Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Enhancing Large Language Models for Portuguese Language: Lexical Normalisation Case Study

Aprimorando LLMs para Língua Portuguesa: Caso de Estudo da Normalização Lexical

Vinícius Di Oliveira

A Thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (Computer Science) in the University of Brasilia

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Enhancing Large Language Models for Portuguese Language: Lexical Normalisation Case Study

Aprimorando LLMs para Língua Portuguesa: Caso de Estudo da Normalização Lexical

Vinícius Di Oliveira

A Thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (Computer Science) in the University of Brasilia

Prof. Dr. Li Weigang (Advisor)
CIC/UnB

Prof. Dra. Aline Marins Paes Carvalho
UFF

Prof. Dr. Rodolfo Ipolito Meneguette
USP

Prof. Dr. Pedro Garcia Freitas
UnB

Prof. Dra. Edna Dias Canedo
UnB

Prof. Dra. Cláudia Nalon
Coordenadora do Programa de Pós-graduação em Informática

Brasília, Dezembro , 2025

Dedication

I dedicate this thesis to my parents, Nivaldo and Aleth, who have always encouraged me to pursue my goals, and to my children, Jordana, Pedro and João Paulo, who have been a constant source of motivation during this challenging journey.

Acknowledgements

I would like to thank my advisor, Prof. Li Weigang, for his consistent support, guidance, generosity, and patience. His encouragement, motivation, and valuable feedback were essential to completing this course.

I also sincerely thank the research partners Pedro Carvalho Brom and Yuri Façanha Bezerra, PPGI students, without whom the task of coding and organising the GitHub repository for this work would have been nearly impossible. I would like to extend special thanks to the members of the evaluation board for their valuable comments, which enriched this work.

I would like to thank my colleagues and professors at TransLab, PPGI, and CIC/UnB for their friendship and support during my doctoral studies. I also extend my sincere thanks to Professor Cláudia Nalon for her coordination and continued support throughout the programme. Similarly, I am very grateful to Professor Victor R. R. Celestino for the rewarding experience of participating in the AI team of the *Gov.br* project, a partnership between FINATEC-UNB and the SGD of the Federal Government. I could also not have undertaken this journey without the understanding and support of my colleagues, Fabíola Venturini and Sérgio Augusto Pará Bittencourt Neto.

Finally, I am grateful to the University of Brasília and the Brazilian public education system, which provide equal opportunities for all students who wish to participate in scientific research.

Resumo

Esta tese de doutorado aborda o desafio premente de aprimorar o desempenho e a confiabilidade de Modelos de Linguagem de Grande Escala (LLMs) em tarefas complexas de classificação e normalização no contexto da língua portuguesa. O trabalho foca especificamente na normalização lexical de descrições de mercadorias e de dados de endereçamento (CEP), essenciais para sistemas fiscais e logísticos. O problema central estudado reside na dificuldade de classificar mercadorias de forma eficaz conforme a Nomenclatura Comum do Mercosul (NCM) e de padronizar códigos postais (CEP) a partir de descrições textuais livres. No contexto da língua portuguesa, a escassez de recursos e de modelos de linguagem otimizados para o idioma apresenta obstáculos significativos. Modelos tradicionais enfrentam dificuldades consideráveis para lidar com a especificidade e a complexidade inerentes a essas tarefas, especialmente em linguagens não-inglesas. A maior parte dos LLMs proeminentes é treinada predominantemente em inglês, o que restringe o desempenho em contextos técnicos e de domínio específico, como a fiscalização tributária.

A tarefa de normalização lexical, que consiste em converter texto não-padrão ou irregular em uma forma canônica e uniforme, é crucial neste cenário. Descrições de produtos ou endereços apresentam grande variabilidade textual, incluindo abreviações, inconsistências e erros ortográficos. Por exemplo, um produto pode aparecer de múltiplas formas em uma Nota Fiscal Eletrônica (NFe), como "T. Pap. FDupla" em vez de "Toalha de Papel Folha Dupla". Sem uma normalização eficaz, sistemas que dependem de correspondência exata falham na classificação. A relevância do problema transcende o processamento linguístico puro. A classificação precisa dos códigos, como os códigos NCM (compostos por oito dígitos), afeta diretamente a conformidade fiscal, as estatísticas de comércio internacional e o monitoramento da receita pública. Erros na atribuição do código NCM podem resultar em perdas financeiras, alterações nos cálculos de tarifas, atrasos no desembaraço aduaneiro e penalidades regulatórias decorrentes de declaração incorreta. Esta tese, portanto, busca desenvolver uma solução que não apenas processe o português de forma eficiente, mas também melhore a precisão e a relevância das saídas de classificação em domínios especializados e de alto risco.

Para enfrentar essas limitações, a pesquisa propõe uma arquitetura híbrida inovadora,

denominada *Tri-Layered Intelligent Framework for LLMs (TRINITY-LLM)*. Essa estrutura modular e integrada visa combinar o ajuste fino eficiente, a geração aumentada por recuperação de informações (*Retrieval Augmented Generation - RAG*) e a avaliação estatisticamente robusta. O TRINITY-LLM é composto por três estágios complementares, cada um correspondendo a uma inovação metodológica desenvolvida na tese: 1) Modelo Simplificado Lógico Inteligente de Ajuste Fino com Geração Aumentada por Recuperação de Informações (SLIM-RAFT): focado no ajuste fino custo-eficiente e na normalização lexical em português. 2) Geração Aumentada por Recuperação em Dois Passos (Two-Step RAG): que aprimora a precisão da recuperação de informações por meio da filtragem contextual orientada por metadados. 3) Modelos Mistos Integrados com Análise em *Bootstrap* (IMMBA): que estabelece uma estrutura estatística rigorosa para a avaliação de LLMs, quantificando a variabilidade de desempenho.

O *SLIM-RAFT* foi concebido para superar o viés linguístico anglocêntrico e as exigências computacionais das *pipelines* tradicionais de ajuste fino. Ele estende os princípios do *Retrieval-Augmented Fine-Tuning (RAFT)*, que treina o modelo para utilizar informações externas recuperadas durante o *fine-tuning*. O RAFT original, no entanto, é quase inviável devido à dependência de outro LLM de alta capacidade para construir o *corpus* de treinamento baseado na Cadeia de Pensamento (*Chain-of-Thought - CoT*). O *SLIM-RAFT* simplifica a fase de geração de dados por meio do paradigma de raciocínio *Sequence-of-Sets (SoS)* no prompt de treinamento, criado e desenvolvido neste trabalho. O SoS representa o raciocínio como sequências lógicas ordenadas entre conjuntos, espelhando a inferência dedutiva: $a \in A, A \subseteq B, \therefore a \in B$. Isso reduz drasticamente o custo de criação do conjunto de dados de ajuste fino, mantendo a consistência de aprendizado. O *fine-tuning* foi realizado em modelos leves, como o *TeenyTinyLLaMA (TTL)*, que já é pré-treinado nativamente em português brasileiro, utilizando a técnica de *Low-Rank Adaptation (LoRA)*, que reduz o custo computacional. O conjunto de dados extenso utilizado incluiu mais de *240.000 registros da NCM* provenientes de notas fiscais eletrônicas (NFe's).

O *Two-Step RAG* aborda a limitação central dos sistemas RAG convencionais, que dependem fortemente da similaridade semântica, o que frequentemente resulta em ruído quando os *prompts* são vagos. Ele organiza a recuperação contextual em duas fases explícitas: 1) Recuperação Comum (R1): Uma busca ampla e irrestrita, garantindo alto *recall* e fornecendo um conjunto inicial de candidatos. 2) Extração e Aplicação de Metadados (M e R2): Um LLM é solicitado a extrair atributos estruturados (metadados, como NCM ou rótulo) diretamente do *prompt* original. Esses atributos são usados para refinar a recuperação por meio de uma filtragem direcionada aos candidatos semânticos iniciais. Essa abordagem equilibra amplitude e precisão, distinguindo-se de métodos como o Multi-

Meta-RAG, que dependem de esquemas rígidos de metadados. A formulação matemática do Two-Step RAG inclui: i) Função de Recuperação Comum. ii) Função de Extração de Metadados. iii) Função de Filtragem.

