_n$ : recall of n-gram overlap,
- $F_n$ : harmonic mean of precision and recall (F1-score),
- $\beta$ : weighting factor (commonly set to 1 for balanced F1),
- The numerator quantifies overlapping n-grams, whereas the denominators quantify the total number of n-grams in the candidate and reference, respectively.

Typically,  $\beta = 1$  to balance precision and recall equally. ROUGE-1 and ROUGE-2 (unigrams and bigrams) are the most commonly reported. While ROUGE-N is simple and efficient, it is limited in capturing semantic similarity or sentence structure.

## 2. ROUGE-L

ROUGE-L finds the Longest Common Subsequence (LCS) between the candidate and reference texts. As long as the order is maintained, it accounts for in-sequence matching and captures the sentence-level organization, hence enabling non-contiguous token alignment.

$$R_{lcs} = \frac{LCS(X, Y)}{m}, \quad P_{lcs} = \frac{LCS(X, Y)}{n} \quad (9)$$

$$F_{lcs} = \frac{(1 + \beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2P_{lcs}} \quad (10)$$

where:

- $LCS(X, Y)$ : length of the longest common subsequence between reference  $X$  and candidate  $Y$ ,
- $m$ : length of the reference text  $X$ ,
- $n$ : length of the candidate text  $Y$ ,
- $R_{lcs}$ : recall of LCS-based overlap,
- $P_{lcs}$ : precision of LCS-based overlap,
- $F_{lcs}$ : harmonic mean of  $R_{lcs}$  and  $P_{lcs}$ ,
- $\beta$ : weighting factor (commonly set to 1 for balanced F1-score).

According to Santhosh [49] ROUGE-L is particularly useful in tasks where fluency and grammaticality are essential, such as document summarization.

## 3. ROUGE-W (Weighted LCS)

ROUGE-W expands ROUGE-L by giving longer, contiguous subsequence matches greater weights. It adds a decay factor punishing non-contiguous or broken matches. This more faithfully and consistently expresses fluency.

ROUGE-W is less often employed because of its computational complexity, but in jobs that give continuity top priority, such as tale production or human-like summarizing, it can more accurately reflect human judgment.

### 2.7.2 BERTScore for Semantic Similarity

Zhang’s team from Cornell University [50], introduced the BERTScore method in 2019 to evaluate the text generation. BERTScore evaluates similarity by analyzing contextual embeddings of candidate and reference texts. In contrast to superficial metrics, it evaluates the semantic alignment between two text segments, regardless of vocabulary discrepancies.

BERT and other pre-trained language models create context-based embeddings for every token. These token embeddings are subsequently cosine similarity matched

across texts. BERTScore computes precision, recall, and F1 by finding the most comparable token in the reference for every token in the candidate text—and vice versa.

$$P_{\text{BERT}} = \frac{1}{|y|} \sum_{y_j \in y} \max_{x_i \in x} \cos(e_{x_i}, e_{y_j}) \quad (11)$$

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{y_j \in y} \cos(e_{x_i}, e_{y_j}) \quad (12)$$

$$F_{\text{BERT}} = 2 \cdot \frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \quad (13)$$

where:

- $x = \{x_1, x_2, \dots, x_m\}$ : the set of tokens in the reference text,
- $y = \{y_1, y_2, \dots, y_n\}$ : the set of tokens in the candidate text,
- $e_{x_i}$ : contextual embedding vector of token  $x_i$ ,
- $e_{y_j}$ : contextual embedding vector of token  $y_j$ ,
- $\cos(e_{x_i}, e_{y_j})$ : cosine similarity between embeddings  $e_{x_i}$  and  $e_{y_j}$ ,
- $P_{\text{BERT}}$ : precision, average of maximum similarities for each candidate token to reference,
- $R_{\text{BERT}}$ : recall, average of maximum similarities for each reference token to candidate,
- $F_{\text{BERT}}$ : harmonic mean (F1) of precision and recall.

According to Zhang [50], BERTScore is well-suited for evaluating paraphrasing, generation quality, or translation when exact wording may vary but meaning is preserved. Its main limitations are computational cost and sensitivity to tokenization differences.

### 2.7.3 Cosine Similarity of Sentence Embeddings

Cosine similarity is a vector-based metric used to compute the angle between two vectors in high-dimensional space, which has been published by Salton’s team [51] in 1975. It provides a scalar value in the range  $[-1, 1]$ . It indicates how similar two vectors (texts) are in direction.

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^d A_i B_i}{\sqrt{\sum_{i=1}^d A_i^2} \sqrt{\sum_{i=1}^d B_i^2}} \quad (14)$$

where:

- $A = [A_1, A_2, \dots, A_d]$ : embedding vector of the reference sentence in  $\mathbb{R}^d$ ,
- $B = [B_1, B_2, \dots, B_d]$ : embedding vector of the candidate sentence in  $\mathbb{R}^d$ ,
- $A \cdot B = \sum_{i=1}^d A_i B_i$ : dot product of the two vectors,
- $\|A\| = \sqrt{\sum_{i=1}^d A_i^2}$ : L2 norm (magnitude) of vector  $A$ ,
- $\|B\| = \sqrt{\sum_{i=1}^d B_i^2}$ : L2 norm of vector  $B$ ,
- $d$ : dimensionality of the embedding space.

Models like Sentence-BERT or Universal Sentence Encoder are typically used to generate  $A$  and  $B$ . Cosine similarity excels in multilingual applications and semantic search but does not directly account for grammatical correctness or word order.

## 2.8 Research Gaps and Opportunities

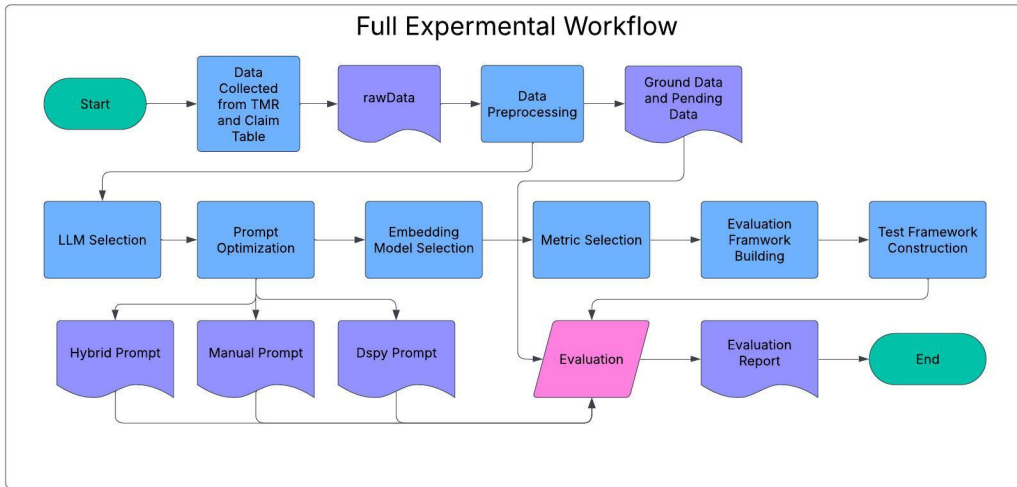
Although manual and automated prompt engineering techniques have shown promise, their fit into automotive after-sales industrial environments is yet to be explored deeply. While most current studies concentrate on either manual or automated approaches independently, few research on hybrid optimization strategies combining both approaches. Further study is presented by the necessity for a methodical evaluation of different approaches in practical situations including vehicle after-sales service.

Furthermore, prompt engineering research mainly focuses on generic NLP applications, so leaving a void in domain-specific adaption for specialized sectors as vehicle diagnostics and maintenance records. The question of how prompt engineering can be tailored for these particular situations is still a difficult one.

Scalability and deployment feasibility also require further investigation. Practical acceptance of prompt-based systems depends on their capacity to remain consistent and interpretable across a range of real-world environments. By filling up these research gaps, NLP and Prompt Engineering applications in the automotive sector could be improved, in the meantime, it could improve efficiency and lower running costs in after-sales service management.

## 3 Methodology

This chapter describes the methodology used in this study, including data preprocessing, test framework construction, prompt optimization strategies, and evaluation methods. The LLaMA-33-70B model, which has been fine-tuned for another domain, is adapted for multilingual information extraction and translation tasks in truck maintenance records. The Figure 8 illustrates the full experimental workflow.



**Figure 8:** Full experimental workflow

### 3.1 Data Collection and Preprocessing

In this study, we leverage two primary datasets provided by Volvo Group: the **Truck Maintenance Records (TMR)** table and the **Claim** table. These datasets capture structured and unstructured after-sales maintenance information originating from a wide range of vehicle service activities.

#### 3.1.1 Claim Table as Ground Truth

The Claim table consolidates structured repair comments obtained from Volvo’s B2B partners, encompassing insurance firms and authorized repair workshops. A major column in this table, labeled `detail`, has multilingual technician comments that generally encompass three essential components—Complaint, Cause, and Correction—integrated in free-text format.

This dataset has been translated and information has been extracted by a group of experienced professional Volvo internal translators in after-sales domain. The English translation is contained in the `Translation` column, whereas the extracted and translated parts are individually recorded in the `Complaint`, `Cause`, and `Correction` columns. In Table 1, it demonstrates the key parts of Claim table.

The Claim table functions as a superior ground truth standard for assessing the efficacy of our prompt-based extraction and translation workflow.

#### 3.1.2 TMR Table as Target Corpus

The Truck Maintenance Records (TMR) table covers both B2B and personal customer repair records, the TMR table is a larger and more all-encompassing collection. It has a `detail` column with raw technician comments, which is similar to the Claim table. However, the TMR table doesn’t have any extracted semantic structure or current translations to English.

**Table 1:** Structure of the key parts of Claim table used for ground truth.

Column	Description
service_id	Unique identifier for each service event or repair operation.
truck_id	Unique identifier for the vehicle associated with the repair.
service_date	Date when the repair event occurred.
detail	Raw technician comments written in multiple languages. Typically includes a combined sequence of Complaint, Cause, and Correction as a single free-text field.
Translation	English translation of the detail field, automatically translated from multilingual sources.
Complaint	Extracted user Complaint from the detail text. Describes the problem reported by the customer.
Cause	Extracted root cause of the issue based on technician analysis. Sometimes may be missing or abbreviated.
Correction	Extracted correction action or repair step taken to resolve the issue. May include part replacements or reprogramming.

The TMR dataset serves as the principal target for model inference due of its scale and representativeness. We utilize large language models (LLMs) augmented with rapid engineering techniques to execute both automatic translation and information extraction from the detail column. In Table 2, it shows key part of the TMR table, which will mainly be used in our research.

**Table 2:** Structure of the TMR (Truck Maintenance Records) table used for model inference.

Column	Description
service_id	Unique identifier for each service event or repair operation.
truck_id	Unique identifier for the vehicle (e.g., VIN number).
service_date	Date when the vehicle was serviced or repaired.
country_code	Country where the repair was conducted. Used to infer potential language in detail.
detail	Free-text technician comments written in various languages, containing a mix of Complaint, Cause, and Correction. This field is the primary target for extraction and translation.

### 3.1.3 Preprocessing the Claim Table

The Claim table serves as a ground truth reference for assessing translation and information extraction tasks, comprising millions of authentic entries provided by Volvo’s business-to-business (B2B) partners. These encompass authorized service workshops, insurance providers, and fleet maintenance suppliers. Notwithstanding the

extraction pipeline implemented for Claim, discrepancies and partial values persist, requiring meticulous preparation prior to model application.

There are several preprocessing steps to handle the Claim table to ensure its quality for the next stages to use.

1. **Missing or Corrupted Field Handling:** Entries where Complaint, Cause, or Correction equals just a single dot (".") are assumed to be failed extractions and replaced with "unknown".
2. **Null Replacement:** Any null values in these structured fields are also replaced with "unknown" to prevent DSPy crashes or malformed prompts.
3. **Deduplication:** Records are filtered to remove exact duplicates based on all 4 columns: detail, Complaint, Cause, and Correction.
4. **Multilingual Diversity Retention:** Entries are retained from various languages (e.g., Spanish, Swedish, German, Chinese, Japanese, French, English) to reflect real-world usage and evaluate multilingual robustness.
5. **Balanced Sampling:** 1000 diverse records are sampled to include:
  - Multiple languages;
  - Cases where Complaint, Cause, or Correction are inherently missing and replaced with "unknown";
  - Varying comment length and structure;
  - Different combinations of missing or valid extractions.

This curated subset of the Claim table forms the foundation for all prompt engineering, model testing, and performance benchmarking conducted in later stages of this work.

### 3.1.4 Dataset Splitting for Model Training and Evaluation

The curated 1000 rows from the Claim table are split into:

- **Training Set (350 rows):** Used for instruction-tuning and prompt optimization via DSPy.
- **Validation Set (150 rows):** Used for prompt evaluation and early stopping.
- **Test Set (500 rows):** Held out entirely to assess generalization performance.

This clean, labeled data subset ensures quality evaluation while accounting for linguistic and structural noise typical in real-world industrial datasets.

### 3.1.5 TMR Subset Sampling

To evaluate the model’s performance on unseen, unstructured data, we extract 500 rows from the TMR table that correspond to the Claim 500-row subset. These 500 rows are exactly the union of the  $10 \times 50$  test subsets used later for significance tests. This alignment is based on approximate matches across:

- Service IDs (`service_id`) and hashed keys;
- Timestamps (`service_date`) within a small window;

This ensures fair evaluation by comparing LLM predictions on noisy multilingual TMR input against the Claim ground truth.

## 3.2 Model Selection

To assess the efficacy of prompt engineering independent of conventional fine-tuning, we examined a variety of Volvo internal deployed LLMs with diverse parameter ranges and different purposes. The objective was not merely to choose the most potent model, but to examine if a small sized LLM model might attain robust performance in a zero-shot or few-shot context utilizing prompt optimization techniques.

**Table 3:** Comparison of candidate models for prompt-only evaluation

Model	Parameters	Version	Access	Adaptation Strategy
GPT-3.5	~175B	-	Azure API	Prompt only
<b>GPT-4o</b>	>175B	-	Azure API	<b>Prompt only (benchmark)</b>
LLaMA-31-405B	405B	3.1	Volvo ML Team	Prompt only
LLaMA-31-70B	70B	3.1	Volvo ML Team	Prompt only
<b>LLaMA-33-70B</b>	70B	3.3	Volvo ML Team	<b>Prompt only (selected)</b>

GPT-3.5 and GPT-4o were utilized through the Azure OpenAI API as internal commercial benchmarks, and the LLaMA-based models were fine-tuned internally within Volvo for distinct, unrelated purposes. These models were not trained on the current multilingual repair log extraction task; therefore, their performance here demonstrates zero-shot generalization achieved solely through prompt engineering.