O *Integrated Mixed Models with Bootstrap Analysis (IMMBA)* é uma estrutura estatística que garante a avaliação rigorosa, interpretável e reprodutível dos LLMs. Ele utiliza modelos lineares multivariados mistos em combinação com o reescalonamento *bootstrap* não-paramétrico (com 1000 iterações). A formulação hierárquica do LMM decompõe a variância observada em fontes sistemáticas e aleatórias: $\mathrm{Var}(Y) = \sigma_f^2 + \sigma_P^2 + \sigma_e^2$ onde $\mathrm{Var}(Y)$ é a variância total, σ_f^2 é a variância explicada por efeitos fixos (arquitetura do modelo, método de recuperação), σ_P^2 representa a variabilidade devida ao fraseado do *prompt* (efeito aleatório), e σ_e^2 captura o erro residual, ou seja, aquilo não explicado pelas variáveis anteriores. A avaliação dos modelos foi baseada em quatro dimensões principais, pontuadas em uma escala ordinal de 0 a 10: Qualidade, Acordo (alinhamento semântico), Precisão (correção factual) e Alucinação (penalização de conteúdo não suportado).

Os resultados empíricos validaram a eficácia da abordagem proposta. O modelo *SLIM-RAFT* (aplicado ao TTL de 160M de parâmetros) apresentou desempenho comparável ao de modelos de ponta na classificação NCM. O SLIM-RAFT alcançou uma precisão média de *8,63 (0 a 10)*, o que representa um aumento significativo em relação aos modelos observados. Notavelmente, superou o *ChatGPT 4.0* (modelo proprietário de bilhões de parâmetros), que alcançou uma média de *4,5 (0 a 10)* no mesmo protocolo de avaliação, e o TTL base, que marcou apenas 0,2. O sucesso foi replicado no domínio do CEP: o modelo *SLIM-RAFT-CEP* (160M parâmetros) obteve uma média de *1,92*, um desempenho comparável ao do modelo de ponta *GPT-4.o mini* (1,90). Isso confirma que a metodologia *SoS* e o ajuste fino direcionado são suficientes para capturar a lógica de domínio específico, mesmo em arquiteturas compactas.

Os experimentos com o Two-Step RAG compararam seu desempenho com o RAG Convencional em cinco arquiteturas de LLMs (incluindo Mistral-7B, Deepseek-chat e GPT-4o-mini). Os resultados confirmaram que a recuperação orientada por metadados é eficaz. O Two-Step RAG *melhorou a Qualidade em 2,20×, o Acordo em 2,63× e a Precisão em 2,91*×, em média, em comparação com a abordagem RAG comum. O modelo *Mistral-7B* teve o maior ganho relativo em Precisão ($\times 4,54$), mas seu desempenho absoluto permaneceu inferior ao dos modelos de ponta. Em contraste, o *RAG Comum* reduziu significativamente a Qualidade média (2,73 vs 5,30) e a Precisão (2,12 vs 4,70), resultando em taxas de *Alucinação mais altas* (4,67 vs 3,93 para Two-Step RAG). Isso reforça a necessidade de refinamento contextual. A análise de correlação de Spearman ρ, que é o indicador numérico que mede a força e a direção da relação entre duas variáveis, revelou associações positivas fortes e monotônicas entre Qualidade, Acordo e Precisão

($\rho \in [0,955; 0,977]$), e correlações negativas fracas, mas consistentes, com Alucinação (aproximadamente $-0,28$).

O framework *IMMBA* forneceu a base estatística para interpretar esses resultados com confiança. A análise de *decomposição da variância* revelou as fontes de incerteza no desempenho dos LLMs: i) *Efeitos Fixos* (arquitetura do modelo, método RAG, etc.): *23,2%* da variância total ($\sigma_f^2 = 2,14$). ii) *Variabilidade do* Prompt (efeito aleatório): *7,2%* da variância total ($\sigma_P^2 = 0,66$). iii) *Erro Residual e Ruído Estocástico*: *69,6%* da variância total ($\sigma_e^2 = 6,42$). Essa grande proporção de variância não explicada pelos fatores controlados reforça a necessidade de metodologias estatísticas robustas, como o *bootstrap*, para inferir a confiabilidade dos resultados. O uso do *bootstrap* aumentou a estabilidade das estimativas e produziu intervalos de confiança mais estreitos, o que comprova que as melhorias reportadas são estatisticamente significativas e robustas.

O trabalho faz uma *contribuição significativa* ao campo do processamento de linguagem natural multilíngue, particularmente para a língua portuguesa. As contribuições metodológicas incluem a técnica de ajuste fino *SLIM-RAFT*, a estratégia de *prompting Sequence-of-Sets (SoS)*, a metodologia de recuperação avançada *Two-Step RAG*, e o *framework* estatístico *IMMBA*. O sucesso em adaptar modelos compactos (160M) para alcançar o desempenho de modelos proprietários massivos (GPT-4o mini) em tarefas de domínio específico no âmbito do processamento de linguagem natural, por meio de estratégias de *fine-tuning* e de recuperação otimizadas, sugere um caminho promissor para o desenvolvimento de inteligência artificial sustentável e linguisticamente inclusiva. O TRINITY-LLM oferece uma solução completa e estatisticamente fundamentada para enfrentar o desafio de alto risco da normalização lexical e da classificação de dados fiscais em português.

Palavras-chave: Grandes Modelos de Linguagem, Geração Aumentada por Recuperação, Processamento de Língua Portuguesa; Engenharia de Prompts

Abstract

This thesis addresses the challenge of enhancing the performance and reliability of Large Language Models (LLMs) for lexical normalisation of short Portuguese texts, focusing on structured classification tasks such as the *Nomenclatura Comum do Mercosul* (NCM) and postal codes (CEP). Traditional language models, typically trained on corpora predominantly in English, encounter significant difficulties in handling the specificity, ambiguity, and linguistic variability (including abbreviations and spelling errors) inherent to specialised Portuguese tasks such as tax auditing and logistics. Inaccuracies in the classification of mandatory fiscal codes, such as the NCM in Brazilian electronic invoices, can lead to financial losses and operational inefficiencies.

To overcome these limitations, this research proposes an integrated and innovative architecture — the **Three-Layer Intelligent Framework for LLMs (TRINITY-LLM)**. The proposed model is structured into three interdependent components, each representing a core study within this thesis: 1) **Simplified Logical Intelligent Fine-Tuning Model with Retrieval-Augmented Generation (SLIM-RAFT)** — a cost-efficient fine-tuning method specifically designed for adaptation to technical Portuguese domains. SLIM-RAFT extends the logic of Retrieval-Augmented Fine-Tuning (RAFT) by simplifying the data generation phase, replacing lengthy narrative explanations with a new reasoning paradigm termed Sequence-of-Sets (SoS). The SoS prompting technique represents reasoning as a set of ordered logical relations, preserving the interpretability of the Chain-of-Thought (CoT). This component focuses on the lexical normalisation of non-standard descriptions. 2) **Two-Step RAG for Advanced Retrieval and Metadata Filtering** — an enhanced retrieval methodology that overcomes the limitations of conventional RAG systems. The *Two-Step RAG* operates in two explicit phases: (i) Common Retrieval (R1), a broad, unfiltered search for high recall; and (ii) Metadata Extraction and Application (M and R2), in which an LLM dynamically extracts structured attributes from the original prompt to refine the search and filter the initial semantic candidates. This design effectively decouples semantic retrieval from metadata application. 3) **Integrated Mixed Models with Bootstrap Analysis (IMMBA)** — a rigorous statistical framework providing a robust and transparent basis for LLM evaluation. IMMBA em-

ploys linear mixed models combined with non-parametric bootstrap resampling, enabling the decomposition of observed variability into fixed and random sources.

The methodology involved extensive experimentation with open-source models, including the 160M-parameter *TeenyTinyLLaMA (TTL)*, fine-tuned on domain-specific Portuguese data. More than 240,000 NCM records were extracted from Brazilian electronic invoices (*Notas Fiscais Eletrônicas* – NFes) for model development.

Empirical results validate the effectiveness of the proposed approach. The SLIM-RAFT model (TTL 160M *fine-tuned*) achieved a mean NCM classification score of 8.63 (standard deviation 2.30), significantly outperforming GPT-4.0 (4.5) and the baseline TTL (0.2) under the same evaluation protocol. These results demonstrate that SoS logic is sufficient to capture domain reasoning even in compact models. SLIM-RAFT also generalised successfully to the CEP classification task, achieving performance comparable to GPT-4o-mini despite its dramatically smaller size.

The Two-Step RAG method consistently improved performance metrics over conventional RAG, achieving on average a 2.20× gain in quality, 2.63× in agreement, and 2.91× in accuracy across all tested models. This enhancement in contextual retrieval precision led to reduced hallucination rates. Statistical analysis through IMMBA confirmed the robustness of the evaluation: decomposition of total variance revealed that fixed factors (architecture, retrieval method, and decoding parameters) accounted for 23.2% of overall variability, random factors associated with prompt phrasing contributed 7.2%, and the remaining 69.6% was attributed to residual stochasticity and experimental noise. These insights highlight the need for statistically grounded frameworks to quantify and interpret uncertainty in LLM performance.