Despite the availability of the huge LLaMA-31-405B, our focus was on evaluating smaller models for deployment. In this study, we want to see after prompt engineering if a relatively small LLM can achieve a competiable performance, compared with relatively huge LLM. Among all candidates, LLaMA-33-70B demonstrated the best balance between these requirements from above. According to Olteanu [14], LLaMA-33-70B consistently outperformed its LLaMA-31-70B version counterparts in translation quality and structured field extraction. In addition, GPT-4o was used as

a benchmark to assess the gap between relatively small and relatively large models under identical prompts.

Based on these findings, we selected LLaMA-33-70B as the core model for all experiments. This choice reflects a realistic and reproducible deployment setting, and supports our hypothesis that carefully designed prompts can enable strong task performance even without any fine-tuning.

### 3.3 Rationale for Choosing Manual and DSPy-Based Prompt Optimization

This study employs a hybrid prompt optimization technique that integrates manual prompt engineering with DSPy-based automated search, both of which are theoretically substantiated and empirically confirmed.

The choice of these two techniques is driven by the explicit input-output structure of our task. It requires extracting `Complaint`, `Cause`, and `Correction` fields from noisy multilingual texts into a fixed and English-translated JSON schema.

Encoding domain-specific patterns, including mechanical abbreviations, mixed-language expressions, and Volvo-specific repair guidelines, required manual prompt design. As shown in recent literature, manual prompt engineering remains crucial in specialized domains where implicit context or companies' internal language cannot be learned solely from general pre-trained models, which demonstrated by Chen in 2025 [52]. Moreover, Wei's team in Google Research department [10] pointed out that techniques like Chain-of-Thought prompting improve multistep reasoning performance, especially when paired with medium-to-large LLMs. While Njifenjou's [53] research also shows that Role-based prompting enhances task alignment by establishing an expert persona. These methods offer transparency, interpretability, and controllability that are hard to replicate through black-box optimization alone.

In contrast, DSPy offers a modular and declarative framework for automatic prompt tuning, which was shown to effectively improve the adequacy of LLMs in Khattab's research [45]. This modular framework is suitable for our input-output structure task. By representing prompts as composable modules and optimizing them using strategies such as `BootstrapFewShotWithRandomSearch` and `ChainOfThought`, DSPy enables efficient exploration of prompt variants across a large search space, without requiring gradient updates or fine-tuning. Study carried out by Sarmah [54] in 2024 that DSPy can outperform handcrafted prompts within minutes of compilation on models like GPT-4o and it is particularly effective when the task output is structured and multilingual, as in our case. Furthermore, DSPy supports reasoning-aware modules that are aligned with our use of CoT and ISO-639-1 code enforcement in earlier prompt iterations. Lastly, according to Khattab [45], DSPy is an open source tool, that allows us to modify its structure in order to achieve our research goals.

The combination of these two techniques, manual design for semantic precision and structural control, and automated optimization for scaling and consistency. The purpose of this project is to create a mutually enhancing process. Theoretically, manual prompts provide a strong inductive bias that guides DSPy's initial search space,

while DSPy helps fine-tune example sets, output formats, and reasoning patterns to maximize model performance.

### 3.4 Prompt Optimization

Since the LLaMA-33-70B model has been fine-tuned for a different domain, it requires adaptation for our specific task.

Instead of fine-tuning, by optimizing prompts, it could improve the performance of LLMs for information extraction and translation tasks. In this study, we use a multi-phase approach to prompt optimization comprising a baseline prompt, hand-based optimization, DSPy-based automatic optimization, and finally a hybrid strategy combining the advantages of human and automated methods.

#### 3.4.1 Baseline Prompt Design

The baseline prompt functions as the foundational pattern for instruction-driven engagement with the LLM. It is engineered to be straightforward, versatile, and proficient in accommodating various technical comment types. The baseline prompt follows an instruction-style format:

```
Extract structured information from the following text and translate it into English.
```

```
Text: "{text}"
```

```
Return the results in JSON format, containing:
```

```
{  
  "language": "...",  
  "Translation": "...",  
  "Complaint": "...",  
  "Cause": "...",  
  "Correction": "..."  
}
```

This baseline guarantees uniformity in LLM output and serves as a foundation for subsequent adjustments. The evaluation was conducted using a zero-shot prompt without examples from the Claim dataset.

#### 3.4.2 Manual Prompt Optimization — Relative Changes from Baseline

The current baseline instructs the model to extract structured information from a given text, translate it into English, and return a strict JSON output with predefined fields. Based on extensive evaluation using metrics ROUGE-L, BERTScore and vector cosine similarity to provide an overall rating. Besides, qualitative error analysis, we introduced several key modifications to address observed shortcomings. Each of iteration of prompt optimization will be evaluated by the metrics, and then we can find out if we will keep the change of prompt or not and what aspects of prompt should be focused on in order to improve the overall score.

### **1. First round optimization - provide an explicit instruction**

Understanding that unclear output can cause discrepancies, the first optimization explicitly tells the model to provide the findings strictly in JSON form. Emphasizing that the structure must fit a preset schema, this version increases format compliance over the baseline. It also includes a new instruction: "Ensure the output is valid JSON," which forces the model to confirm its own response before to output. It also substitutes specific example(e.g., "language": "fr", "Translation": "English translation of the text").

### **2. Second round optimization - handle missing information explicitly**

Building on the previous version, the second round presents reasoning for managing insufficient inputs. In particular, should any of the necessary fields—Complaint, Cause, or Solution) be absent from the original text, the model is now directed to substitute the value "unknown" in place of blank. This change guarantees that the output remains structurally complete and machine-readable by addressing the frequent incomplete or noisy repair records.

### **3. Third round optimization - assign a role to the model**

In this phase, a role-specific directive was implemented by explicitly designating the model as a “structured data extraction assistant.” Although the job itself has not significantly altered from the prior version, the incorporation of this role definition gently delineates the model’s identity and responsibilities, promoting more targeted and consistent conduct.

### **4. Fourth round optimization - introduce dual-role specification for multilingual understanding**

This round defines the model as both a multilingual translator and a structured data extraction assistant, therefore extending its assigned function. Although earlier cues were only on structured extraction, this improvement clearly indicates that the model has to be able to handle multilingual input well. The prompt emphasizes the idea that language variety is fundamental to the work by including a translation-oriented function beside extraction.

### **5. Fifth round optimization - add domain-specific expertise through role refinement**

This round, the model’s job is further honed to incorporate explicit knowledge in the after-sales automobile sector in addition to general experience in structured data extraction and multilingual translating. In car diagnostics, repair terminology, and maintenance procedures, this domain-specific framing helps match the model’s reasoning with actual knowledge. Even in cases when the input is scattered or non-standard, the model is more likely to infer implicit features, understand technical terms precisely, and produce high-quality structured outputs by directly incorporating the industrial context right into the prompt.

**6. Sixth round optimization - incorporate implicit chain-of-thought reasoning through stepwise decomposition**

Based on the fifth round optimization, this version presents a methodical, step-by-step task formulation that suggests indirectly chain-of-thought reasoning. The prompt consists of three ordered steps: identify the language, translate into English, then extract three organized fields rather than outlining the work in a single instruction block. Although the model is not directed to provide intermediate thinking, this explicit deconstruction encourages it to reason internally in a logical sequence.

**7. Seventh round optimization - standardize language detection using ISO-639-1 code**

From the sixth round result, there is a problem related to inaccurate language detection. This round enhances the previous stepwise structure by stipulating that the identified language must be presented in ISO-639-1 code format. This version formalizes the output structure to conform to generally known standards, whereas prior prompts contained language identification. This alteration aims to diminish ambiguity in language representation (e.g., “French” vs “fr”) and improve the accuracy of language detection. The prompt concurrently directs the model through a triadic reasoning process—language identification, translation, and systematic extraction—maintaining the implicit advantages of the previously established CoT.

**8. Eighth round optimization - emphasize full preservation of field-level details**

Based on the result of the seventh round, we found that there is a missing information problem in `Complaint`, `Cause`, and `Correction` field, which are not fully extracted from the raw data. This round enhances the prompt by explicitly instructing the model not to omit any important details when extracting the `Complaint`, `Cause`, and `Correction` fields. The prompt increases the expectation for completeness and granularity in the obtained output by including emphatic sentences such "without omitting any important details" to every field declaration. In the context of after-sales vehicle records, where technical details and references (e.g., component numbers, dates, fault codes) hold diagnostic relevance, this is especially important. Unlike previous iterations that gave structural correctness and reasoning flow top priority, this round brings a quality-based improvement that guarantees semantic integrity and guarantees that all pertinent information from the input text is kept in the final JSON.

**9. Ninth round optimization - introduce one-shot example to guide structure and semantics**

This round adds a whole example of both the input log and the anticipated structured JSON output using a one-shot prompting technique. Unlike previous iterations depending just on instruction-based prompting, the introduction of a realistic input-output pair gives the model a specific pattern to follow. This clarifies in long, technical, noisy repair logs how to treat field boundaries—especially

between Complaint, Cause, and Correction. The one-shot example greatly lowers uncertainty and increases consistency by thus strengthening both semantic interpretation and format compliance. This method improves the generalizing capacity of the model by using in-context learning to augment prior rule-based directions.

**10. Tenth round optimization - introduce few-shot examples to enhance generalization**

This round uses a few-shot prompting technique, giving two whole input-output examples before the primary task instruction. The chosen examples have realistic complexity and full JSON-formatted outputs, and logs in several languages—e.g., English and German. This enables the model to identify in real-world truck repair records structural patterns, field borders, and cross-lingual differences. Grounding the work in actual antecedents helps the few-shot setup enhance the generalizing capacity of the model to new, unseen inputs. It also provides contextual reference points that support format consistency, semantic alignment, and multilingual robustness, so complementing the role specification and stepwise reasoning brought in previous rounds.

**11. Eleventh round optimization - integrate few-shot prompting with dual-language instruction**

This round we expand on the last few-shot technique by localizing the prompt into many dual-language formats: English-Chinese, English-French, English-German, English-Spanish. Task stages and field descriptions, for example, are shown in both English and Chinese, thereby supporting the bilingual capacity of the model.

**12. Twelfth round optimization - multilingual fusion prompt with full instruction translation**

This round constructs a single prompt that embeds multilingual instructions directly into the template. All task directives and field definitions are presented in English, Chinese, French, German, and Spanish. Each field label, such as Complaint, Cause, and Correction, is annotated with its equivalent term in the respective languages (e.g., Complaint ( Réclamation / Beschwerde / Reclamación)). The core structure remains consistent with prior rounds, including step-based task decomposition and the use of ISO-639-1 language code specification. Two complete input-output examples are retained in the prompt to support in-context learning. This configuration allows the model to operate with multilingual inputs and instructions without requiring prompt language switching.

Collectively, over twelve rounds, the manual prompt optimization method consisted in a sequence of organized changes. Each round brought a particular change to the base prompt: output formatting clarity, field completeness requirements, role-based contextualizing, stepwise task decomposition, multilingual instruction embedding.

Changes in language specificity (e.g., ISO-639-1 code enforcement), semantic preservation requirements, and the introduction of few-shot examples helped the prompt to be gradually improved. Dual-language and multilingual teaching strategies helped to solve language accessibility in subsequent rounds. Every change was implemented under stringent formatting restrictions and a consistent output structure. The resulting prompt serves as the foundation for subsequent automatic prompt optimization and performance evaluation.

### 3.4.3 Automatic Prompt Optimization with DSPy

We developed a DSPy-based optimization pipeline centered on a task-specific Signature description in order to enable automatic prompt building and refining. Explicit signature declarations specify the input-output schema of a language model task in DSPy programs; this abstraction lets DSPy execute modular controlled prompt search, example selection, and reasoning compilation.

Designed to jointly execute multilingual translation and structured repair information extraction, this work uses a special signature called `RepairExtraction`. The detailed pseudocode is provided in the Appendix B.1. Table 4 shows the one input field and five output fields defined by the interface. `detail` from the Claim table column 'detail' is the input, while the language, Translation, Complaint, Cause and Correction are the output.

**Table 4:** Field specification of the `RepairExtraction` signature

Field	Type	Description
<code>detail</code>	Input	Multilingual repair log entry in free-text form. May contain abbreviations, technical terms, and mixed-language content.
<code>language</code>	Output	ISO-639-1 language code inferred from the input text (e.g., "de" for German).
<code>Translation</code>	Output	Full English translation of the <code>detail</code> field. Serves as the canonical version for field extraction.
<code>Complaint</code>	Output	The issue or complaint explicitly or implicitly stated by the customer or technician.
<code>Cause</code>	Output	The identified cause or diagnosis that explains the reported problem.
<code>Correction</code>	Output	The corrective action taken, recommended, or inferred from the log entry.

In all DSPy configurations, the prompt search process was controlled using a fixed Teleprompter configuration, which can be seen in table 5.

This configuration equilibrates exploration and calculation, enabling DSPy to assess many prompt alternatives while adhering to practical runtime limitations.

In addition, we developed four distinct DSPy program configurations by altering two control parameters: reasoning mode and prompt search technique. These combinations

**Table 5:** Teleprompter configuration used in DSPy automatic prompt optimization

Parameter	Type	Description
metric	Function	Composite evaluation metric combining semantic similarity, ROUGE-L, and BERTScore. Used to guide prompt selection.
max_bootstrapped_demos	Integer	Maximum number of few-shot examples selected for each candidate program. Set to 5 in all experiments.
num_candidate_programs	Integer	Number of prompt variants DSPy evaluates per round. Set to 10.
num_threads	Integer	Number of parallel threads used to evaluate prompt candidates. Set to 5 for efficient batch execution.

provide a systematic comparison of the effects of reasoning behavior and supervision scale on prompt optimization.

1. **Predict + BootstrapFewShot:** Uses the standard `Predict` module to generate outputs directly, combined with deterministic few-shot prompt construction from 10 annotated examples. This configuration represents a lightweight, fast-converging baseline.
2. **Predict + BootstrapFewShotWithRandomSearch:** Keeps the direct generation mode but replaces deterministic prompt selection with randomized sampling over 100 candidate examples, allowing broader exploration of exemplar composition and prompt phrasing.
3. **ChainOfThought + BootstrapFewShot:** Activates DSPy’s `ChainOfThought` module, which enables multi-step reasoning before generating final answers. Though still limited to 10 few-shot examples, the reasoning steps (e.g., language → translation → extraction) are made explicit during inference.
4. **ChainOfThought + BootstrapFewShotWithRandomSearch:** Combines full CoT reasoning with large-scale randomized few-shot sampling (100 examples). This is the most expressive and resource-intensive configuration, designed to explore the largest prompt and reasoning space.

Each DSPy program was implemented using a shared workflow: the system first chooses few-shot examples (either fixed or sampled), compiles them into candidate prompts, then uses them on the signature-defined task from a validation set of annotated repair logs. DSPy also compiles intermediate reasoning traces into the prompt structure in CoT setups.

To supervise the optimization process, we defined a composite evaluation metric that serves as DSPy’s feedback signal during prompt selection. For each example, we

compute the Cosine similarity, ROUGE-L F1 score and BERTScore F1. Each sub-metric is applied independently to the four key output fields: Translation, Complaint, Cause, and Correction. Their average is taken as a field-level score, and the final instance-level feedback score is computed in Eq.15.

$$s_i = \frac{1}{3} \sum_{j=1}^3 m_j(i) \quad (15)$$

where:

- $m_j(i)$  is the  $j$ -th metric (cosine similarity, ROUGE-L, or BERTScore) calculated for output field  $i$
- $s_i$  is the average score over the three metrics for field  $i$
- FinalScore is the overall composite score across the four output fields

This final score is returned to DSPy as the optimization reward signal, guiding prompt selection, reranking, and example refinement in each experimental configuration.

#### 3.4.4 Hybrid Optimization: Combining Manual and DSPy-Based Methods

We used a hybrid approach whereby the best manually optimized prompt acted as the fixed instruction template to investigate the merging of human-engineered cues with machine optimization. We injected this high-quality prompt as the prefix instead of having DSPy create the prompt structure from start, therefore keeping its structure, roles, field definitions, multilingual instructions, and example form. The detailed pseudocode is provided in the Appendix B.2.

For DSPy configurations, it remains the same in using a fixed Teleprompter configuration in Table 5. The DSPy pipeline was then applied on top of this fixed instruction template to perform exemplar selection and dynamic prompt augmentation. All four previously defined DSPy configurations were re-evaluated under this hybrid setup:

1. **Predict + BootstrapFewShot**: Reuses the manual instruction block and appends 10 fixed examples chosen via deterministic bootstrapping. DSPy compiles them with the manual prompt for direct output generation.
2. **Predict + BootstrapFewShotWithRandomSearch**: Retains the manual template and combines it with 100 examples sampled randomly for prompt construction. This expands exemplar diversity while maintaining structural guidance.
3. **ChainOfThought + BootstrapFewShot**: Uses the fixed manual instructions and appends 10 few-shot examples that contain intermediate reasoning traces. DSPy orchestrates the CoT execution while preserving the human-authored prompt core.

4. **ChainOfThought + BootstrapFewShotWithRandomSearch:** Combines reasoning-augmented few-shot learning with random sampling over 100 candidates, applied on top of the manually constructed prompt. This is the most expressive hybrid configuration in terms of reasoning depth and sampling scale.

The same composite evaluation metric (Eq. 15) was used to evaluate all hybrid configurations. This design is subject to uniform scoring standards and an output schema. This architecture allows for direct comparison of totally manual, fully automatic, and hybrid tactics.

### 3.5 Rationale for Embedding Model Selection

In our system, the selection of the Ada embedding model is primarily influenced by its strong integration with Azure AI Search.

The first reason is due to the advantage of the platform integration. OpenAI's embedding models are tailored for Azure AI Search. The Ada embedding paradigm guarantees a smooth interface with Azure's ecosystem, making use of the platform's native capabilities and streamlining development and deployment.

The other reason is the effective performance and response speed of the Ada embedding model. It has demonstrated robust performance and low latency in Azure's environment. This high efficiency in processing extensive real-time queries is essential for providing an accurate and responsive semantic search experience.

### 3.6 Rationale for Metric Selection

Lexical overlap at the surface level, contextual semantic coherence, and general semantic similarity between the generated text and the reference text are important in automatic evaluation. Based on these comprehensive requirements, we selected ROUGE-L, BERTScore, and Cosine Similarity (based on Azure OpenAI embeddings) as the primary evaluation metrics. The following justifies the reason not using the alternative measurements mentioned in Chapter 2.

- **ROUGE-L:** According to Lin [48], in contrast to ROUGE-N, ROUGE-L identifies the longest common subsequence (LCS) between texts, thereby more accurately representing sequence preservation and structural fluency alongside word overlap. Another study carried out by Schaik [55] who is Microsoft machine learning expert pointed out, ROUGE-L, as a recall-focused metric, is especially beneficial for activities necessitating comprehensive coverage of essential information, including summarization and question answering. It determines the LCS between the candidate and reference texts without necessitating a predetermined n-gram length, enabling the identification of long-distance yet order-preserving correspondences. In contrast to ROUGE-N, which solely counts fixed-length n-grams, ROUGE-L's adaptability in managing word order enhances its ability to capture sentence structure and content coherence. Therefore, ROUGE-L satisfies our criteria for surface-level matching and is easier to compute and more efficient than ROUGE-W and ROUGE-N.

- **BERTScore:** BERTScore evaluates semantic similarity beyond precise lexical matching using contextual embeddings produced by pre-trained language models. Additionally, BERTScore performs exceptionally well in multilingual machine translation evaluation. According to Zhang’s team [50], they tested BERTScore on the WMT benchmarks across various language pairs, including English–German, English–French, and English–Chinese, and found that it correlates significantly better with human judgments compared to traditional metrics like BLEU. For example, in their study, BERTScore can more accurately capture semantic equivalence, and it does not penalize translations that differ in surface form but are semantically correct, unlike BLEU. Another study from Yousuf [56], it indicates BERTScore that shows much higher correlation with human judgment than BLEU across various language pairs, such as English-Hausa and English-Chinese. This demonstrates that BERTScore is effective at recognizing paraphrases and semantic translations in language pairs. Therefore, it provides a more accurate reflection of translation quality. This makes it better in its evaluation performance in handling the paraphrasing, translating, and generating texts tasks, better matching the evaluation with human judgment.
- **Cosine Similarity:** Cosine Similarity quantifies semantic proximity by assessing the angle between vectors in the embedding space. For example, According to Karadeniz [57] on the BioNLP 2016 Bacteria Biotope normalization task, this method achieved a precision of 65.9%, outperforming the best system at the time by 2.9 percentage points. Compared to another approach based on Word Mover’s Distance (WMD), the model using cosine similarity demonstrated a significantly higher precision on the development set (62.9% vs. 49.0%). This confirms that semantic embeddings combined with cosine similarity can more effectively capture the semantic association between entity names and standardized concepts. In addition, Hammer [58] demonstrated that cosine similarity is suitable for use in cross-lingual evaluation tasks. Cross-lingual alignment refers to mapping semantically equivalent or similar texts (words or sentences) from different languages to vector representations that are close to each other. If alignment is effective, sentences with similar meanings across languages should exhibit high cosine similarity in their representations. Lastly, employing Azure OpenAI embeddings guarantees superior vector representations, rendering the metric efficient for assessing our multilingual semantic alignment.

Overall semantic proximity (Cosine Similarity), deep semantic alignment (BERTScore), and surface-level lexical similarity (ROUGE-L) are the three main evaluation aspects where the selected metrics complement each other. Due to their individual shortcomings in computing efficiency or robustness, alternative metrics such as ROUGE-N and ROUGE-W were not included.

### 3.7 Evaluation Framework

We use an automatic measure framework to assess the success of several prompt optimization techniques.

Using three complementing metrics at the automated evaluation—ROUGE-L, BERTScore, and Cosine Similarity based on Azure OpenAI embeddings—we evaluate the similarity between model-generated output and the annotated ground truth. Chapter 2 contains the mathematical definitions of these measures.

Each metric is applied to the four key output fields: Complaint, Cause, Correction, and Translation. The average of the three metric scores produces a field-level composite score. The final evaluation score for each sample is obtained by averaging across all four fields, as defined in Equation 15. This scoring logic is consistently applied to outputs from all prompt optimization strategies, including manual, DSPy-based, and hybrid methods.

**Table 6:** Core components of the automatic evaluation process

<b>Evaluation Step</b>	<b>Description</b>
Metric Calculation	Compute BERTScore, ROUGE-L, and semantic similarity for each field (Complaint, Cause, Correction, Translation).
Composite Scoring	Aggregate field-level metrics into a unified composite score using the formulation in Equation 15.
Result Interpretation	Compare composite scores across all prompt types (manual, DSPy, hybrid) for consistent performance benchmarking.

The methodology provides scalable automated scoring metrics to guarantee comprehensive and in-depth assessment of the quality of multilingual information extraction.

### 3.8 Test Framework Construction

The test framework is intended to assess the LLaMA-33-70B model under controlled conditions. By use of the Azure OpenAI API, the model is implemented in a cloud-based environment, therefore facilitating effective handling of vast textual data. The evaluation pipeline includes pre-processing maintenance records, creating predictions using optimal prompts, comparing outputs with the Ground Truth Table, and computing evaluation metrics. Following a methodical review approach guarantees consistent and repeatable outcomes by use of the framework.

We built a Python-based batch processing framework to enable modular and large-scale experiments. Each .txt file in the pipeline corresponds to one version of a handcrafted prompt; the pipeline reads all manually designed prompt files kept in a designated prompt/ directory. The script iteratively delivers each prompt to the LLaMA-33-70B model together with the input maintenance data; it then parses and stores the prediction results in a structured way. Every output saves itself as a .parquet file within the extraction\_and\_translation\_result/ folder. The related

prompt version or optimization technique determines the automatic assignment of the filenames.

The same procedure is applied to prompts optimized using the DSPy and hybrid methods. That is, for each DSPy-based or hybrid-enhanced prompt, the script performs prediction over the same input set and stores outputs as .parquet files in the same results directory. This unified file structure allows for seamless integration and comparison of different strategies.

To evaluate the quality of each set of predictions, we implemented an automated scoring script that loads every .parquet file in the `extraction_and_translation_result/` folder, applies the composite evaluation metric introduced in Section 15, and computes individual field-level and overall scores. The evaluation outputs are stored in the `evaluation_result/` directory as .txt files. Each result file is named to reflect the corresponding prompt source (manual, DSPy, hybrid), ensuring traceability and version control.

Batch running, performance benchmarking, and extensive repeatability across all prompt design techniques are made possible by this automated testing system.

## 4 Results and Analysis

The empirical evaluation of several prompt optimization techniques for multilingual repair log interpretation is given in this chapter. The study emphasizes quantitative assessment applied with automatic metrics. We evaluate specifically manually created prompts, DSPy-generated prompts, hybrid prompts on four fundamental output fields: Complaint, Cause, Correction, and Translation. In addition, the effects of LLaMA-33-70B and GPT-4o were compared with the same prompts and the results are shown.

### 4.1 Environment and Tools

All experiments were conducted using Python version 3.12.3. The core library for prompt engineering and optimization was DSPy version 2.6.3. The computational tasks were executed within the Volvo Data Science Platform environment, utilizing a Jupyter Notebook interface. The allocated hardware resources consisted of 4 CPU cores and 64 GB of RAM. No Graphics Processing Unit (GPU) acceleration was employed for these experiments.

### 4.2 Results

All reported scores in the following evaluation tables are the **average results across ten distinct test sets**, designed to evaluate model generalizability under different data conditions. Each test set contains multilingual repair records with varying levels of language complexity, domain-specific vocabulary, and annotation consistency. This multi-set evaluation ensures that our findings are robust and not overly tailored to a single test distribution.

To systematically compare different prompt engineering strategies, we report four evaluation metrics: ROUGE-L, BERTScore, Semantic Similarity, and Composite Score on each output field. The results are computed across 1 baseline, 12 manually optimized prompts, 4 DSPy-optimized prompts, and 4 hybrid-enhanced prompts. Each metric captures complementary aspects of prediction quality, allowing for a comprehensive performance comparison.

#### **4.2.1 Complaint Field Results(LLaMA-33-70B)**

The Complaint field evaluates the model’s ability to correctly extract the issue as described by the customer or technician. Table 7 and Figure 9 illustrate the performance of all prompt strategies on this field. The results show a clear upward trend across manual prompts, with the later prompts incorporating better instructions and examples. DSPy-generated prompts further improve extraction quality, especially when using Chain-of-Thought reasoning. The hybrid prompts yield the highest scores, with Hybrid-4 achieving the best overall performance across all metrics, indicating that combining manual engineering with automated optimization can significantly enhance Complaint-level understanding.

#### **4.2.2 Cause Field Results(LLaMA-33-70B)**

The Cause field evaluates the model’s ability to identify the underlying reason for each reported Complaint. The results follow a similar pattern to the Complaint field (see Table 8 and Figure 10), with improvements across manual iterations and further gains from DSPy strategies. Hybrid-4 remains the top-performing configuration.

#### **4.2.3 Correction Field Results(LLaMA-33-70B)**

The Correction field assesses the model’s ability to extract the applied solution from the repair logs. As with previous fields (Table 9 and Figure 11), performance increases progressively through manual, DSPy, and hybrid strategies. Again, Hybrid-4 achieves the best overall scores across all metrics.

#### **4.2.4 Translation Field Results(LLaMA-33-70B)**

The Translation field measures the model’s translation accuracy from the original language to English. The results in Table 10 and 12 show trends consistent with the previous fields: manual prompts improve over iterations, DSPy adds further gains, and hybrid prompts deliver the highest quality outputs. Hybrid-4 continues to outperform all other methods.

#### **4.2.5 Overall Performance Summary(LLaMA-33-70B)**

Table 11 and Figure 13 summarize the average performance across the four structured fields for each prompt type. We observe that best hybrid prompts(Hybrid-4) consistently outperform both the best manual(M9 in Translation field and M11 in other three fields)

and the best DSPy-only prompts(D4), highlighting the effectiveness of combining expert-designed structure with data-driven optimization.

#### **4.2.6 Cross-Model Comparison Results: LLaMA-33-70B vs. GPT-4o**

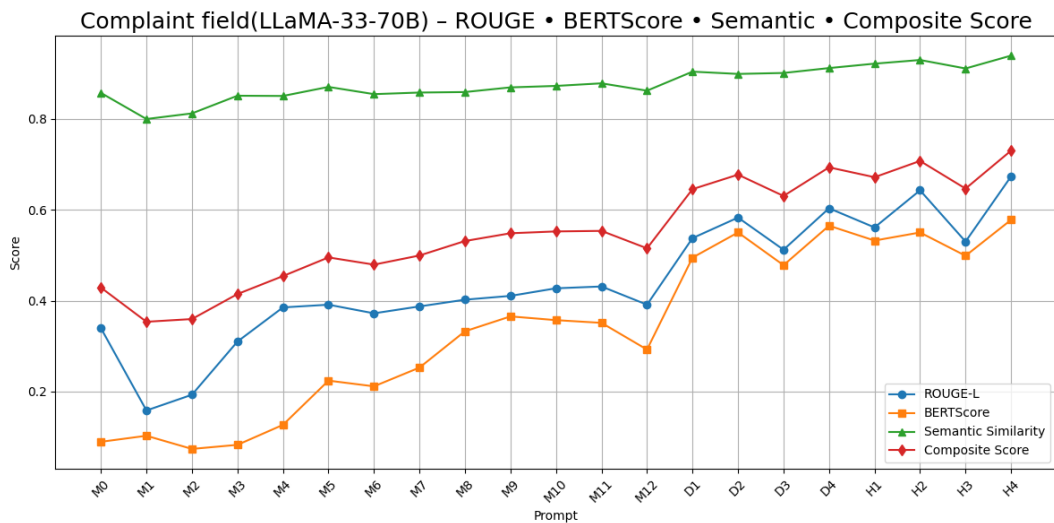
To investigate whether a relatively small LLM can perform as well as a relatively large LLM after our prompt engineering strategy, we compared the performance of LLaMA-33-70B, a relatively lightweight model, with GPT-4o, a model with significantly larger parameter count and training resources.