This research makes a significant contribution to the field of multilingual Natural Language Processing. The TRINITY-LLM framework offers not only a practical solution for improving product classification according to the NCM, but also establishes a precedent for how hybrid models — combining efficient fine-tuning, metadata-augmented retrieval, and rigorous statistical evaluation — can enhance the reliability and transparency of language models in specialised and high-stakes domains such as regulatory auditing.

Keywords: Large Language Models, Retrieval-Augmented Generation, Portuguese Language Processing, Prompt Engineering

Contents

List of Figures

List of Tables

Acronyms

CEP Address Postal Code.

CV Coefficient of Variation.

IMMBA Integrated Mixed Models with Bootstrap Analysis.

LLM Large Language Model.

LLMs Large Language Models.

NCM Mercosur Common Nomenclature.

NLP Natural Language Processing.

PEFT Parameter-efficient Fine-Tuning.

RAFT Retrieval Augmented Fine-Tuning.

RAG Retrieval Augmented Generation.

SD Standard Deviation.

SLIM-RAFT Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning.

TRINITY-LLM Tri-Layered Intelligent Framework for LLMs.

Two-Step RAG Two-Step RAG for Advanced Retrieval and Metadata Filtering.

Chapter 1

Introduction

Large Language Models (LLMs) have rapidly transformed Natural Language Processing (NLP) by enabling generative reasoning, multilingual understanding, and context-aware classification across a wide range of tasks [1, 2, 3]. Nevertheless, their performance in non-English contexts and domain-specific applications remains inconsistent [4, 5], especially when applied to highly structured problems such as the lexical normalisation of short-text descriptions. This thesis addresses this gap by enhancing LLMs for the Portuguese language through a case study focused on the lexical normalisation of product descriptions and address data used in fiscal and logistical systems.

The central problem addressed in this research concerns the difficulty of classifying and standardising free-text descriptions of goods and addresses into structured code systems, notably the Mercosur Common Nomenclature (NCM) and Address Postal Code (CEP). In practice, such classification errors can lead to financial losses, regulatory penalties, and inefficiencies in customs and tax inspection. Traditional methods based on deterministic rules or static classifiers fail to capture the linguistic variability and ambiguity inherent in real-world textual data. This motivates the development of intelligent models capable of understanding contextual nuances and performing accurate normalisation at scale.

The relevance of this problem extends beyond linguistic processing, as accurate code classification directly affects fiscal compliance, international trade statistics, and public revenue monitoring. From an economic perspective, incorrect classification of goods under the Harmonised System (HS) [6] or NCM [7] codes may distort tariff applications and trade balances [8]. Politically, ensuring accurate fiscal data supports transparency and efficiency in public administration. Therefore, improving lexical normalisation accuracy has both operational and systemic implications.

To address this, the thesis proposes a novel integrated architecture — the **Tri-Layered Intelligent Framework for LLMs: TRINITY-LLM** — which combines efficient fine-tuning, advanced retrieval-augmented reasoning, and statistically robust evaluation.

TRINITY-LLM is structured in three layers, each corresponding to one of the original studies developed in this research: (i) *Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning (SLIM-RAFT)* introduces a cost-efficient fine-tuning method for lexical normalisation in Portuguese [9]; (ii) *Two-Step RAG for Advanced Retrieval and Metadata Filtering (Two-Step RAG)* enhances retrieval accuracy through metadata-driven contextual filtering [10]; and (iii) *Integrated Mixed Models with Bootstrap Analysis (IMMBA)* establishes a statistically principled framework for LLM evaluation based on Linear Mixed Models and bootstrap resampling [11]. Together, these layers form a pipeline capable of fine-tuning, retrieval, and evaluation in a unified, reproducible manner, achieved through fixed data partitions, versioned prompt templates, deterministic retrieval configurations, and statistically grounded evaluation protocols that explicitly model variability across runs and prompts.

Empirical results validate the effectiveness of the proposed approach. The Two-Step RAG method improved quality by 2.20×, agreement by 2.63×, and accuracy by 2.91× on average across models. At the same time, IMMBA's variance decomposition revealed that 23.2% of total variability was due to fixed factors (model and configuration), 7.2% to prompt phrasing, and 69.6% to residual stochasticity. These results demonstrate that integrating retrieval-augmented fine-tuning with robust statistical evaluation significantly enhances LLM reliability in structured, high-stakes domains such as fiscal data classification.

This chapter introduces the study's context and motivation, defines the research problem and objectives, describes the devised model, and summarises the contributions of this doctoral research.

1.1 Motivation

In the dynamic realm of international trade, customs brokers, exporters, and importers confront the critical task of accurately classifying goods under the HS Code, which underpins the NCM code [7, 12, 13]. Likewise, in logistics and postal systems, the CEP ensures geospatial precision and operational efficiency [14]. The HS structure and heading can be seen in Tables 1.1 and 1.2. The NCM structure is in the Table 1.3.

Size	**Index**
2 digit (01-97)	Chapter
4 digit (01.01 - 97.06)	Heading
6 digit (0101.21 - 9706.00)	Subheading

Table 1.1: Structure of the Harmonized System Codes [12].

Code	Description
08.08	Apples, pears and quinces, fresh.
0808.10	- Apples
0808.30	- Pears
0808.40	- Quinces
...	...
08.13	- Fruit, dried, other than that of headings 08.01 to 08.06; mixtures of nuts or dried fruits of this Chapter.
0813.10	- Apricots
0813.20	- Prunes
0813.30	- Apples
0808.40	- Other Fruit
0808.50	- Mixtures of nuts or dried fruits of this Chapter

Table 1.2: Headings 08.08 and 08.13 in the Harmonized System [12].

Size	Index
2 digit (01-97)	Chapter
4 digit (01.01 - 97.06)	Heading
6 digit (0101.21 - 9706.00)	Subheading
7 digit (0101.21.1 - 9706.00.9)	Item
8 digit (0101.21.10 - 9706.00.90)	Sub-item

Table 1.3: Structure of the Mercosur Common Nomenclature (NCM) Codes [7].

However, both domains share a persistent challenge: the need to transform unstructured, often abbreviated or inconsistent free-text descriptions into standardised codes — a process known as *lexical normalisation*. Lexical normalisation is the process of converting diverse textual expressions into a canonical, uniform form that preserves semantic meaning while enabling machine interpretation and structured classification. In the context of this research, normalising short product or address descriptions so they can be mapped consistently to their corresponding NCM or CEP codes. The CEP components are shown in Table 1.4.

Digit	Area
X0000-000:	Region
0X000-000:	Sub-region
00X00-000:	Sector
000X0-000:	Subsector
0000X-000:	Subsector divisor
00000-XXX:	Distribution suffix

Table 1.4: Components of the Postal Code (CEP) Code [14].

This problem is particularly acute in fiscal and logistical databases, where textual variability arises from human-authored descriptions. A single product, such as "*Toalha de Papel Folha Dupla*" (double-ply paper towel), may appear as "*T. Pap. FDupla*" or "*Toalha Papel F2*," complicating automated classification. Similarly, addresses like "*SQN*

209 Bloco E Asa Norte Brasília" may appear as "*SBN 209 Bl E A. N. BSB.*" Without effective normalisation, systems that rely on pattern matching or simple embeddings often misclassify these entries. Table 1.5 shows some abbreviations that can be easily found on invoices, electronic bills or export documentation.

Abbreviation	**Full description**
English	
Coc. 2L	= Coca-Cola 2 Liters
P. W. Rice	= Parboiled White Rice
Portuguese	
Fr. Desc.	= Fralda descartável (*Disposable diaper*)
T. Pap. FDupla	= Toalha de Papel Folha Dupla (*Double Ply Paper Towel*)
French	
EDT	= Eau de Toilette
EDP	= Eau de Parfum

Table 1.5: Abbreviation Examples.

These inconsistencies are not merely linguistic but have concrete economic and legal consequences. Errors in NCM code assignment can alter tariff calculations, delay customs clearance, or even trigger fiscal penalties for misdeclaration. In Brazil, each electronic invoice (*Nota Fiscal Eletrônica*) must include an NCM code as a legal requirement [15]. Similar issues occur in CEP classification, where inconsistent address data can impair logistics, taxation, and public service delivery. It is possible to examine variations of the descriptions for the same CEP code in Table 1.6. Therefore, accurate lexical normalisation is essential for both commercial compliance and public governance.

Language	**Address description**
English	
1	= Super Block North 209, Block E, Asa Norte Neighbour., City: Brasília
2	= SBN 209, bl E, Asa N. Brasília
3	= SBNorth 209, block E, A. Norte BSB
Portuguese	
1	= Super Quadra Norte 209, Bloco E, Bairro Asa Norte, Cidade: Brasília
2	= SQN 209, bl E, Asa N. Brasília
3	= SQNorte 209, bloco E, A. Norte Brasília

Table 1.6: Address descriptions for the Postal Code (CEP) 70.854-050.