We evaluated both models under five identical prompt configurations: Baseline (M0), Manual-9 (M9), Manual-11 (M11), DSPy-4 (D4), and Hybrid-4 (H4). The results can be seen in Table 12, Figure 14 and Figure 15.

**Table 7:** Evaluation Results for Complaint Field (LLaMA-33-70B). Composite Scores are reported as mean  $\pm$  SD over 10 test sets. Statistical significance is based on paired two-tailed t-tests versus Baseline.

Prompt	ROUGE-L	BERTScore	Semantic Sim.	Composite Score
Baseline (M0)	0.3394	0.0892	0.8574	0.4287 $\pm$ 0.0195
Manual-1 (M1)	0.1582	0.1025	0.7999	0.3535 $\pm$ 0.0121**
Manual-2 (M2)	0.1928	0.0735	0.8123	0.3595 $\pm$ 0.0069**
Manual-3 (M3)	0.3099	0.0825	0.8512	0.4145 $\pm$ 0.0096*
Manual-4 (M4)	0.3851	0.1265	0.8507	0.4541 $\pm$ 0.0102**
Manual-5 (M5)	0.3911	0.2239	0.8706	0.4952 $\pm$ 0.0097**
Manual-6 (M6)	0.3721	0.2113	0.8547	0.4793 $\pm$ 0.0112**
Manual-7 (M7)	0.3873	0.2526	0.8582	0.4994 $\pm$ 0.0056**
Manual-8 (M8)	0.4021	0.3326	0.8593	0.5313 $\pm$ 0.0080**
Manual-9 (M9)	0.4105	<b>0.3653</b>	0.8696	0.5484 $\pm$ 0.0108**
Manual-10 (M10)	0.4272	0.3571	0.8728	0.5524 $\pm$ 0.0065**
<b>Manual-11 (M11)</b>	<b>0.4312</b>	0.3511	<b>0.8785</b>	<b>0.5536 <math>\pm</math> 0.0085**</b>
Manual-12 (M12)	0.3911	0.2923	0.8623	0.5152 $\pm$ 0.0052**
DSPy-1(D1)	0.5377	0.4948	0.9042	0.6455 $\pm$ 0.0164**
DSPy-2(D2)	0.5829	0.5501	0.8992	0.6774 $\pm$ 0.0095**
DSPy-3(D3)	0.5121	0.4781	0.9013	0.6305 $\pm$ 0.0088**
<b>DSPy-4(D4)</b>	<b>0.6034</b>	<b>0.5650</b>	<b>0.9121</b>	<b>0.6935 <math>\pm</math> 0.0065**</b>
Hybrid-1(H1): M11+D1	0.5612	0.5323	0.9218	0.6718 $\pm$ 0.0091**
Hybrid-2(H2): M11+D2	0.6429	0.5501	0.9299	0.7076 $\pm$ 0.0107**
Hybrid-3(H3): M11+D3	0.5301	0.4988	0.9112	0.6467 $\pm$ 0.0251**
<b>Hybrid-4(H4): M11+D4</b>	<b>0.6735</b>	<b>0.5779</b>	<b>0.9395</b>	<b>0.7303 <math>\pm</math> 0.0106**</b>

**Note:** \* $p < 0.05$ , \*\* $p < 0.01$ , based on paired  $t$ -test versus Baseline (M0).

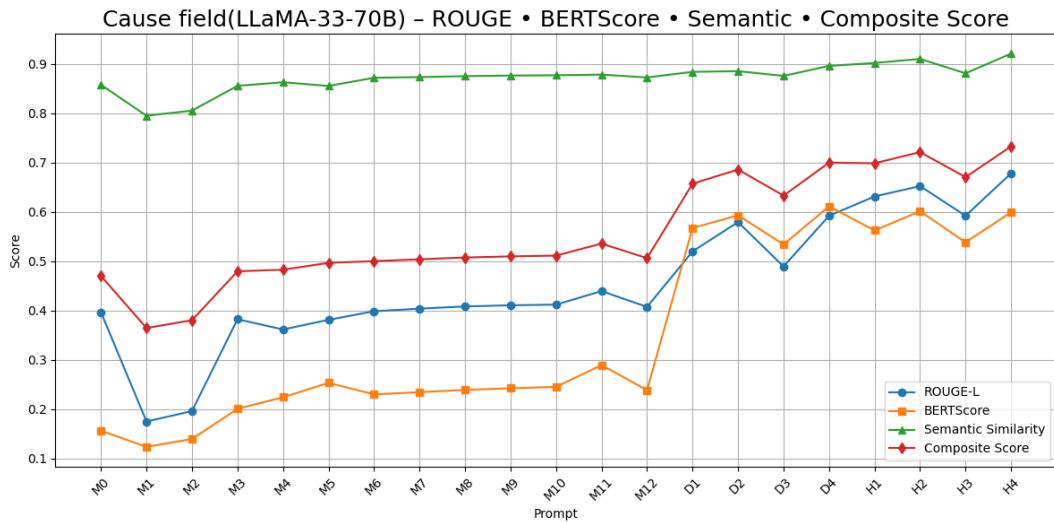


**Figure 9:** Complaint Field - All Metrics(LLaMA-33-70B).

**Table 8:** Evaluation Results for Cause Field (LLaMA-33-70B). Composite Scores are reported as mean  $\pm$  SD over 10 test sets. Statistical significance is based on paired two-tailed t-tests versus Baseline.

Prompt	ROUGE-L	BERTScore	Semantic Sim.	Composite Score
Baseline (M0)	0.3963	0.1560	0.8579	0.4700 $\pm$ 0.0154
Manual-1 (M1)	0.1747	0.1231	0.7950	0.3643 $\pm$ 0.0098**
Manual-2 (M2)	0.1958	0.1392	0.8055	0.3802 $\pm$ 0.0111**
Manual-3 (M3)	0.3822	0.2004	0.8557	0.4794 $\pm$ 0.0114
Manual-4 (M4)	0.3614	0.2239	0.8630	0.4827 $\pm$ 0.0078*
Manual-5 (M5)	0.3810	0.2531	0.8555	0.4965 $\pm$ 0.0053**
Manual-6 (M6)	0.3984	0.2297	0.8721	0.5001 $\pm$ 0.0061**
Manual-7 (M7)	0.4037	0.2342	0.8735	0.5038 $\pm$ 0.0052**
Manual-8 (M8)	0.4081	0.2388	0.8754	0.5074 $\pm$ 0.0048**
Manual-9 (M9)	0.4105	0.2421	0.8766	0.5097 $\pm$ 0.0045**
Manual-10 (M10)	0.4118	0.2450	0.8772	0.5113 $\pm$ 0.0042**
<b>Manual-11 (M11)</b>	<b>0.4393</b>	<b>0.2891</b>	<b>0.8785</b>	<b>0.5356 <math>\pm</math> 0.0041**</b>
Manual-12 (M12)	0.4072	0.2383	0.8728	0.5061 $\pm$ 0.0046**
<hr/>				
DSPy-1(D1)	0.5198	0.5673	0.8841	0.6571 $\pm$ 0.0043**
DSPy-2(D2)	0.5793	0.5930	0.8855	0.6859 $\pm$ 0.0040**
DSPy-3(D3)	0.4892	0.5339	0.8759	0.6330 $\pm$ 0.0049**
<b>DSPy-4(D4)</b>	<b>0.5921</b>	<b>0.6113</b>	<b>0.8962</b>	<b>0.6999 <math>\pm</math> 0.0037**</b>
<hr/>				
Hybrid-1(H1): M11+D1	0.6313	0.5623	0.9021	0.6986 $\pm$ 0.0042**
Hybrid-2(H2): M11+D2	0.6523	<b>0.6012</b>	0.9103	0.7213 $\pm$ 0.0044**
Hybrid-3(H3): M11+D3	0.5921	0.5382	0.8812	0.6705 $\pm$ 0.0051**
<b>Hybrid-4(H4): M11+D4</b>	<b>0.6782</b>	0.5993	<b>0.9211</b>	<b>0.7329 <math>\pm</math> 0.0040**</b>

Note: \* $p < 0.05$ , \*\* $p < 0.01$ , based on paired  $t$ -test versus Baseline (M0).

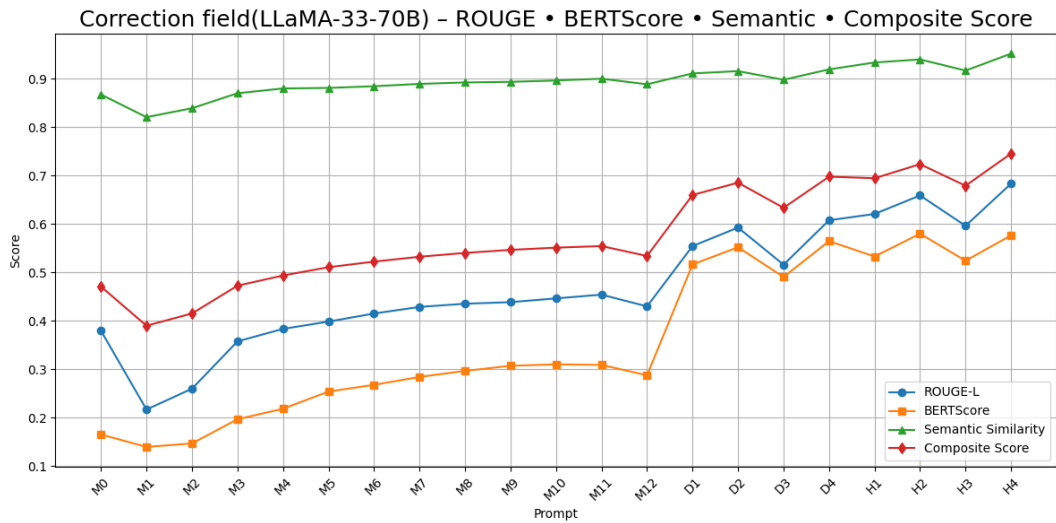


**Figure 10:** Cause Field - All Metrics(LLaMA-33-70B).

**Table 9:** Evaluation Results for Correction Field (LLaMA-33-70B). Composite Scores are reported as mean  $\pm$  SD over 10 test sets. Statistical significance is based on paired two-tailed t-tests versus Baseline.

Prompt	ROUGE-L	BERTScore	Semantic Sim.	Composite Score
Baseline (M0)	0.3794	0.1653	0.8671	0.4706 $\pm$ 0.0111
Manual-1 (M1)	0.2168	0.1396	0.8203	0.3898 $\pm$ 0.0081**
Manual-2 (M2)	0.2598	0.1468	0.8388	0.4151 $\pm$ 0.0086**
Manual-3 (M3)	0.3575	0.1968	0.8697	0.4725 $\pm$ 0.0064*
Manual-4 (M4)	0.3833	0.2184	0.8796	0.4935 $\pm$ 0.0060**
Manual-5 (M5)	0.3983	0.2539	0.8807	0.5106 $\pm$ 0.0051**
Manual-6 (M6)	0.4150	0.2678	0.8842	0.5221 $\pm$ 0.0046**
Manual-7 (M7)	0.4286	0.2841	0.8890	0.5322 $\pm$ 0.0043**
Manual-8 (M8)	0.4352	0.2966	0.8920	0.5401 $\pm$ 0.0040**
Manual-9 (M9)	0.4385	0.3071	0.8933	0.5464 $\pm$ 0.0039**
Manual-10 (M10)	0.4462	<b>0.3101</b>	0.8960	0.5510 $\pm$ 0.0037**
<b>Manual-11 (M11)</b>	<b>0.4540</b>	0.3089	<b>0.8997</b>	<b>0.5542 <math>\pm</math> 0.0065**</b>
Manual-12 (M12)	0.4297	0.2875	0.8882	0.5338 $\pm$ 0.0042**
<hr/>				
DSPy-1(D1)	0.5542	0.5163	0.9106	0.6597 $\pm$ 0.0066**
DSPy-2(D2)	0.5924	0.5518	0.9154	0.6851 $\pm$ 0.0052**
DSPy-3(D3)	0.5158	0.4904	0.8975	0.6332 $\pm$ 0.0059**
<b>DSPy-4(D4)</b>	<b>0.6075</b>	<b>0.5644</b>	<b>0.9190</b>	<b>0.6975 <math>\pm</math> 0.0049**</b>
<hr/>				
Hybrid-1(H1): M11+D1	0.6205	0.5327	0.9332	0.6943 $\pm$ 0.0081**
Hybrid-2(H2): M11+D2	0.6587	<b>0.5798</b>	0.9395	0.7233 $\pm$ 0.0113**
Hybrid-3(H3): M11+D3	0.5960	0.5239	0.9165	0.6788 $\pm$ 0.0080**
<b>Hybrid-4(H4): M11+D4</b>	<b>0.6832</b>	0.5764	<b>0.9515</b>	<b>0.7452 <math>\pm</math> 0.0073**</b>

Note: \* $p < 0.05$ , \*\* $p < 0.01$ , based on paired  $t$ -test versus Baseline (M0).