Despite advances in NLP, traditional machine learning models struggle with the contextual disambiguation required in such structured domains. The diversity of linguistic forms, domain-specific abbreviations, and multi-language terms presents additional difficulties. Addressing this challenge requires models that can understand context, adapt to domain-specific patterns, and remain robust across prompts—objectives that motivated this doctoral research.

1.2 Problem Statement and Objectives of the Research

The central problem of this research is how to improve the performance and reliability of Large Language Models when applied to the lexical normalisation of Portuguese-language short texts, particularly in structured classification tasks involving the Mercosur Common Nomenclature (NCM) and Address Postal Code (CEP).

Although recent LLMs have shown remarkable generative and reasoning capabilities, their effectiveness in domain-specific, non-English contexts remains inconsistent. High stochasticity, sensitivity to prompt phrasing, and lack of evaluation rigour hinder their adoption in high-stakes applications such as fiscal auditing or compliance verification. This research aims to overcome these limitations by developing an integrated framework that combines fine-tuning, retrieval-augmented reasoning, and statistically grounded evaluation.

General Objective: To improve the performance of Large Language Models for the classification of product descriptions in Brazilian Portuguese, by developing and validating an integrated methodology for fine-tuning, retrieval, and evaluation that explicitly addresses the challenges posed by lexical variability in fiscal and logistical datasets.

Specific Objectives:

- To develop a cost-efficient fine-tuning approach for domain-specific adaptation and lexical normalisation in Portuguese.
- To design a metadata-aware retrieval method that dynamically extracts contextual metadata from prompts to improve retrieval precision.
- To construct a statistically robust evaluation framework based on linear mixed models and bootstrap resampling to quantify performance variability and interaction effects.

Collectively, these objectives establish a unified pipeline that enhances LLM interpretability, accuracy, and reproducibility in structured classification scenarios.

1.3 Devised Model

The proposed **TRINITY-LLM** framework integrates three complementary components: SLIM-RAFT, Two-Step RAG, and IMMBA. They form a tri-layered architecture for training, retrieval, and evaluation. The first layer, Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning (SLIM-RAFT) [9], provides an efficient fine-tuning

process for domain-specific adaptation in Portuguese, reducing computational cost while maintaining performance. It leverages retrieval-augmented strategies to expose the model to structured contextual data during fine-tuning, optimising its lexical normalisation capability.

The second layer, Two-Step RAG for Advanced Retrieval and Metadata Filtering [10], enhances retrieval accuracy through a two-phase mechanism. The first phase performs a broad semantic search, while the second extracts structured metadata (e.g., code, label, item) from the prompt using an LLM and applies it as a filter for refined contextual retrieval. This design reduces retrieval noise and improves answer specificity, achieving accuracy gains of up to 4.5× compared with traditional RAG methods.

The final layer, Integrated Mixed Models with Bootstrap Analysis (IMMBA) [11], introduces a statistically principled evaluation methodology. It combines Linear Mixed Models with bootstrap resampling to decompose output variability into fixed effects (model architecture, retrieval type, decoding parameters), random effects (prompt phrasing), and residual noise. This allows precise quantification of stochastic variability and identifies significant interactions between LLM configurations, thereby improving evaluation transparency and reproducibility. Together, these layers form the TRINITY-LLM architecture: a unified, end-to-end pipeline for domain adaptation, retrieval optimisation, and statistical evaluation of LLMs in Portuguese.

1.4 Contributions

This research provides methodological, empirical, and applied contributions to the field of large language models, particularly in non-English and domain-specific contexts. The main contributions are summarised as follows:

- **SLIM-RAFT:** A fine-tuning technique for domain-specific LLM adaptation in Portuguese, combining retrieval-augmented training with efficiency-driven optimisation to enhance lexical normalisation while reducing computational cost. In addition to the derived academic output, a patent (Number BR10-2025-001971-0) was registered for this framework due to its innovative characteristics.
- **Sequence of Sets (SoS) Prompting:** A structured prompting strategy that improves consistency across inference runs by balancing semantic coverage and contextual coherence (incorporated in the SLIM-RAFT patent).
- **Two-Step RAG:** An advanced retrieval-augmented methodology that incorporates metadata extraction and filtering, achieving average improvements of 2.2× in quality, 2.6× in agreement, and 2.9× in accuracy over conventional retrieval methods. This

framework produced a small language model (150 million parameters) that scored as well as a state-of-the-art LLM.

- **IMMBA Evaluation Framework:** A robust statistical approach integrating Linear Mixed Models and bootstrap analysis to decompose performance variability and measure prompt-induced randomness, demonstrating that 23.2% of total variance arises from fixed effects, 7.2% from prompt phrasing, and 69.6% from residual stochasticity.

Together, these contributions advance the scientific understanding of how LLMs can be adapted, retrieved, and evaluated in structured, multilingual, and high-stakes domains. The proposed TRINITY-LLM framework not only improves the accuracy and reliability of Portuguese-language models but also provides a replicable methodology applicable to other non-English languages and domain-specific contexts.

1.5 Organisation of the Thesis

This thesis is organised into eight chapters, each designed to systematically address the research objectives and provide a comprehensive understanding of the development and application of Large Language Models (LLMs) tailored for the Portuguese language.

- **Chapter 1** introduces the motivation behind the study, outlines the problem statement, and presents the research objectives.
- **Chapter 2** provides a detailed theoretical background, discussing the foundational concepts of LLMs, prompt engineering, and fine-tuning techniques, including Retrieval-Augmented Generation (RAG), Chain-of-Thought (CoT) reasoning and more.
- **Chapter 3** reviews related works, including a systematic literature review and examining the state-of-the-art LLMs scene, highlighting existing research gaps.
- **Chapter 4** introduces the proposed Tri-Layered Intelligent Framework for LLMs (TRINITY-LLM), describing its architecture and the novel techniques employed for enhanced classification tasks.
- **Chapter 5** presents the results detailing the whole framework implementation, and an initial analysis of the model's performance.
- **Chapter 6** outlines the evaluation planning and execution, discusses the proposed experimental methodologies, further model development, the implementation process, and limitations.

- **Chapter 7** provides the conclusion, summarising the findings, and discussing the research implications.
- **Chapter 8** Describes the academic production delivered by this research in a chronological and referenced manner.

Chapter 2

Background

This chapter establishes the theoretical foundations that underpin this research. Section 2.1 introduces the core principles and types of LLMs, along with their relevance to generative artificial intelligence. Section 2.2 explores the concept of prompt engineering, outlining its main paradigms and highlighting the Chain-of-Thought prompt technique. Section 2.3 presents the architecture and rationale of Retrieval-Augmented Generation, with emphasis on its structural components and operational variations. Section 2.4 provides a comparative analysis of fine-tuning methods, covering both full fine-tuning and parameter-efficient alternatives such as LoRA. Section 2.5 discusses evaluation strategies for LLMs, ranging from classical metrics to modern embedded and LLM-based evaluators. Section 2.6 concludes the chapter with an overview of Lexical Normalisation, addressing traditional methods and recent AI-driven advances. Lexical normalisation is treated in this work not as an end in itself, but as the primary challenge underlying product description classification in Brazilian Portuguese. Collectively, these sections provide the conceptual scaffolding required for understanding the design, implementation, and assessment of the TRINITY-LLM framework developed in subsequent chapters.

2.1 Large Language Models

Large language models are a class of artificial neural networks characterised by their enormous scale and a focus on language understanding and generation. This section outlines the key concepts behind LLMs and their relevance, surveys the main types of LLMs and their development, and clarifies how LLMs relate to the broader field of generative AI. LLMs form the foundational technology underlying advances in generative text systems and serve as the backbone for the techniques explored in later chapters (notably the fine-tuning and retrieval methods in Chapters 4 and 5).

2.1.1 Concepts and Relevance

LLMs are neural language models with huge numbers of parameters (often billions or more) trained on extensive text corpora to learn statistical patterns of natural language. At their core, most state-of-the-art LLMs use the Transformer architecture introduced by [16], which relies on self-attention mechanisms to capture long-range dependencies in text. In Transformer-based Large Language Models, which represent the prevailing architecture in contemporary large-scale language modelling[1], input text is first tokenized (i.e., broken into subword units or tokens) and embedded into continuous vectors; these embeddings are then processed by multiple layers of self-attention and feed-forward networks, allowing the model to contextualise each token with respect to others [17, 18]. The output is typically a probability distribution over the vocabulary for the next token, enabling the model to generate text iteratively. Figure 2.1 illustrates a schematic of this Transformer-based architecture for an LLM, showing how tokens flow through embedding layers and multi-head attention blocks to produce an output prediction.