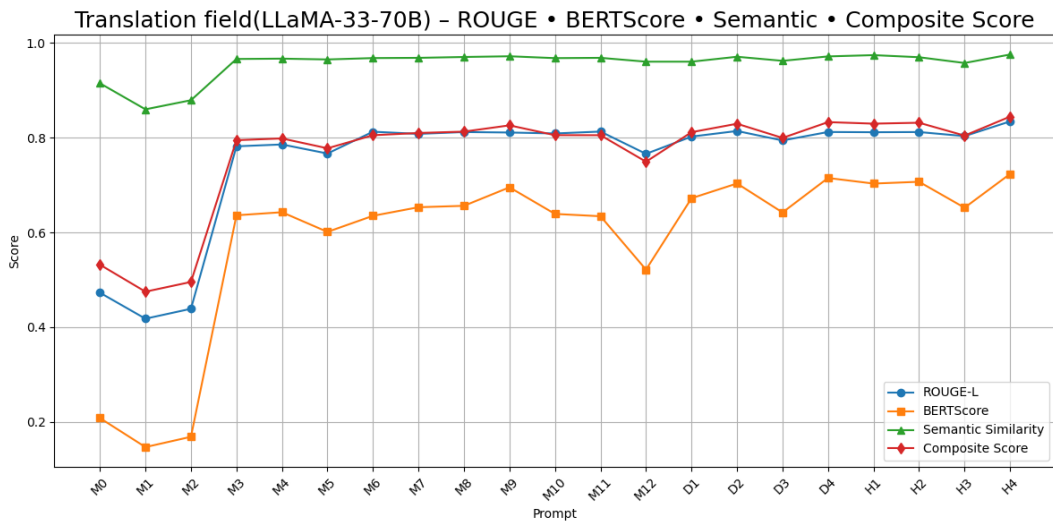


**Figure 11:** Correction Field - All Metrics(LLaMA-33-70B).

**Table 10:** Evaluation Results for Translation Field (LLaMA-33-70B). Composite Scores are reported as mean  $\pm$  SD over 10 test sets. Statistical significance is based on paired two-tailed t-tests versus Baseline.

Prompt	ROUGE-L	BERTScore	Semantic Sim.	Composite Score
Baseline (M0)	0.4728	0.2084	0.9145	0.5319 $\pm$ 0.0069
Manual-1 (M1)	0.4180	0.1472	0.8596	0.4750 $\pm$ 0.0111**
Manual-2 (M2)	0.4389	0.1689	0.8788	0.4955 $\pm$ 0.0118**
Manual-3 (M3)	0.7816	0.6361	0.9658	0.7945 $\pm$ 0.0041**
Manual-4 (M4)	0.7855	0.6426	0.9665	0.7982 $\pm$ 0.0045**
Manual-5 (M5)	0.7664	0.6011	0.9648	0.7774 $\pm$ 0.0048**
Manual-6 (M6)	0.8124	0.6350	0.9677	0.8050 $\pm$ 0.0047**
Manual-7 (M7)	0.8078	0.6531	0.9682	0.8097 $\pm$ 0.0042**
Manual-8 (M8)	<b>0.8117</b>	0.6561	0.9700	0.8126 $\pm$ 0.0044**
<b>Manual-9 (M9)</b>	0.8107	<b>0.6951</b>	<b>0.9715</b>	<b>0.8258 <math>\pm</math> 0.0052**</b>
Manual-10 (M10)	0.8087	0.6390	0.9676	0.8051 $\pm$ 0.0097**
Manual-11 (M11)	0.8127	0.6341	0.9682	0.8050 $\pm$ 0.0068**
Manual-12 (M12)	0.7657	0.5215	0.9601	0.7491 $\pm$ 0.0075**
DSPy-1(D1)	0.8021	0.6721	0.9601	0.8114 $\pm$ 0.0056**
DSPy-2(D2)	<b>0.8139</b>	0.7032	0.9705	0.8292 $\pm$ 0.0075**
DSPy-3(D3)	0.7939	0.6421	0.9619	0.7993 $\pm$ 0.0083**
<b>DSPy-4(D4)</b>	0.8118	<b>0.7148</b>	<b>0.9712</b>	<b>0.8326 <math>\pm</math> 0.0070**</b>
Hybrid-1(H1): M9+D1	0.8113	0.7030	0.9739	0.8294 $\pm$ 0.0081**
Hybrid-2(H2): M9+D2	0.8117	0.7068	0.9695	0.8313 $\pm$ 0.0096**
Hybrid-3(H3): M9+D3	0.8031	0.6521	0.9573	0.8042 $\pm$ 0.0112**
<b>Hybrid-4(H4): M9+D4</b>	<b>0.8341</b>	<b>0.7233</b>	<b>0.9752</b>	<b>0.8442 <math>\pm</math> 0.0082**</b>

Note: \* $p < 0.05$ , \*\* $p < 0.01$ , based on paired  $t$ -test versus Baseline (M0).

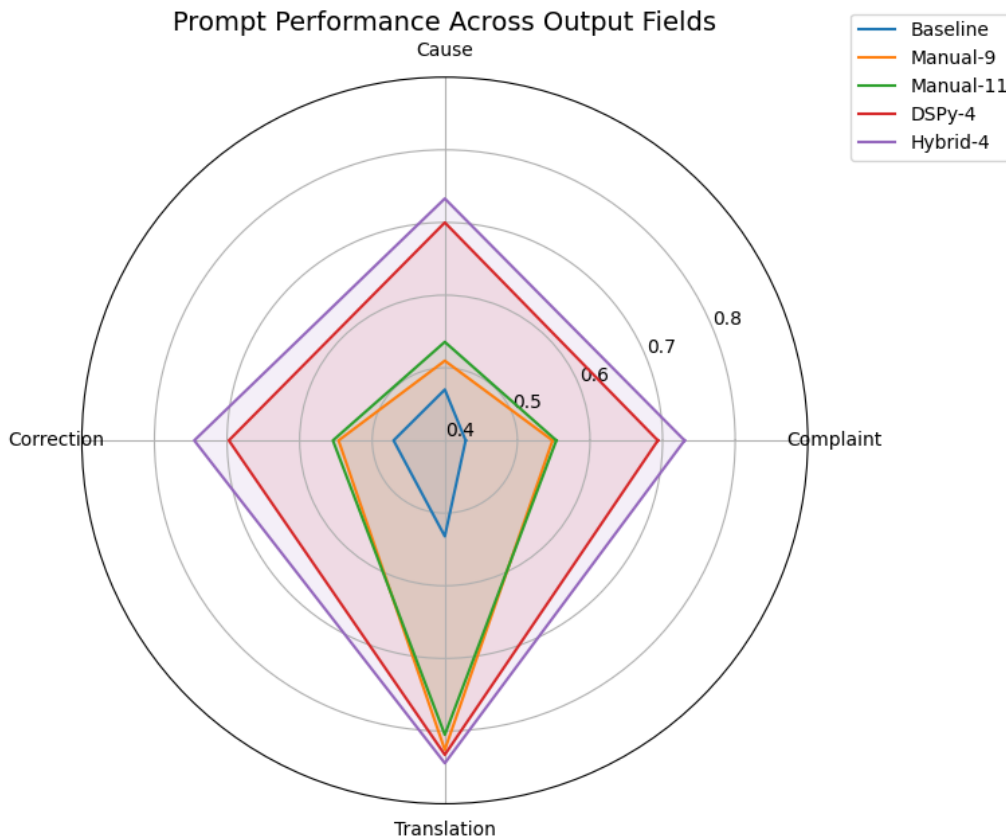


**Figure 12:** Translation Field - All Metrics(LLaMA-33-70B).

**Table 11:** Average Composite Scores Across All Fields (LLaMA-33-70B). Scores are reported as mean  $\pm$  SD over 10 test sets.

Prompt Type	Complaint	Cause	Correction	Translation
Baseline	0.4287 $\pm$ 0.0195	0.4700 $\pm$ 0.0154	0.4706 $\pm$ 0.0111	0.5319 $\pm$ 0.0069
Manual-9	0.5484 $\pm$ 0.0108	0.5097 $\pm$ 0.0045	0.5464 $\pm$ 0.0039	0.8258 $\pm$ 0.0052
Manual-11	0.5536 $\pm$ 0.0085	0.5356 $\pm$ 0.0041	0.5542 $\pm$ 0.0065	0.8050 $\pm$ 0.0068
DSPy-4	0.6935 $\pm$ 0.0065	0.6999 $\pm$ 0.0037	0.6975 $\pm$ 0.0049	0.8326 $\pm$ 0.0070
Hybrid-4	<b>0.7303 <math>\pm</math> 0.0106</b>	<b>0.7329 <math>\pm</math> 0.0040</b>	<b>0.7452 <math>\pm</math> 0.0073</b>	<b>0.8442 <math>\pm</math> 0.0082</b>

**Note:** Along the path from Baseline  $\rightarrow$  Manual-9  $\rightarrow$  Manual-11  $\rightarrow$  DSPy-4  $\rightarrow$  Hybrid-4, all pairwise differences are statistically significant ( $p < 0.05$ ), although some changes represent performance drops rather than improvements.



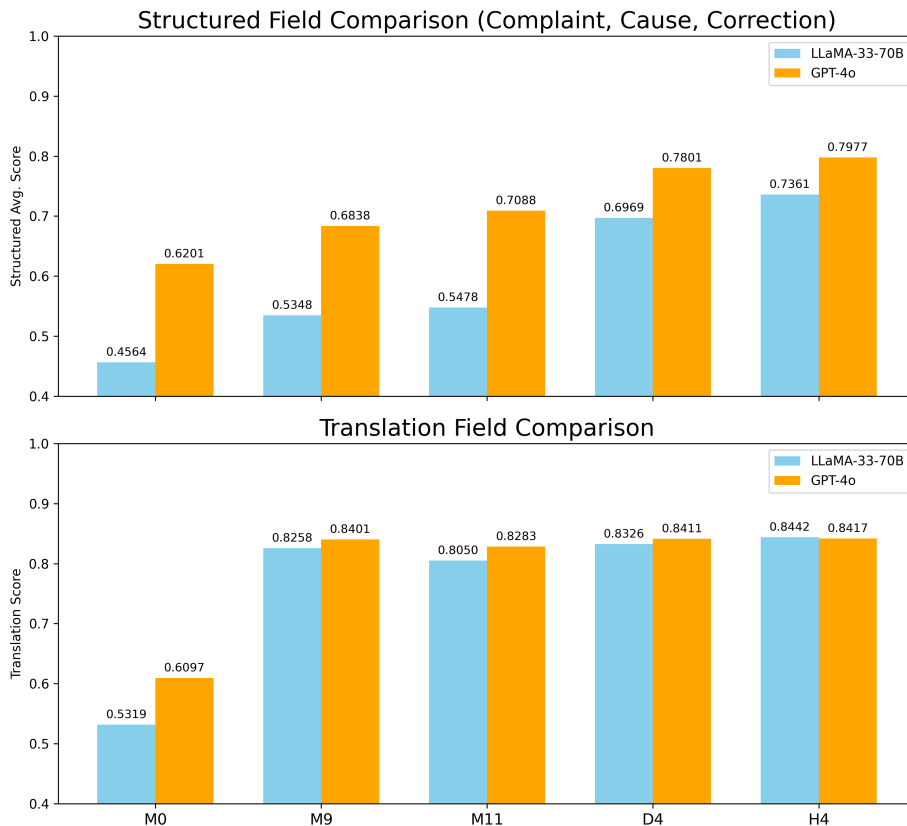
**Figure 13:** Composite scores of the best prompts across different optimization strategies (LLaMA-33-70B).

**Table 12:** Comparison of Composite Scores between LLaMA-33-70B (LLaMA) and GPT-4o across Key Prompt Configurations. Scores are reported as mean  $\pm$  SD over 10 test sets.

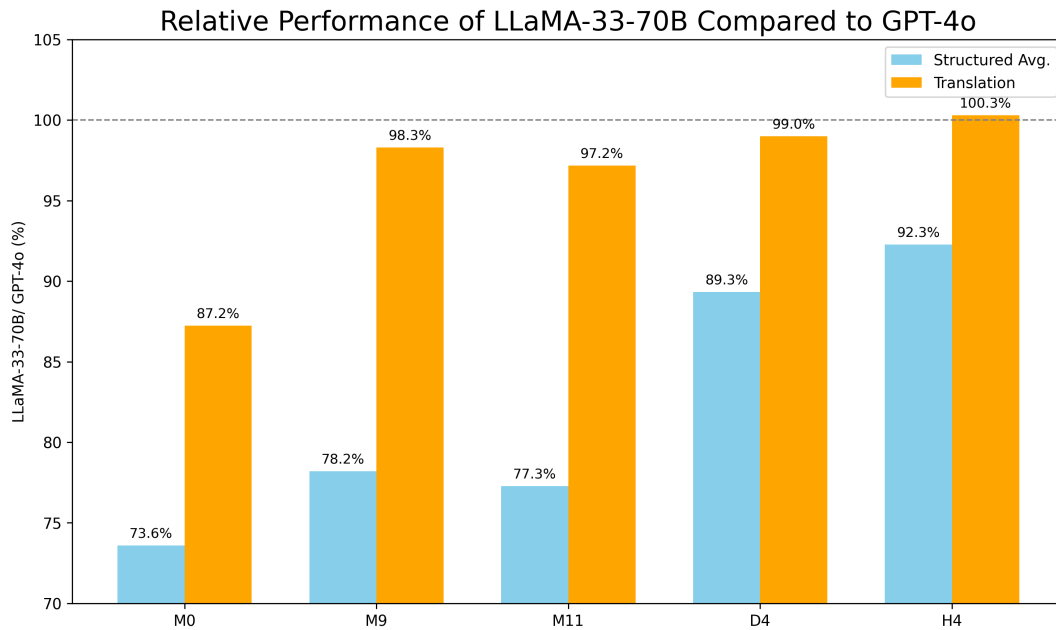
Model	Prompt	Complaint	Cause	Correction	Translation
LLaMA	Baseline (M0)	0.4287 $\pm$ 0.0195	0.4700 $\pm$ 0.0154	0.4706 $\pm$ 0.0111	0.5319 $\pm$ 0.0069
GPT-4o	Baseline (M0)	0.6011 $\pm$ 0.0093	0.6238 $\pm$ 0.0178	0.6353 $\pm$ 0.0141	0.6097 $\pm$ 0.0110
LLaMA	Manual-9 (M9)	0.5484 $\pm$ 0.0108	0.5097 $\pm$ 0.0085	0.5464 $\pm$ 0.0093	0.8258 $\pm$ 0.0052
GPT-4o	Manual-9 (M9)	0.6659 $\pm$ 0.0126	0.7021 $\pm$ 0.0103	0.6835 $\pm$ 0.0189	0.8401 $\pm$ 0.0148
LLaMA	Manual-11 (M11)	0.5536 $\pm$ 0.0085	0.5356 $\pm$ 0.0041	0.5542 $\pm$ 0.0065	0.8050 $\pm$ 0.0068
GPT-4o	Manual-11 (M11)	0.7139 $\pm$ 0.0077	0.6929 $\pm$ 0.0146	0.7197 $\pm$ 0.0123	0.8283 $\pm$ 0.0101
LLaMA	DSPy-4 (D4)	0.6935 $\pm$ 0.0065	0.6999 $\pm$ 0.0037	0.6975 $\pm$ 0.0049	0.8326 $\pm$ 0.0070
GPT-4o	DSPy-4 (D4)	0.7639 $\pm$ 0.0174	0.7769 $\pm$ 0.0115	0.7996 $\pm$ 0.0152	0.8411 $\pm$ 0.0087
LLaMA	Hybrid-4 (H4)	0.7303 $\pm$ 0.0106	0.7329 $\pm$ 0.0040	0.7452 $\pm$ 0.0073	<b>0.8442 <math>\pm</math> 0.0082</b>
GPT-4o	Hybrid-4 (H4)	<b>0.7875 <math>\pm</math> 0.0145</b>	<b>0.7949 <math>\pm</math> 0.0097</b>	<b>0.8106 <math>\pm</math> 0.0183</b>	0.8417 $\pm$ 0.0054

**Note:** Scores are averaged over 10 test sets. According to paired two-tailed *t*-tests between LLaMA-33-70B and GPT-4o, all comparisons are statistically significant at  $p < 0.01$ , **except for the Translation field under DSPy-4 and Hybrid-4**, where  $p < 0.05$ .