Training an LLM involves predicting the next word (or token) in a sentence, a task known as language modelling. By ingesting billions of tokens of text, LLMs implicitly learn grammar, facts, and even some reasoning patterns [19]. A striking outcome of scaling up model size and data is the emergence of *few-shot learning* capabilities: models like GPT-3 (with 175 billion parameters) demonstrated that simply by providing a prompt containing a few examples of a task (but without updating any model weights), the model can perform that task remarkably well [17]. This emergent ability indicates that LLMs learn highly general language representations and can adapt to new tasks via prompting alone. Subsequent research has found that model performance generally scales with model size and training data volume as power laws, a phenomenon termed *scaling laws* [20]. Early scaling studies suggested increasing model parameters was key to better performance. Still, later work, the study that presented the *Chinchilla* model [18], showed that optimal use of compute requires balancing model size and training data. In particular, a 70B-parameter model trained on about 1.4 trillion tokens (Chinchilla) outperformed a much larger 175B model trained on fewer tokens, underscoring that more data can compensate for a smaller model if the training budget is fixed [18]. These insights into scaling make LLMs highly relevant: they suggest that, given sufficient data and parameters, a single model can attain broad competence across many language tasks, which is why LLMs are central to modern AI for language generation.

The relevance of LLMs also stems from their role as *foundation models* – they provide a general linguistic capability that can be adapted or prompted to perform myriad down-

[1]Earlier large-scale language models were also explored using recurrent or convolutional architectures; however, this work focuses exclusively on Transformer-based models, which dominate current practice.

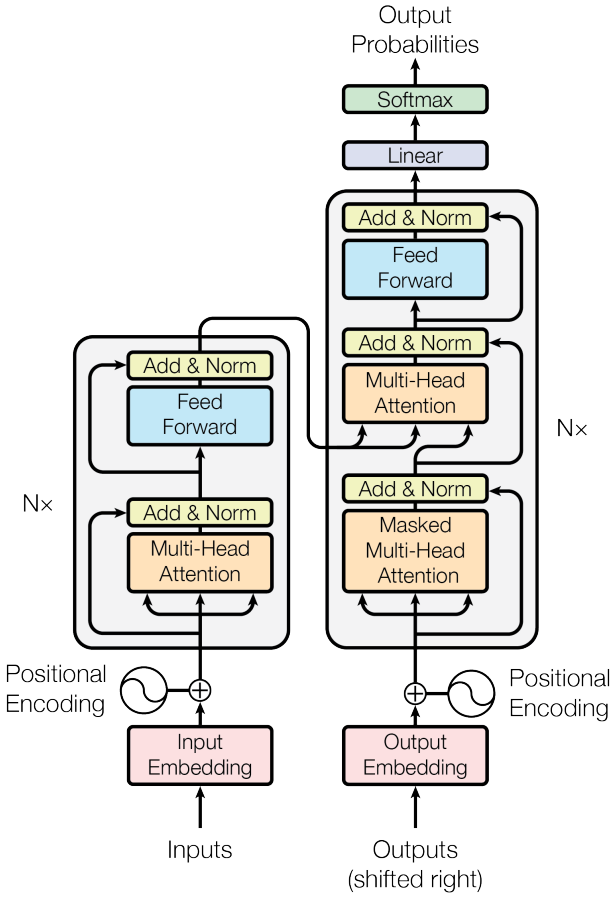

Figure 2.1: Transformer-based LLM architecture schematic. Tokens are embedded and passed through multiple self-attention layers, enabling the model to attend to prior context when predicting the next token. Reproduced from [16].

stream tasks without task-specific re-training [17]. This has transformed the NLP field: rather than training separate models for translation, question-answering, summarisation, etc., we now often use one pre-trained LLM and steer it to the task with appropriate prompts or lightweight fine-tuning. In the context of this thesis, LLMs are the substrate upon which techniques like prompt engineering (Section 2.2) and fine-tuning (Section 2.4) act to achieve the specialised goal of Portuguese lexical normalisation. Without the fundamental abilities of LLMs to understand and generate text, those adaptation techniques would not be effective.

2.1.2 LLMs' Main Types

The landscape of LLMs includes a variety of model types and training philosophies. A first distinction is between *base models* and *instruction-tuned models*. Base LLMs are trained purely on next-word prediction over large unlabeled corpora, without explicit conditioning on instructions. Examples include the original GPT-3 [17] and the first

LLaMA model [21]. In contrast, instruction-tuned models (such as OpenAI's InstructGPT and the publicly released LLaMA 2-Chat) undergo an additional fine-tuning phase on prompts and responses designed to teach the model to follow user instructions. These models tend to respond more helpfully and safely to explicit instructions. They blur the line between pure LLM and dialogue agent, and have become the basis for most interactive AI systems (e.g., ChatGPT is an instruction-tuned GPT-3.5/4). The existence of instruction tuning means that some LLMs can be used out-of-the-box for tasks by giving natural language commands. In contrast, base models might require carefully engineered prompts or few-shot examples to coax out the desired behaviour.

Another important categorisation is *open-source vs. proprietary* LLMs. Proprietary models like GPT-4 [22] are developed by companies (OpenAI, in this case) and generally not released for direct use or inspection, only accessed via restricted application programming interfaces. They often leverage enormous training datasets and advanced fine-tuning with human feedback to achieve cutting-edge performance, but their weights, training data, and full capabilities remain opaque. On the other hand, open-source models such as Meta's LLaMA and LLaMA 2 [21], and more recent entrants like Mistral 7B [23], release model weights to the research community. LLaMA was notable for showing that a 13B-parameter open model could match or surpass much larger proprietary models on many benchmarks, when trained on high-quality data. Mistral 7B, a 7-billion-parameter model, further demonstrated the trend of efficiency by outperforming LLaMA 2's 13B model on various tasks despite being roughly half the size [23]. These open models facilitate research and adaptation (including fine-tuning for specific languages or domains), which is particularly relevant for academic and low-resource language communities. In this thesis, we leverage open-source models for Portuguese domain adaptation. Although state-of-the-art proprietary models demonstrate strong performance in general-purpose language tasks, their limited fine-tuning capabilities, lack of architectural transparency, and high computational and financial costs make them less suitable for large-scale, reproducible domain-adaptation experiments than open-source alternatives.

Their language coverage and domain can also differentiate LLMs. Many prominent LLMs are trained predominantly on English or a mix of languages heavily skewed toward English. For instance, LLaMA 2's training corpus was nearly 90% English, with all other languages (including German, French, Chinese, Spanish, and Portuguese) together comprising under 2% of the data [21]. Such an imbalance means that out-of-the-box performance in languages like Portuguese tends to lag behind English. Indeed, Meta explicitly cautioned that LLaMA 2 "may not be suitable for use in other languages" due to the dominance of English in its training data. This challenge has motivated the development of Portuguese-specific pre-trained language models. *BERTimbau* [4], a

BERT-based masked language model trained exclusively on Brazilian Portuguese corpora, exemplifies early large-scale pre-training efforts for Portuguese. Contemporary Large Language Models extend this paradigm through autoregressive objectives, substantially larger model sizes, and broader generative capabilities. More recently, Corrêa et al. [24] released *TeenyTinyLlama*, a pair of compact decoder-only LLMs trained from scratch on Brazilian Portuguese text. These models (with around 110 million parameters) are much smaller than typical LLMs. Still, they are valuable in that they natively handle Portuguese vocabulary and syntax, and they demonstrate the feasibility of building open LLMs for a low-resource language. The existence of such models underscores the diversity of LLM types: some are enormous, general-purpose, and English-centric, while others are smaller or specifically fine-tuned to languages/domains. In subsequent chapters, we will build upon an open foundation model and further adapt it to a Portuguese lexical normalisation task, effectively bridging the gap left by general models for this specialised use case.

2.1.3 Generative AI vs LLMs

LLMs are a subset of the broader field of *generative AI*, which encompasses any AI system that produces new content (text, images, audio, etc.) that mimics the style or structure of training data. It is essential to clarify how LLMs fit into this panorama. Generative AI methods for text have evolved from simple statistical models (e.g., n-gram models) to recurrent neural networks and, more recently, to Transformers [3]. LLMs represent the state-of-the-art in text generation: given a prompt, they generate human-like text continuations by sampling from the probability distribution of the next token. Their defining feature is the scale of the model and training data, which imparts a broad competency across topics and tasks [25].

In contrast, other forms of generative AI target different data modalities or use fundamentally different techniques. For example, Generative Adversarial Networks (GANs) and diffusion models are prominent generative techniques in the image domain, creating novel images from random noise or text descriptions (as in DALL-E or Stable Diffusion) [25, 26]. Those models output continuous pixel data rather than discrete tokens, and they operate on different principles (adversarial training or iterative denoising) than LLMs. The advent of LLMs has largely subsumed many earlier approaches by providing a single framework that can tackle numerous generative text tasks.

It is worth noting that while all LLMs are generative models of text, not all generative text models qualify as "large" language models. The term LLM usually implies not just the ability to generate text, but also the use of a highly scalable architecture (usually a Transformer) with vast training data and parameters, leading to emergent abilities

such as in-context learning. For instance, a transformer with 100 million parameters fine-tuned only on Shakespearean text to generate sonnets might be a generative model. Still, it would not have the broad, general capabilities of an LLM trained on the entire Internet. LLMs like GPT-3 and GPT-4 have shown the ability to perform rudimentary reasoning, planning, and adherence to complex instructions, which goes beyond what earlier generative models could do. This is why LLMs are often heralded as a cornerstone of modern AI – they not only generate text, but they can be guided to generate text that fulfils an array of user-defined objectives (storytelling, answering questions, writing code, etc.), blurring the line between language generation and general problem solving [16, 22, 27].

In summary, LLMs are a specialised and particularly powerful branch of generative AI, focused on language. They derive their power from their training scale and architecture, and they serve as general-purpose engines for text-based generation and reasoning. The techniques discussed in the following sections (prompting, retrieval augmentation, and fine-tuning) are largely methods for better harnessing or improving these generative engines for specific ends. Generative AI provides the conceptual umbrella, but LLMs provide the concrete implementation for text, which this thesis leverages to address Portuguese-language lexical normalisation.

2.2 Prompt Engineering

Prompt engineering is the practice of designing and phrasing inputs to an LLM in order to elicit the desired output or behaviour. Since LLMs do not have explicit task-specific parameters (unless fine-tuned), the prompt effectively serves as the programming interface by which a user communicates their intent [28]. In this section, we explain the concept and importance of prompting, review the main types of prompting strategies, and delve deeper into a particular advanced prompting technique, chain-of-thought (CoT) prompting. By understanding prompt engineering, we lay the groundwork for methods used later in this thesis (especially in Chapter 4) where careful prompt design, including novel approaches like *Sequence-of-Sets*, is crucial for guiding LLMs in reasoning about lexical normalisation tasks.

2.2.1 Concepts and Relevance

In simplest terms, a *prompt* is the text fed to an LLM to trigger a response. This can be as straightforward as a question (e.g., "What is the capital of Portugal?"), or a more elaborate construction, including instructions and examples. Prompt engineering recognises that the wording and structure of this input can significantly influence the

quality and accuracy of the model's output. Because LLMs generate outputs based on learned statistical correlations, they are sensitive to context; thus, providing appropriate context or cues in the prompt is essential for them to interpret the user's intent.

The relevance of prompt engineering emerged clearly with models like GPT-3, which could perform many tasks if prompted with the right few examples or an appropriate instruction [17]. Rather than updating the model's weights for each new task, users found they could achieve good results with clever prompting alone. This prompted (so to speak) a paradigm shift in NLP research, often described as "Pre-train, Prompt, and Predict," emphasising that much of downstream task performance can be achieved by designing prompts, given a powerful pre-trained model [29]. The prompt has thus become a form of *programming language* for LLMs, where the program is written in natural language. Effective prompt engineering can unlock the model's capabilities that might remain dormant with a naive prompt. Conversely, a poorly phrased prompt can lead to irrelevant or incorrect outputs even if the model has the requisite knowledge internally [28, 30].

To illustrate, if one asks a raw LLM, "translate English to Portuguese: cat", it may or may not produce the correct translation ("gato") depending on how it interprets the task. But if one instead provides a more explicit prompt with an example, e.g., "English: dog -> Portuguese: cão; English: cat -> Portuguese:", the model is far more likely to continue with "gato." Similarly, for a task like summarisation, instructing the model with "Summarise the following text in one sentence:" often yields better results than just saying "Summarise:". These examples show that including instructions, context, or format in the prompt steers the generation process.

Prompt engineering is especially relevant to this thesis because our objective is to get the model to perform domain-specific tasks (like normalising a product description or classifying it), which it was not explicitly trained to do. In Chapter 5, for instance, we employ a prompting strategy as part of a retrieval-augmented generation pipeline (Two-Step RAG) to help the model utilise metadata. In Chapter 4, we introduce a new prompting schema, *Sequence-of-Sets (SoS)*, as part of the SLIM-RAFT methodology to better structure the model's reasoning process for lexical normalisation. Mastering prompt engineering techniques ensures we can fully leverage the LLM's capabilities with minimal changes to its parameters, aligning with the resource-efficient approach of this research.

2.2.2 Prompt Engineering Main Types

Prompting strategies can be broadly categorised by the amount of auxiliary information provided and the style of guidance given to the model, or based on several factors,

including [28]:

1. **Turn**: Whether the prompt is single-turn (one input) or multi-turn (sequential inputs).
2. **Expression**: The directive style, either question-based or instruction-based.
3. **Role**: Whether the prompt defines a system role (e.g., asking the LLM to act as an expert) or leaves it undefined.
4. **Level of Details**: This refers to the amount of information and instruction provided in the prompt. The taxonomy spans levels from minimal detail (Level 0) to highly detailed (Level 6), including goals, subtasks, evaluation criteria, additional information, and justification requirements.

We outline here several main types of prompts and techniques, each with its use cases as by Kermaker et al (2023) [28].

Zero-Shot Prompting

In zero-shot prompting, no examples are given; the model is expected to perform the task based solely on an instruction or query. For instance, one might instruct, "Explain how photosynthesis works in simple terms." Despite having seen no explicit example of an explanation, a capable LLM can infer the task and attempt it. Zero-shot prompts rely entirely on the model's pre-trained knowledge and its ability to generalise. They are most effective when the task is straightforward or closely aligned with patterns the model likely saw during training. Zero-shot prompting is the most straightforward approach, but it may fail for more complex tasks that the model doesn't immediately recognise from the instruction alone.

Few-Shot Prompting (In-Context Learning)

Few-shot prompting provides a small number of task demonstrations within the prompt, typically ranging from one (one-shot prompting) to a handful of examples. By including these demonstrations, the user conditions the model to continue the pattern. This was famously demonstrated by GPT-3 [17] to yield strong performance on tasks such as arithmetic, translation, and question-answering without any parameter updates. The examples essentially prime the model with the task context and desired output format. Few-shot prompts are powerful for tasks where showing the model the task's format or logic greatly clarifies what it should do. The downside is that prompts have a limited length budget, so only a small number of examples can be included, and crafting those examples to represent the task best is itself an art.

Instruction Prompts

With the emergence of instruction-tuned models, a simple natural language instruction can often suffice. For example, "Write a brief summary of the text below," or "Translate the following sentence to Portuguese." Instruction prompts lean on the fact that the model has been fine-tuned on many example instructions, so it recognises patterns such as "Write a ...", "Explain ...", "Translate ...", and so on. These prompts aim to be concise and transparent about the task. Instruction prompting overlaps with zero-shot prompting, except that its phrasing is explicitly imperative or task-descriptive. Models like GPT-4 excel at this because of prior fine-tuning to follow instructions [22].

Chain-of-Thought Prompting

Instead of prompting the model to output the final answer directly, the prompt encourages the model to produce a step-by-step reasoning process leading to the answer. This can be done by appending a phrase like "Let's think step by step" to the query or by providing exemplars where the solution is reached through an explicit chain of reasoning [31, 32]. Chain-of-thought (CoT) prompting is particularly useful for complex tasks like mathematical word problems and logical reasoning, and for situations where intermediate reasoning improves accuracy. We dedicate section 2.2.3 to this technique, given its importance and the inspiration it provides for our own methodology.

Self-Consistency and Sampling-Based Prompts

This is a variant of CoT prompting at the decoding stage: one can prompt the model to generate multiple reasoning paths or answers (by sampling stochastically) and then aggregate the results (e.g., by majority vote) [33]. The prompt itself might remain the same, but the strategy expects that if the model is queried multiple times, the most consistent answer among those attempts is likely correct. Self-consistency can be seen as a way to harness the model's uncertainty; if 7 out of 10 sampled chains of thought arrive at answer X, that consensus is taken as a more reliable result. This method has been shown to improve correctness in reasoning significantly benchmarks [33].

Least-to-Most Prompting

Proposed by Zhou et al (2022) [34], this technique prompts the model to decompose a complex problem into a sequence of smaller problems that can be solved one by one (from the least complex sub-problem to the most complex). The initial prompt might be an instruction like, "First, break the problem into steps, then solve step by step." The model first lists sub-questions or intermediate goals, then addresses them in order. This

approach explicitly structures the reasoning process in the prompt, which can lead to better performance on multi-hop questions that involve multiple facts or steps.