LLaMA-33-70B vs GPT-4o – Prompt Effectiveness by Field



**Figure 14:** LLaMA-33-70B vs GPT-4o – Prompt Effectiveness by Field.



**Figure 15:** LLaMA-33-70B vs GPT-4o – relative improvement.

### 4.3 Analysis

Using four evaluation metrics—ROUGE-L, BERTScore, Semantic Similarity, and Composite Score—this part offers a thorough study of the results across the four output fields: Complaint, Cause, Correction, and Translation. To ensure uniformity and robustness, each result is averaged over three test sets. We focus on assessing the effects of manual prompt engineering, DSPy automatic optimization, and hybrid solutions including both methods.

#### 4.3.1 Manual Prompt Engineering(LLaMA-33-70B)

With each iteration addressing a specific limitation noted in previous outputs, the twelve rounds of manual prompt refinement offer a clear trend of improvement. Starting with structural clarity and format enforcement (M1), the procedure moved through management of missing fields (M2), role assignment (M3), and multilingual awareness (M4–M5). Incremental job decomposition (M6), ISO-639-1 language standard (M7), and field completeness (M8) all produced significant gains. Using one-shot and few-shot examples, the next rounds (M9–M12) included in-context learning and produced multilingual fusion prompts.

With top Composite Scores in three of the four fields— Complaint (0.5536), Cause (0.5356), and Correction (0.5542)—Manual-11 (M11) is the greatest overall performance among all manual prompts. These three fields’ manual optimization baseline is Complaint (0.4287), Cause (0.4700), and Correction (0.4706), which are improved by around 29.1%, 14.0% and 17.8%. These findings reveal that methodically layering instructions—especially involving role-based framing, reasoning decomposition, and

language disambiguation—especially can greatly increase structured output quality. M11 also shows more stability and generalization over more input kinds than M12. M12 combines comprehensive multilingual training by embedding five languages (English, Chinese, French, German, Spanish) into the prompt; this ambitious fusion brings major prompt complexity and possible cognitive overload for the model. The long multilingual instructions could unintentionally weaken attention, raise token sparsity, and compromise the clarity of structural direction instead of simplifying the work.

However, Manual-9 (M9), which introduced a one-shot example, achieves the highest score in the Translation field with score 0.8258. By contrast, the second and third highest scores in manual prompt optimization process are reached by M10 and M11, which are 0.8050 and 0.8051. M9 outperforms all other manual prompts, this indicates that example-driven prompting is a little effective for semantic alignment in multilingual translation tasks. Nevertheless, M9 lags behind M11 in all other structured fields, revealing that while in-context examples boost semantic precision, they may not provide the same robustness for reasoning and field-level parsing across noisy technical records.

#### **4.3.2 DSPy Optimization(LLaMA-33-70B)**

DSPy-generated prompts consistently outperform manually engineered ones in most fields. The best-performing DSPy strategy, DSPy-CoT-RandomSearch(D4), leverages chain-of-thought decomposition and randomized few-shot bootstrapping to deliver strong generalization.

Compared to baseline(M0), D4 improved the performance in field Complaint by 61.8%, Cause by 48.9%, and Correction by 48.2% and Translation by 56.5%.

Manual-11 stands out as the top-performing prompt in the Complaint, Cause, and Correction fields within manual prompt optimization. In comparison, D4 achieves improvements in Complaint (+25.3%), Cause (+30.7%), and Correction.

Manual-9 performance best in field Translation across manual prompt optimization. In contrast, D4 gain in Translation is relatively modest (+0.8%).

These findings validate that DSPy is exceptionally proficient in capturing compositional reasoning and sustaining organized output across various linguistic inputs. The most effective DSPy technique, D4, exhibits consistent and substantial enhancements compared to manual prompts in the majority of domains. In the field of Translation, despite carefully constructed one-shot prompts such as M9 demonstrating robust performance, D4 achieves the higher composite score (0.8326 compared to 0.8258), albeit with a marginal advantage of 0.8%. This illustrates that DSPy optimization, particularly when integrating Chain-of-Thought reasoning with randomized few-shot bootstrapping, can surpass meticulously designed human prompts in both reasoning-intensive and language-sensitive tasks.

### 4.3.3 Hybrid Prompt Strategies(LLaMA-33-70B)

Hybrid prompt optimization, which combine the strengths of manual and DSPy optimization outputs, yield the best overall performance. Among them, Hybrid-4—a fusion of best performance in manual prompt and D4 consistently achieves the highest Composite Scores across all fields: Complaint (0.7303), Cause (0.7329), Correction (0.7452), and Translation (0.8442).

Compared Hybrid-4(M11+D4) with baseline(M0) in Complaint,Cause and Correction field, it demonstrates the improvement by 70.4%, 55.9% and 58.4%. Compared Hybrid-4(M9+D4) with M0 in Translation field, prompt effect improved by 58.7%.

When compared to M11, Hybrid-4(M11+D4) offers huge relative gains: +31.9% in Complaint, +36.8% in Cause, +34.5% in Correction. These enhancements underscore the synergy attained by integrating human-curated structure with DSPy’s data-driven reasoning and comprehensive information extraction capabilities. Compared Manual-9 in Translation field, Hybrid-4(M9+D4) offer slight gain, which is +2.2%. This indicates Translation is already well-performed in M9, and Hybrid-4 maintains this performance with marginal improvement.

In conclusion, Hybrid-4 exhibits the most reliable and formidable performance across all four domains. By combining the systematic direction of manual prompts with DSPy’s compositional reasoning and optimization features, the hybrid method attains distinct benefits in semantic precision and structural integrity. It substantially improves the results of manual optimization while maintaining and augmenting the advantages of DSPy. Significantly, even in domains such as Translation, where manual prompting alone exhibits robust performance, the hybrid approach sustains and somewhat enhances this elevated baseline. These findings highlight the importance of integrating manual prompt optimization with automatic prompt optimization to design highly successful prompt techniques for multilingual information extraction and translation jobs.

### 4.3.4 Summary of Trends(LLaMA-33-70B)

The findings reveal a clear hierarchy in the effectiveness of prompts. Manual prompt engineering, through twelve iterative enhancements, establishes a solid foundation by methodically addressing issues related to structure, completeness, multilingual processing, and semantic accuracy. Manual-11 excels in structured domains because of its role-specific framing and reasoning disaggregation, whereas Manual-9 is superior for translation through one-shot in-context learning.

Building on this basis, strategic few-shot selection and automatic compositional reasoning greatly improve performance by DSPy optimization method. The best performing DSPy configuration, D4, performs better than refined manual prompts in all domains. Indicates the power of data-driven, chain-of-thought-based automatic optimization.

Hybrid strategies further elevate performance by combining human-curated instruction design with DSPy’s algorithmic efficiency. Hybrid-4 achieves the highest scores in all fields, underscoring the power of synthesis between manual intuition

and automated learning. In particular, even in translation, where M9 excels, hybrid-4 maintains and slightly improves on that strong baseline, confirming the robustness of the hybrid approach.

These trends imply that, whereas DSPy enables scalable reasoning and consistency, manual prompt engineering provides deep domain alignment and language sensitivity. Their integration using hybrid approaches results in state-of-the-art performance across all assessment criteria, providing a generalizable and pragmatic approach for multilingual information extraction tasks.

#### **4.3.5 Cross-Model Comparison: LLaMA-33-70B vs. GPT-4o**

In structured fields—Complaint, Cause, and Correction—the LLaMA-33-70B baseline achieves about 73.6% of GPT-4o’s accuracy. Manual prompting raises this ratio to 78.2% (M9) and 77.3% (M11), while DSPy optimization further improves it to 89.3%. Most notably, the hybrid strategy (Hybrid-4) brings LLaMA-33-70B up to 92.3% of GPT-4o’s performance on structured extraction.

The result lies in the Translation field: LLaMA-33-70B already reaches 87.2% of GPT-4o’s translation quality under baseline prompting and climbs to 98.3% with M9 and 97.2% with M11. Using DSPy and hybrid strategies, LLaMA-33-70B nearly matches GPT-4o (99.0%) and even slightly surpasses it with Hybrid-4 (100.3%).

These findings are significant. They demonstrate that effective prompt optimization can dramatically amplify the capability of smaller models, allowing a 70B-parameters model like LLaMA-33-70B to rival or even outperform GPT-4o in certain real-world multilingual tasks. This highlights not only the power of prompt engineering but also the potential of efficient LLMs in production-grade information extraction pipelines in automotive after-sales industry.

## **5 Discussion**

The results and conclusions of our investigation are thoroughly discussed in this chapter together with their implications. Assesses our hybrid prompt optimization strategy’s theoretical and practical relevance as well as performance. The following parts include an overview of the results, a thorough comparative analysis, consequences, limits and future areas of study.

### **5.1 Introduction and Overview of Findings**

This section briefly reviews the research background, objectives, and the central research question of this study:

How can a Hybrid Prompt Optimization approach, integrating manual prompt optimization and DSPy-based automatic optimization applied on a pre-trained LLM in other field, effectively enhance information extraction and translation performance from multilingual vehicle maintenance records?

The experiments focused on processing multilingual, noisy, and unstructured vehicle maintenance records that were manually typed by vehicle repair technicians. The results indicate that manual prompt optimization effectively captures nuanced context and ensures strict output formatting. While DSPy-based automatic optimization enhances efficiency through systematic exemplar sampling and prompt search. The hybrid prompt optimization approach, which combines both methods, yields superior performance in structured information extraction and translation accuracy.

## **5.2 Interpretation of Results(LLaMA-33-70B)**

This section presents the analytical framework used to evaluate and compare the performance of our prompt optimization methods. We examine manual prompt optimization, DSPy-based automatic optimization, and hybrid prompt optimization by the results from Chapter4.

### **5.2.1 Comparison of Manual and Automatic Methods**

The manual prompt optimization approach involves an iterative design process where explicit instructions are refined over twelve rounds. These include enforcing JSON output, role assignment, chain-of-thought reasoning, and the incorporation of one-shot or few-shot examples. Using these methods in one prompt is particularly valuable because it enables the extraction of detailed domain-specific nuances and ensures high format compliance. Although it requires significant expert input, careful manual design plays a crucial role in understanding the underlying data characteristics and tailoring the prompts accordingly.

In contrast, the DSPy-based automatic optimization employs techniques BootstrapFewShot and random sampling. Based on the ground truth table - Claim table, using evaluation metrics (ROUGE-L, BERTScore, and cosine similarity) to systematically and automatically refine the prompts. This automated method enhances efficiency and scalability, yet it may occasionally overlook the subtle domain-specific details that are critical for the automotive after-sales context. Besides, its performance is mainly decided by the training epochs and the quality of the ground truth table.

### **5.2.2 Performance Drop from Baseline (M0) to Manual-1 (M1)**

The performance decline from Baseline (M0) to Manual-1 (M1) occurred primarily due to differences in the level of exploration and flexibility allowed by each prompt. In the Baseline (M0), the fields `Complaint`, `Cause`, and `Correction` were simply marked by ellipses ('...'). It allows the language model to freely interpret and explore the relevant content. While, M1 introduced explicit yet imprecise text extraction guidelines, unintentionally limiting the model's attention to some details in these three fields. Consequently, essential information and specific nuances were occasionally overlooked by the model, which leads to a decrease in the accuracy and completeness of extraction results.

### **5.2.3 Performance Drop from Manual-11 (M11) to Manual-12 (M12)**

A comparable decline was noted from Manual-11 (M11) to Manual-12 (M12). M12 sought to augment multilingual capability by explicitly incorporating several languages (English, Chinese, French, German, and Spanish), although it resulted in excessive prompt complexity. The incorporation of comprehensive multilingual instructions inside a single prompt resulted in cognitive overload and diminished the clarity and specificity previously refined in M11. The consequent rise in token sparsity hindered the model's ability to concentrate effectively on task-specific instructions and examples, ultimately compromising the structural clarity and accuracy meticulously established in M11. Thus, while possessing theoretically superior linguistic resources, M12 exhibited reduced performance, underscoring the necessity of balancing linguistic richness with rapid clarity and conciseness.

### **5.2.4 Performance Drop from DSPy-Predict-FewShot (D1) to DSPy-CoT-FewShot (D3)**

Interestingly, although both used few-shot techniques, DSPy-CoT-FewShot (D3) fared noticeably worse than DSPy-Predict-FewShot (D1). The main difference is the increased complexity that the CoT reasoning in D3 introduces. Although CoT thinking usually improves organized thinking capacity, in a confined few-shot environment without thorough prompt refining it unintentionally raised complexity without enough examples to support such reasoning processes. The consequent uncertainty resulted in disjointed thinking and poor output quality. This suggests that, even if CoT thinking is helpful, it requires either better examples or more rapid refinement techniques (like randomized few-shot sampling found in D4) to reap its full benefits. As shown by D3, insufficiently rigorous exemplar selection of CoT reasoning can thus significantly affect model performance.

### **5.2.5 Significance of Manual Prompt Optimization**

Manual prompt optimization contributed to this research. It provided a clear baseline (M0) for comparison, but its function went beyond that. Through careful manual refinement, we capture critical domain expertise. Technical terms, multilingual nuances, and specific formatting requirements were addressed precisely. This manual approach has created a reference standard for audiences who are interested in after-sales data preprocessing. Besides, no previous relative studies have targeted prompt engineering specifically within automotive after-sales. Thus, our manual method fills a research gap in this area. It offers practical value and provides a strong foundation for future relative research.

### **5.2.6 Significance of DSPy Automatic Prompt Optimization**

This study set DSPy-based automatic optimization as a foundation, which provides a scalability, efficiency, and repeatability of the prompt optimization process. DSPy methodically uses algorithm-driven methods, like Bootstrap FewShot and randomized

example sampling, unlike manual prompt optimization. Objective criteria including ROUGE-L, BERTScore, and cosine similarity accurately guide these approaches, allowing fast designs to be adapted across multilingual inputs. As thus, this automation greatly lessens the subjectivity inherent in human-driven optimization and manual work. Moreover, DSPy creates consistent baselines and highlights structural and semantic patterns in repair logs, thereby directing later manual adjustments. Although it occasionally struggles to capture nuanced, domain-specific subtleties in a short time, DSPy’s automated approach remains invaluable, particularly when addressing large-scale data tasks and immediate practical deployment. This makes DSPy indispensable for advancing future prompt engineering research and real-world industry applications.