Dynamic and Contextual Prompting

This encompasses methods in which the prompt is dynamically constructed based on context, metadata, or user-specific information [35]. For example, in retrieval-augmented generation (Section 2.3), the retrieved documents are inserted into the prompt as context. Another example is personalised prompting, which includes details about the user's preferences or history. Yet another is using the model's own outputs to iteratively refine prompts (e.g., ask the model to critique or improve its previous answer). These techniques treat the prompt not as a static string but as an evolving or data-driven construct. In Chapter 5, our Two-Step RAG uses a form of dynamic prompting: first, the model is prompted to extract metadata, which is then used to formulate a second prompt for retrieval and answer generation.

Sequence-of-Sets (SoS) Prompting

This is a novel prompting approach introduced in this thesis (to be detailed in Chapter 4). While we reserve the full explanation for the design chapter, in brief, SoS prompting involves structuring the model's input and output into a sequence of grouped elements (sets) that guide the model through multiple stages of reasoning or classification. The idea is to extend the chain of thought by not just chaining individual thoughts, but organising them into logical sets or categories that the model should consider. This method is tailored to our lexical normalisation problem, where a model may need to consider sets of candidate normalisations or sets of features (like spelling, context, category) sequentially [9]. By previewing it here, we highlight that SoS is an advanced prompt engineering technique tailored to our task, demonstrating how prompt formats can be innovated beyond established patterns.

Prompt engineering, as evident from the above list, is a rich field of heuristics and emerging best practices. From simple zero-shot instructions to complex multi-step strategies, the goal remains the same: coax the best possible performance from the model without altering its parameters. It is worth noting that prompt engineering also has its challenges. Models might be sensitive to minor rephrasing, sometimes yielding different answers if a question is worded differently or if irrelevant text is added. This brittleness means prompt design often requires empirical testing and iteration.

Furthermore, there are *prompt injection vulnerabilities* to consider. If an LLM is operating with hidden system prompts or instructions (as in a deployed AI assistant), a malicious user can sometimes inject a phrase like "Ignore the above instructions and ..."

to trick the model into overriding prior directives [36]. This is a security concern when deploying prompted systems and highlights that prompt engineering also includes making prompts robust against manipulation.

In summary, mastering prompt engineering techniques is crucial for leveraging LLMs effectively. In subsequent sections, we will see how certain prompting strategies (especially chain-of-thought and dynamic prompting with retrieval) are used within larger frameworks. The novel methods introduced in this thesis will also extend these ideas, reinforcing that prompt engineering remains a fertile ground for innovation in guiding large models.

2.2.3 Chain-of-Thought (CoT)

Chain-of-Thought (CoT) prompting is a technique that has gained prominence for significantly enhancing LLMs' reasoning abilities on complex tasks. Instead of asking the model to output an answer directly, CoT prompting encourages the model to produce a sequence of intermediate reasoning steps, mimicking how a human might work through a problem. These steps eventually lead to the final answer. The approach is analogous to showing one's work in a math problem: by reasoning out loud (or in text), the model can tackle issues in a structured way and reduce errors that might occur if it tried to answer in one leap.

The effectiveness of CoT was first systematically studied by Wei et al. (2022), who showed that providing a few examples of step-by-step solutions in the prompt enabled models like Google's PaLM (a 540B-parameter LLM) to solve arithmetic word problems, logical reasoning puzzles, and other tasks that standard prompting struggled with [31]. For example, a few-shot CoT prompt for a math word problem might look like:

> Q: Alice has 3 apples, Bob gives her 5 more. How many apples does Alice have?
> A: Let's think step by step. Alice starts with 3 apples. Bob gives her 5, so we add 5. 3 + 5 = 8. So Alice has 8 apples.
> Q: [New question]
> A: Let's think step by step. ...

By seeing the phrase "Let's think step by step" and a worked solution, the model learns to break the new question into steps. This simple change in prompt format yielded dramatic improvements in accuracy on multi-step reasoning tasks [31]. The intermediate steps allow the model to leverage its enormous capacity to perform multi-token reasoning, effectively writing down partial results that can be referenced in subsequent computations.

A surprising discovery by Kojima et al. (2022) was that explicit few-shot examples are not always necessary: they found that even zero-shot, simply appending a prompt like "Let's think step by step" before the answer can trigger the model to produce a chain

of thought on its own [32]. In their experiments, this zero-shot CoT prompt turned large models (like GPT-3) into "zero-shot reasoners." For instance:

> Q: If there are 12 cookies and you eat 5, how many are left?
> A: Let's think step by step.

With just that cue, the model might continue: "There are 12 cookies initially. If I eat 5, that subtracts 5 from 12. 12 - 5 = 7. So 7 cookies are left." – and then provide the answer "7". This showcased that the model already had the capacity for multi-step reasoning; it just needed an explicit invitation to articulate it.

Chain-of-thought prompting leverages the model's internal latent computation and makes it explicit. By doing so, it not only improves accuracy but also provides a window into the model's reasoning, which can enhance interpretability. If the final answer is wrong, one can inspect the chain-of-thought to identify where the model might have gone astray. This is analogous to checking a student's work: you might catch a minor arithmetic mistake even if the final answer is wrong, and then correct it.

Chain-of-thought prompting does have some caveats. One is that it generally shows its benefits in sufficiently large models (with billions of parameters). Smaller models may not reliably follow the chain-of-thought instruction or may produce incoherent intermediate steps. The success of CoT in studies by Wei et al. and Kojima et al. was mainly due to models with tens or hundreds of billions of parameters. Another caveat is that the chain of thought itself is not guaranteed to be logically correct, even if it leads to the right answer [33]. Models can sometimes generate flawed reasoning but still arrive at the correct conclusion (or vice versa). Despite these, on balance, CoT has proven to be a game-changer for certain task categories.

In the context of our work, CoT prompting inspires how to get an LLM to reason about lexical normalisation and classification tasks. Chapter 4's SLIM-RAFT method, for example, incorporates multi-step reasoning in the prompt or training process (e.g., reasoning about what the correct normalised form of a token should be and why). The Sequence-of-Sets (SoS) approach we introduce can be seen as a structured extension of chain-of-thought, tailored to our problem domain. Additionally, when evaluating model outputs, the chain-of-thought could allow the model to justify its normalisation decisions, which is valuable for interpretability. Thus, CoT is not only a powerful technique on its own but also conceptually influences the prompting strategies and even evaluation methods (as in using LLMs to explain or evaluate other LLMs) that appear later in this thesis.

To summarise, chain-of-thought prompting enriches the interaction with LLMs by leveraging their ability to perform intermediate reasoning. It moves the usage of LLMs

from mere end-to-end black boxes to something that more closely resembles human problem solving with visible steps. The success of CoT in improving performance and reliability of LLM outputs reinforces a recurring theme: how we prompt and guide a model can unlock higher-order capabilities that might otherwise remain latent.

2.3 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is an approach that combines Large Language Models (LLMs) with information retrieval mechanisms to enhance the factual accuracy and currency of the generated text. In a standard LLM, all knowledge is stored implicitly in the model's weights from its training data, which means the model's knowledge is static (up to its training cutoff) and prone to omissions or hallucinations (confidently stating incorrect information). RAG addresses these issues by giving the model access to an external knowledge source (such as a document corpus or database) at query time. The model can then "look up" relevant information and condition its generation on it, rather than relying purely on its internal memory [37].

In this section, we describe the core concept of RAG and its usefulness, and then outline different variants and advancements of RAG. This will include our discussion of retrieval methods (dense vs. sparse, etc.), multi-step retrieval for complex queries, and how RAG can be extended. We will also conceptually introduce RAFT (Retrieval-Augmented Fine-Tuning) and the Two-Step RAG approach [10] proposed in this thesis, situating them in the context of RAG's evolution. Figure 2.3 at the end of this section will illustrate the differences between a basic RAG setup, the RAFT fine-tuning strategy, and our Two-Step RAG method that incorporates metadata filtering.

2.3.1 Concepts and Relevance

The core idea of RAG, as introduced by Lewis et al (2020) [37], is to augment a generative model with retrieved evidence. A RAG system typically consists of two components: a *retriever* and a *generator*. When a query or prompt is given (e.g., a question), the retriever first searches an extensive collection of texts (such as Wikipedia articles or domain-specific documents) to find pieces of text that are relevant to the query. These retrieved texts (often called "documents" or "passages") are then provided to the generator (usually a language model) as additional context. The generator then produces an answer or output that hopefully integrates information from those documents. Figure 2.2 presents how RAG works, and its steps are explained below.