### **5.2.7 Significance of the Hybrid Approach**

The advantages of both automated and manual processes are combined in the hybrid approach. The hybrid approach preserves structured guidance while dynamically adjusting exemplar selection and prompt phrasing by using a high-quality manually constructed prompt as a fixed starting point and then optimizing it using DSPy. The accuracy of structured information extraction and translation fidelity have significantly increased as a result of this integration, particularly when it comes to multilingual vehicle maintenance records.

## **5.3 Cross-Model Generalization and Efficiency(LLaMA-33-70B vs. GPT-4o)**

Although GPT-4o hold a significant difference in model capacity and training resources compared to LLaMA-33-70B. From our experiment, it demonstrates that LLaMA-33-70B can achieve up to 92.3% of GPT-4o’s structured extraction performance under Hybrid-4 prompting. In the Translation task, LLaMA-3 even outperforms GPT-4o by 0.3%, highlighting the effectiveness of well-engineered prompts in compensating for model size.

These discoveries include both theoretical and practical importance. Theoretically, it indicates that prompt engineering, particularly hybrid optimization we carried out, can significantly narrow the performance disparity between relatively small LLM and relatively large LLM in practical applications. Practically, this suggests that smaller models may serve as effective substitutes for costly commercial APIs, especially in areas with financial, privacy, information security or deployment limitations.

This information is essential for industries with constrained computing budgets or those pursuing on-premise deployment. It also enables researchers and developers in resource-constrained environments to utilize LLMs efficiently without significantly compromising performance. The conclusions support further investment in prompt-based approaches as a lightweight and scalable strategy for task modification.

## **5.4 Theoretical, Practical, and Ethical Implications**

This section discusses the implications of our findings from theoretical, practical, and ethical perspectives, including model performance, deployment feasibility, and user trust in real-world applications.

### **5.4.1 Theoretical Contributions**

The work offers theoretical contributions by proving that efficient prompt optimization of pre-trained LLMs can reach competitive performance on specialized tasks, hence lowering the demand for fine-tuning. By combining automated refinement with manual prompt insights, the proposed hybrid framework increases the methodological tools accessible for modifying language models in few-shot and zero-shot environments. Especially, this effort closes a knowledge vacuum in the literature. To our knowledge, there are not many studies specifically targeted on prompt engineering for the vehicle after-sale market right now. Consequently, our efforts not only improve theoretical knowledge but also provide a useful guide for further research in this domain.

### **5.4.2 Practical Significance**

The hybrid prompt optimization framework provides significant advantages for the automotive after-sales sector. It enables the rapid and cost-effective adaptation of language models to process multilingual and unstructured maintenance records, thus improving diagnostic workflows, reducing vehicle downtime, and enhancing overall customer satisfaction. This method shows potential for wider use in businesses encountering similar issues with unstructured, multilingual data, rendering it a flexible solution for practical use.

### **5.4.3 Trust and Ethical Considerations**

The hybrid prompt optimization architecture in this is intended for practical usage in automotive after-sales diagnostics, where user confidence in the derived results poses both a practical and ethical issue. The reliability of model outputs is significant, because technicians and support staff depend on these results for decision-making and maintenance solution provision.

To ensure the reliability, this study uses the Claim Table from Volvo's official after-sales data as the ground truth. The Claim Table is based on structured repair records provided by insurance companies and authorized workshops, with multilingual technician comments translated and annotated by experienced internal professionals. The assessment is additionally reinforced by meticulously chosen measures that measure semantic similarity, contextual alignment, and lexical overlap to guarantee a thorough quality rating.

In addition, considering the potential impact that incorrect outputs will probably mislead engineers and bring bad customer experience, we plan to introduce a continuous professional review mechanism after full deployment of the system. This human-in-the-loop process will include periodic manual checks and feedback-based prompt

refinements to enhance the reliability and accountability of the output. It will be crucial in sustaining user trust and facilitating responsible AI implementation.

## 5.5 Limitations

Although the result has been shown to be promising, several limitations have to be acknowledged.

### 5.5.1 Company Domain Generalizability

The generalizability of the results may be affected by the dataset used. This study relies solely on Volvo after-sales data, which may not fully represent the diversity of maintenance records across the entire automotive industry. Specifically, we lack data from other automotive brands, making it uncertain whether the proposed prompt strategy can be directly applied to datasets from other manufacturers.

Nevertheless, the proposed approach can still serve as a valuable reference for after-sales departments in other automotive companies.

### 5.5.2 Scalability to Other Datasets, Domains, and Smaller Models

Despite the promising performance achieved within the Volvo dataset, the broader applicability of the Hybrid Prompt Optimization (HPO) strategy to other datasets, domains, or model configurations is limited by several fundamental factors.

First, the prompts, especially the manually optimized instructions, are highly domain-dependent. They implicitly encode assumptions about the structure of after-sales maintenance narratives, including technical jargon (e.g., component codes, repair abbreviations), event ordering (Complaint → Cause → Correction), and linguistic conventions (e.g., mixed use of imperative and passive voice). Transferring these prompts to other domains such as healthcare, insurance, or legal services would likely require substantial semantic adaptation, and naive reuse may yield misleading or incoherent outputs.

Second, the structured ground truth used in this study (from Volvo’s Claim table) is exceptionally well-curated, enabling reliable scoring during DSPy-based optimization. However, many real-world datasets, especially those in low-resource settings or emerging markets, lack parallel labels for extraction and translation tasks. Without supervision signals, automatic prompt selection becomes unreliable, and evaluation metrics such as BERTScore or ROUGE may not correlate with task-specific objectives.

Lastly, the proposed HPO scheme has only been evaluated on relatively powerful LLMs such as LLaMA-33-70B and GPT-4o in our study. If test the HPO method on smaller models with less than 10B parameters (e.g., LLaMA-7B or FLAN-T5-Large), it may cause severe degradation in reasoning capability, JSON format compliance, and multilingual robustness. For example, in Mayilvaghanan’s [59] research, they focus on prompt optimization for small and medium-sized LLMs (with 1B to 8B parameters). They found that smaller models show less instruction-following capacity than larger models, so many automatic prompt optimization techniques depending on model

feedback are useless when applied to small models. This suggests that the implicit reasoning steps assumed by role-based and chain-of-thought prompts exceed the architectural capacity of smaller LLMs. In addition, Soylu [60], who is from a Stanford University research team, compares different optimization strategies on tasks such as multihop question answering, mathematical reasoning, and feature classification, using small models such as Mistral-7B, LLaMA-2-7B-chat, and LLaMA-3-8B under DSPy prompt optimization. The results show that prompt engineering alone yields limited improvements. This indicates that for 7B-scale small models, manual or automatic prompt tuning alone is insufficient to compensate for the limited model capacity. Hence, the efficacy of HPO is improbable to be substantial in smaller models, as their restricted capacity inherently limits the advantages that fast optimization may provide.

In summary, the current HPO pipeline—while effective within a narrowly defined industrial setting—faces significant challenges in generalization across datasets, domains, languages, and model scales.

### 5.5.3 Infrastructure and API Limitations

The Volvo internal LLM API has a speed and token limit during deployment, which leads to real-world problems. Some batch requests may encounter major delays or fail with API problems during periods of high demand, which lowers the effectiveness of large-scale inference. These failures currently require either post-hoc patching or manual retrying, neither of which is scalable.

It is supposed to have an automatic error recovery mechanism, such as failover to backup models or retry logic with exponential backoff, to guarantee reliable operation. Furthermore, in corporate contexts, implementing specialized inference endpoints or raising the token rate restrictions will greatly increase system throughput and stability.

## 5.6 Sustainability Considerations

This study presents a sustainable and resource-efficient approach for information extraction and translation through HPO, in addition to performance enhancements. With the rising computational and environmental expenses associated with training and fine-tuning LLMs, sustainability has emerged as a paramount issue in AI implementation, particularly within industrial settings. Conventional fine-tuning typically requires GPU clusters, prolonged training durations, and substantial amounts of labeled data, collectively resulting in considerable carbon emissions and infrastructure expenses.

In contrast, the proposed HPO strategy completely avoids parameter updates, relying instead on prompt engineering to adapt pre-trained models to specific downstream tasks. All experiments in this thesis were conducted exclusively on CPU environments, without the need for GPU acceleration. This underscores the method’s computational efficiency and low-energy footprint, making it suitable for companies, organizations, and faculties with limited hardware capabilities.

More importantly, prompt engineering not only improves the performance of smaller models (e.g., LLaMA-3-70B), but also significantly narrows the performance gap between small and large LLMs. Experimental results show that, under optimized

prompts, the small LLaMA-33-70B model achieves 92% of GPT-4o’s performance in structured information extraction, and slightly outperforms GPT-4o in translation quality. This finding demonstrates that with well-crafted prompts, even lightweight models can achieve near-state-of-the-art results—without the computational and energy demands of fine-tuning or large-scale inference.

Additionally, compared to fine-tuning, prompt-based methods are more scalable, flexible, and easier to update in response to changes in task requirements, such as the introduction of new vehicle models or terminology. This makes HPO not only a low-carbon and sustainable approach, but also a more agile and cost-effective strategy for real-world deployment.

In summary, as the field moves toward more environmentally responsible AI practices, methods like HPO offer a promising direction. By balancing performance, energy efficiency, and operational feasibility, this study contributes to the growing paradigm of environmentally friendly AI where practical solutions are achieved without compromising sustainability.

## 6 Conclusion and future work

This chapter presents the general conclusions of the study and provides potential future research directions. In the following subsections, we first summarize the main findings of the study, and subsequently examine potential directions for future work.

### 6.1 Conclusion

In this study, we explored prompt engineering strategies for multilingual information extraction and translation in after-sales vehicle maintenance records. We have deployed manual prompt optimization, DSPy-based automatic prompt optimization, and hybrid prompt optimization method. It indicates that carefully designed prompts can significantly enhance performance without extensive fine-tuning.

Our experimental results show that, relative to the initial baseline, the best manual prompt improved overall performance by approximately 23% information extraction (after translation) field. Building on this, the best DSPy-based method further enhanced performance by around 27% compared to the best manual approach. Finally, in our best hybrid framework, an additional improvement of approximately 5% was achieved relative to the method based on DSPy. In total, these sequential enhancements resulted in an overall performance improvement of nearly 60% over the baseline.

For the translation of the entire text, the sequential improvements were approximately 55% from the baseline to the best manual prompt, then an additional 2% with DSPy-based optimization, and finally another 2% with the hybrid approach.

Notably, our cross-model experiments revealed that the smaller LLaMA-33-70B model, when guided by the Hybrid prompt, achieved over 92% of the greater performance of GPT-4o in structured extraction and even slightly outperformed GPT-4o in translation. This highlights that prompt engineering can effectively

bridge the performance gap between different sizes of LLMs, especially in resource-constrained or domain-specific contexts.

## **6.2 Future Work**

Future research can further improve the robustness, adaptability, and scalability of hybrid prompt optimization in real-world vehicle maintenance scenarios.

### **6.2.1 Advanced Prompt Optimization Techniques**

Future research may examine more complex methods like reinforcement learning-based prompt search, which dynamically modifies prompts based on incentive signals such as accuracy or structure conformity, even if DSPy has shown efficacy in prompt optimization. While parameter-efficient tuning techniques like prefix or soft prompts may provide low-cost domain adaptation with little labeled data, adversarial or contrastive prompt tuning may also increase robustness against ambiguous inputs.

### **6.2.2 Cross-OEM Generalization and Domain Expansion**

The framework has thus only been tested using Volvo after-sales data. To evaluate its generalizability, future research should use hybrid optimization in maintenance data from other automakers. Domain-adaptive prompt transfer and multilingual prompt templates could be applied to fit current strategies to new domains, brands, and languages with few modifications.

### **6.2.3 Dynamic Prompting in Real-Time Applications**

Static prompts might not be able to handle the range of maintenance requests that arrive in practical deployment situations. Dynamic prompt adaption systems that change prompt templates depending on real-time input properties or user profiles could be included in future systems. For example, the prompts might be customized depending on the knowledge of the technician and the regional context or automatically directed based on the semantic classification of the failure types.

### **6.2.4 Improved Evaluation and Feedback Integration**

Current evaluation relies mainly on text similarity metrics, which may not fully capture domain-specific correctness. In order to improve this point in the future, by integrating technician feedback and self-verification modules, such as logic checks or schema validation, the system could enhance trustworthiness and reduce hallucinations in high-stakes industrial applications.

### **6.2.5 Scalable Inference and Error Recovery Mechanisms**

Large-scale batch processing employing LLM APIs sometimes suffers technical constraints in practical enterprise deployment, such as rate limits, long response times,

or occasional failure. Future work should thus contain strong inference infrastructure including asynchronous task scheduling, retry methods with exponential backoff, and error recording pipelines. Deploying private or on-site LLM instances would also help to lower reliance on cloud APIs and increase latency, therefore guaranteeing seamless functioning in real-time following-sales situations.

## References

- [1] F. Bianchi, “Large language models and the future of custom, fine-tuned llms,” 2023, accessed on 9 June 2025. [Online]. Available: <https://outerbounds.com/blog/custom-llm-tuning>
- [2] E. Ohiri and R. Poole, “What is the cost of training large language models?” in *Cudo Compute Blog*, 2024, accessed on 9 June 2025. [Online]. Available: <https://www.cudocompute.com/blog/what-is-the-cost-of-training-large-language-models>
- [3] Y. Xia *et al.*, “Understanding the performance and estimating the cost of llm fine-tuning,” in *Proceedings of the 2024 IEEE International Symposium on Workload Characterization (IISWC)*, 2024, pp. 210–223, accessed on 3 April 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10763668>
- [4] N. Nananukul, K. Sisaengsuwanchai, and M. Kejriwal, “Cost-efficient prompt engineering for unsupervised entity resolution,” *arXiv preprint arXiv:2310.06174*, 2024, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/pdf/2310.06174>
- [5] A. Sabbatella, A. Ponti, I. Giordani, A. Candelieri, and F. Archetti, “Prompt optimization in large language models,” *Mathematics*, vol. 12, no. 6, p. 929, 2024, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.3390/math12060929>
- [6] V. Giordano and G. Fantoni, “Decomposing maintenance actions into sub-tasks using natural language processing: A case study in an italian automotive company,” *Computers in Industry*, vol. 164, p. 104186, 2025, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.1016/j.compind.2024.104186>
- [7] C. Lozano, “Word order in second language spanish,” in *The Handbook of Spanish Second Language Acquisition*, 2013, ch. 17, accessed on 9 June 2025. [Online]. Available: <https://wpd.ugr.es/~cristoballozano/wp-content/uploads/Lozano-2013-Word-order-in-second-language-Spanish..pdf>
- [8] J. Cheng, X. Liu, K. Zheng, P. Ke, H. Wang, Y. Dong, J. Tang, and M. Huang, “Black-box prompt optimization: Aligning large language models without model training,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024, accessed on 9 June 2025. [Online]. Available: <https://aclanthology.org/2024.acl-long.176.pdf>
- [9] Y. Lin and L. Liu, “Sixteen ways of prompt engineering,” in *AWS China Blog*, 2024, accessed on 9 June 2025. [Online]. Available: <https://aws.amazon.com/cn/blogs/china/sixteen-ways-of-prompt-engineering/>
- [10] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language

- models,” *arXiv preprint arXiv:2201.11903*, 2022, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2201.11903>
- [11] J. Kim, N. Yang, and K. Jung, “Persona is a double-edged sword: Enhancing the zero-shot reasoning by ensembling the role-playing and neutral prompts,” in *Submitted to ACL Rolling Review – August 2024*, 2024, accessed on 9 June 2025. [Online]. Available: <https://openreview.net/forum?id=nwPGunqL3F>
- [12] J. Y. Mendoza, “Large language models in web3 and loyalty programs: A study at sony’s r&d center brussels laboratory,” Master’s thesis, Université de Liège, Sony, 2024, accessed on 9 June 2025. [Online]. Available: <http://hdl.handle.net/2117/414255>
- [13] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2302.13971>
- [14] A. Olteanu and J. Waples, “What is meta’s llama 3.3 70b? how it works, use cases & more,” 2024, accessed on 9 June 2025. [Online]. Available: <https://www.datacamp.com/blog/llama-3-3-70b>
- [15] S. Feeser, “Llama.cpp architecture,” 2024, accessed on 9 June 2025. [Online]. Available: <https://stuartfeeser.com/blogs/ai-engineers/llama-model/index.html>
- [16] S. Jung, “Mastering llama — understanding rotary positional embedding (rope),” 2024, accessed on 9 June 2025. [Online]. Available: <https://medium.com/@hugmanskj/mastering-llama-understanding-rotary-positional-embedding-rope-91d1e707e95a>
- [17] B. Zhang and R. Sennrich, “Root mean square layer normalization,” 2019, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/1910.07467>
- [18] N. Shazeer, “Glu variants improve transformer,” 2020, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2002.05202>
- [19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008, accessed on 9 June 2025. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
- [20] OpenAI, “Gpt-4 technical report,” *OpenAI Technical Report*, 2023, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2303.08774>
- [21] S. M. Kerner, “Gpt-4o explained: Everything you need to know,” 2025, accessed on 9 June 2025. [Online]. Available: <https://www.techtarget.com/whatis/feature/GPT-4o-explained-Everything-you-need-to-know>

- [22] GeeksforGeeks, “Introduction to generative pre-trained transformer (gpt),” 2024, accessed on 9 June 2025. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-generative-pre-trained-transformer-gpt/>
- [23] R. Cotton and M. Crabtree, “Gpt-4o guide: How it works, use cases, pricing, benchmarks,” 2024, accessed on 9 June 2025. [Online]. Available: <https://www.datacamp.com/blog/what-is-gpt-4o>
- [24] J. Briggs, “Openai’s text embeddings v3,” 2024, accessed on 9 June 2025. [Online]. Available: <https://www.pinecone.io/learn/openai-embeddings-v3/>
- [25] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, “Bge m3 — embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation,” 2024, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2402.03216>
- [26] J. Zheng, H. Hong, X. Wang, Y. Liang, S. Wu, and J. Su, “Fine-tuning large language models for domain-specific machine translation,” *arXiv preprint arXiv:2402.15061*, 2024, accessed on 5 June 2025. [Online]. Available: <https://arxiv.org/abs/2402.15061>
- [27] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, “Scaling instruction-finetuned language models,” *arXiv preprint arXiv:2210.11416*, 2022, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2210.11416>
- [28] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” *arXiv preprint arXiv:2305.14314*, 2023, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2305.14314>
- [29] J. Chen, M. Liu, D. Shen, J. Wu, Z. Liu, and C. Zhang, “Domain specialization as the key to make large language models disruptive: A comprehensive survey,” in *arXiv preprint*, 2023, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2305.18703>
- [30] J. Pavlopoulos, E. Parlar, R. Friberg, and A. Papangelis, “Automotive fault nowcasting with machine learning and natural language processing,” in *Machine Learning (Springer)*, 2023, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.1007/s10994-023-06398-7>
- [31] P. Parthasarathy, K. Shridhar, and N. S. Keskar, “The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review,” in *arXiv preprint*, 2024, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2408.13296>

- [32] W. Jiang, D. Yin, and Y. Liu, “Effective and parameter-efficient reusing fine-tuned models,” in *arXiv preprint*, 2023, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2310.01886>
- [33] M. Josifoski, N. D. Cao, M. Peyrard, F. Petroni, and R. West, “Genie: Generative information extraction,” in *Proceedings of NAACL-HLT*, 2022, accessed on 9 June 2025. [Online]. Available: <https://aclanthology.org/2022.naacl-main.342/>
- [34] Z. Kan, L. Feng, Z. Yin, L. Qiao, X. Qiu, and D. Li, “A unified generative framework based on prompt learning for various information extraction tasks,” *arXiv preprint arXiv:2209.11570*, 2022, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2209.11570>
- [35] Y. Shen, Z. Tan, S. Wu, W. Zhang, R. Zhang, Y. Xi, W. Lu, and Y. Zhuang, “Promptner: Prompt locating and typing for named entity recognition,” in *Proceedings of ACL*, 2023, accessed on 9 June 2025. [Online]. Available: <https://aclanthology.org/2023.acl-long.698/>
- [36] Y. K. Chia, L. Bing, S. Poria, and L. Si, “Relationprompt: Leveraging prompts to generate synthetic data for zero-shot relation triplet extraction,” in *Findings of ACL*, 2022, accessed on 9 June 2025. [Online]. Available: <https://aclanthology.org/2022.findings-acl.5/>
- [37] A. S. Kwak, C. T. Morrison, D. E. Bambaauer, and M. Surdeanu, “Classify first, and then extract: Prompt chaining technique for information extraction,” in *Proceedings of the Natural Legal Language Processing Workshop*, 2024, accessed on 9 June 2025. [Online]. Available: <https://aclanthology.org/2024.nllp-1.25/>
- [38] C. Hsu, C. Zan, L. Ding, L. Wang, X. Wang, W. Liu, F. Lin, and W. Hu, “Prompt-learning for cross-lingual relation extraction,” *arXiv preprint arXiv:2304.10354*, 2023, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2304.10354>
- [39] A. Jaradat, M. Alsmirat, M. Alshraideh, A. Alsmirat, and M. Alshraideh, “Multitask learning for crash analysis: A fine-tuned llm framework using twitter data,” *Smart Cities*, 2024, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.3390/smartcities7050095>
- [40] J. Yang, T. Su, and X. Chen, “Research on text information extraction and analysis of civil transport aircraft accidents based on large language model,” in *Proceedings of the 2nd International Conference on Green Aviation (ICGA)*, 2024, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.3390/engproc2024080004>
- [41] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1877–1901, accessed on 9 June 2025. [Online]. Available:

- <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf>
- [42] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” in *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS)*, 2022, accessed on 9 June 2025. [Online]. Available: <https://dl.acm.org/doi/10.5555/3600270.3602070>
- [43] Z. Zhang, A. Zhang, M. Li, and A. Smola, “Automatic chain of thought prompting in large language models,” *arXiv preprint arXiv:2210.03493*, 2022, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/abs/2210.03493>
- [44] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, X. Zhou, E. Wang, and X. Dong, “Better zero-shot reasoning with role-play prompting,” *arXiv preprint arXiv:2308.07702*, 2024, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/html/2308.07702v2>
- [45] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia, and C. Potts, “Dspy: Compiling declarative language model calls into self-improving pipelines,” *arXiv preprint arXiv:2310.03714*, 2023, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2310.03714>
- [46] D. Cleary, “Using reinforcement learning and llms to optimize prompts,” 2025, accessed on 9 June 2025. [Online]. Available: <https://prompthub.us/blog/using-reinforcement-learning-and-llms-to-optimize-prompts>
- [47] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. Xing, and Z. Hu, “RLPrompt: Optimizing discrete text prompts with reinforcement learning,” in *Proceedings of the Association for Computational Linguistics (ACL)*, 2022, accessed on 9 June 2025. [Online]. Available: <https://par.nsf.gov/biblio/10431391-rlprompt-optimizing-discrete-text-prompts-reinforcement-learning>
- [48] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Proceedings of the Workshop on Text Summarization Branches Out*, 2004, pp. 74–81, accessed on 9 June 2025. [Online]. Available: <https://aclanthology.org/W04-1013/>
- [49] S. Santhosh, “Understanding bleu and rouge score for nlp evaluation,” 2023, accessed on 9 June 2025. [Online]. Available: <https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-rouge-score-for-nlp-evaluation-1ab334ecadcb>
- [50] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating text generation with BERT,” in *Proceedings of the International*

- Conference on Learning Representations (ICLR)*, 2020, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/pdf/1904.09675>
- [51] G. Salton, A. Wong, and C. S. Yang, “A vector space model for automatic indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.1145/361219.361220>
- [52] F. Chen, “From manual training to domain-specific adaptation through constrained prompt engineering and self-training,” 2025, accessed on 9 June 2025. [Online]. Available: <https://www.preprints.org/manuscript/202504.1301/v1>
- [53] A. Njifenjou, V. Sual, B. Jabaian, and F. Lefèvre, “Role-play zero-shot prompting with large language models for open-domain human-machine conversation,” *arXiv preprint arXiv:2406.18460*, 2024, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2406.18460>
- [54] B. Sarmah, K. Dutta, A. Grigoryan, S. Tiwari, S. Pasquali, and D. Mehta, “A comparative study of dspy teleprompter algorithms for aligning large language models evaluation metrics to human evaluation,” *arXiv preprint arXiv:2412.15298*, 2024, accessed on 9 June 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2412.15298>
- [55] T. VanSchaik, B. Cmit, and GitHub Actions, “A list of metrics for evaluating llm-generated content,” 2024, accessed on 9 June 2025. [Online]. Available: <https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/evaluation/list-of-eval-metrics>
- [56] O. Yousuf, “Improving bertscore for machine translation evaluation through contrastive learning,” Master’s thesis, Uppsala University, 2022, accessed on 9 June 2025. [Online]. Available: <https://www.diva-portal.org/smash/get/diva2:1671848/FULLTEXT01.pdf>
- [57] I. Karadeniz and A. Özgür, “Linking entities through an ontology using word embeddings and syntactic re-ranking,” *BMC Bioinformatics*, vol. 20, no. 1, p. 156, 2019, accessed on 9 June 2025. [Online]. Available: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-2678-8>
- [58] K. Hämmerl, J. Libovický, and A. Fraser, “Understanding cross-lingual alignment—a survey,” *arXiv preprint arXiv:2404.06228*, 2024, accessed on 9 June 2025. [Online]. Available: <https://arxiv.org/html/2404.06228v2>
- [59] K. Mayilvaghanan, V. Nathan, and A. Kumar, “Propel: Prompt optimization with expert priors for small and medium-sized llms,” in *Proceedings of the 6th Workshop on Knowledge-Aware NLP (KnowledgeNLP)*, 2025, accessed on 9 June 2025. [Online]. Available: <https://aclanthology.org/2025.knowledgenlp-1.25.pdf>

- [60] D. Soylu, C. Potts, and O. Khattab, “Fine-tuning and prompt optimization: Two great steps that work better together,” *arXiv preprint arXiv:2407.10930*, 2024, accessed on 9 June 2025. [Online]. Available: <https://ar5iv.labs.arxiv.org/html/2407.10930>

## A Appendix: Sample Records from Datasets

### A.1 Sample from Claim Table (Ground Truth)

**Table A1:** Sample record from Claim table used for ground truth.

Column	Example Value
service_id	SRV20240315-001
truck_id	VOLVO-TRK-001
service_date	2024-03-15
detail	Kunden beschwert sich über ungewöhnliche Motorgeräusche beim Beschleunigen. Diagnose ergab lockeren Turboladeranschluss. Anschluss wurde befestigt und getestet.
Translation	Customer complained about unusual engine noises during acceleration. Diagnosis found a loose turbocharger connection. The connection was tightened and tested.
Complaint	Unusual engine noises during acceleration
Cause	Loose turbocharger connection
Correction	Tightened and tested the turbocharger connection

**Note:** This is a synthetic example and does not reflect real service data.

### A.2 Sample from TMR Table (Target Corpus)

**Table A2:** Sample record from TMR (Truck Maintenance Records) table used for model inference.

Column	Example Value
service_id	SRV20240310-007
truck_id	VOLVO-TRK-007
service_date	2024-03-10
country_code	DE
detail	Motor macht laute Geräusche beim Gasgeben. Techniker prüfte den Turbolader und stellte eine lose Verbindung fest. Verbindung wurde repariert.

**Note:** This is a synthetic example and does not reflect real service data.

## B Appendix: pseudocode from prompt optimization

### B.1 Appendix: pseudocode from Dspy method

---

**Algorithm 1** RepairTranslatorExtractor Module (DSPy Prompt Optimisation)

---

**Require:** *detail* ▶ raw maintenance-log text  
**Ensure:** *response* ▶ structured result

- 1: **function** REPAIRTRANSLATOREXTRACTOR(*detail*)
- 2:     *response* ← PREDICT/CoT<sub>RepairExtraction</sub>(*detail*) ▶ invoke predictor/CoT
- 3:     **return** *response*
- 4: **end function**

---

### B.2 Appendix: pseudocode from Hybrid method

---

**Algorithm 2** RepairTranslatorExtractor Module (Hybrid Prompt Optimisation)

---

**Require:** *detail* ▶ raw maintenance-log text  
**Ensure:** *response* ▶ structured JSON with language, Translation, Complain, Cause, Correction

- 1: **function** REPAIRTRANSLATOREXTRACTOR(*detail*)
- 2:     *prompt* ← BUILD\_PROMPT(*detail*) ▶ embed the log in the best manual prompt template
- 3:     *response* ← PREDICT/CoT<sub>RepairExtraction</sub>(*prompt*) ▶ invoke the LLM and obtain structured output
- 4:     **return** *response*
- 5: **end function**
- 6: **procedure** BUILD\_PROMPT(*detail*) ▶ construct the prompt as the start point
- 7:     **Data:** fixed instruction template (language detection → translation → field extraction) with placeholder {*detail*}
- 8:     **Result:** fully populated *prompt*
- 9:     **return** template with placeholder replaced
- 10: **end procedure**

---