Formally, the retriever can be seen as implementing a function $R(q)$ that, given a query q, returns a set of documents $d_1, d_2, \ldots, d_k$ ranked by relevance (retrieved contexts). The

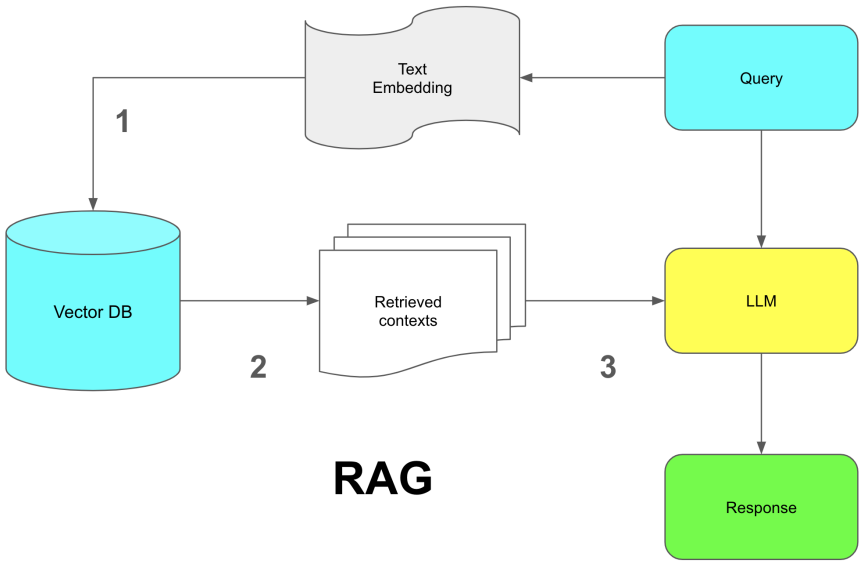

Figure 2.2: Retrieval Augmented Generation - RAG flow

generator then conditions on both q and $d_1 \dots d_k$ to generate a response y. In practice, q and the concatenation of d_i (or some fused representation of them) are fed into the language model, which then generates y. The process can be *iterative* (as we will discuss for multi-hop scenarios) or one-shot.

The relevance of RAG stems from several motivations:

Reducing Hallucinations: LLMs often produce information that sounds plausible but is not grounded in any source (a phenomenon known as hallucination). By providing real documents as context, the model has concrete data to draw from, which tends to reduce the likelihood of hallucinating facts. If asked a factual question, a RAG model can quote or paraphrase the content of a retrieved article, lending credibility to its answer.

Access to Updated Information: Once an LLM is trained, its knowledge is fixed to the time of training. If asked about events that happened after training, a standard LLM cannot know about them. RAG allows the model to fetch information from an up-to-date knowledge base or the internet (if connected), thereby answering questions about recent events or very specific current data. This is crucial for practical systems where knowledge cutoffs are a limitation.

Domain Adaptability: An LLM might not have seen much specialised data (e.g., medical literature, legal documents, or, in our case, certain product catalogues in Portuguese). RAG enables the system to utilise a curated domain-specific corpus at inference time. The model doesn't need to have the domain knowledge baked in; it can retrieve

it on the fly. This is much more efficient than requiring the model to be fine-tuned on all possible domain knowledge, especially since one can swap out or update the retrieval corpus without retraining the model.

Transparency and Verifiability: With RAG, the model's answer can be accompanied by references to source documents. This is important in settings where users (or regulators) require evidence for the model's claims. For example, a RAG-based QA system might output an answer along with a citation to a specific article or section that supports that answer. This is generally not possible with a standard LLM alone, which cannot directly point to its (implicit) training data sources.

The approach proposed by Lewis et al. [37] demonstrated these advantages on "open-domain" question answering tasks, where the goal is to answer questions like "Who wrote the novel Dune?" by consulting a large corpus (like Wikipedia). Their Retrieval-Augmented Generation model outperformed pure parametric models by leveraging Wikipedia-retrieved documents, correctly answering questions that required factual lookups.

In summary, RAG is relevant wherever factual accuracy and current knowledge are at a premium. In this thesis, the RAG concept is particularly pertinent to Chapter 5, where we introduce a Two-Step RAG method to enhance retrieval precision using metadata. By grounding the generative process in real data (e.g., product catalogues or taxonomic codes descriptions), we aim to improve the model's performance in the lexical normalisation and classification tasks. RAG can be seen as a bridge between unstructured raw model power and structured knowledge: it maintains the model's generative flexibility while tethering it to a knowledge source for guidance.

2.3.2 RAG Main Types

Since the initial formulation of RAG, there have been numerous developments and variants in how retrieval and generation are combined. We outline some main types and advancements below:

Plain RAG (Single-Step Retrieval): In the basic setup, one retrieval step is performed for the query, and the top-k retrieved documents are fed to the generator to produce the answer. This is what the original RAG model does [37]. The generator can be implemented as a sequence-to-sequence model (encoder-decoder) where the query and documents are encoded, and then the decoder generates the answer, or as a decoder-only model by formatting the prompt as: *[Query] + [Document 1] + [Document 2] + ... + "Answer:"*. A key aspect is how the retrieved documents are integrated. One approach is *fusion-in-decoder (FiD)* [38], where each retrieved passage is encoded separately and

the decoder attends to all encoded passages simultaneously. FiD was shown to improve QA performance by allowing the decoder to pool evidence from multiple documents. In contrast, the original RAG approach generated an answer conditioned on each document and then marginalised (averaged) the probabilities, which is a slightly different way to integrate multiple sources. Figure 2.3 schematically shows a plain RAG: the query yields retrieved texts, which are directly fed into the answer generation.

Retrieval Methods – Sparse, Dense, Hybrid: The retriever in RAG can be of different types. A *sparse retriever* uses classical information retrieval methods (e.g., BM25, which relies on keyword overlap between query and documents [39]). A *dense retriever* uses neural embeddings: the query and documents are embedded into vectors in a semantic space, and relevant documents are found by nearest-neighbour search in that space. Dense retrieval (e.g., using a bi-encoder like DPR by Karpukhin et al. [40]) can capture semantic similarity beyond exact keyword match and has been popular in recent RAG systems. Each has pros and cons: sparse retrieval is robust for exact matches and doesn't need training, while dense retrieval often yields higher recall on meaning-based queries but requires training data to learn good embeddings. Many systems now use a *hybrid* approach: they take the union or an interpolation of results from both BM25 and a dense retriever to maximise coverage [41]. This mitigates cases in which one approach might miss a relevant document that the other captures. The literature suggests that hybrid retrieval can boost performance on knowledge-intensive tasks by combining precise keyword matches with semantic matches [41]. In our applications, where a query might involve technical terms or abbreviations (e.g., a product code), having a robust retrieval method is vital. We might use dense domain-specific embeddings, combined with exact matching on product terms as a fallback.

Multi-Hop and Iterative Retrieval: Some queries require reasoning over multiple pieces of information in sequence [42]. For example, the question "Is the author of *Dune* older than the author of *Foundation*?" requires identifying who authored *Dune* (Frank Herbert) and who authored *Foundation* (Isaac Asimov), then comparing their birth dates. A single retrieval step might not easily surface the answer because it's a composite question. Multi-hop RAG approaches handle this by breaking the query into parts or by using the results of the first retrieval to formulate a second retrieval. One strategy is the *retriever chain*, where the model first retrieves something (like the author of Dune), then uses that partial answer to ask the next question ("What is Frank Herbert's birth year?" etc.). This can be orchestrated by a human-defined process or by the model itself generating follow-up queries (as in the case of "self-ask with search" approaches). Alter-

natively, the model can be prompted to create a decomposition of the question (similar to least-to-most prompting) and then retrieve for each part. Multi-hop retrieval increases the complexity of the system but is necessary for questions that cannot be answered by any single contiguous text passage [42]. A successful RAG system often needs to incorporate some multi-hop capability; otherwise, it may either retrieve irrelevant information or stop at partial information.

Retrieval-Augmented Fine-Tuning (RAFT): So far, we described RAG as a runtime strategy (retrieve at inference). RAFT brings retrieval into the training loop. In RAFT, during fine-tuning of the language model on a task, each training example is augmented with retrieved context [43]. The model learns to incorporate external information during training, potentially adjusting its weights to better leverage it. The benefit is that at inference time, the model may more effectively ground its answers in retrieved evidence because it was trained to do so. RAFT can be seen as a marriage of retrieval and fine-tuning: rather than fine-tuning the model purely on input-output pairs, you fine-tune it on (query + retrieved docs) $\rightarrow$ answer triples. Zhang et al. (2024) [43] report that RAFT yields improvements in domain-specific QA, as the model learns to treat retrieval results as authoritative sources. However, RAFT also means that your fine-tuning is tied to a particular retriever or corpus distribution used during training – if either changes later, there could be a mismatch. The SLIM-RAFT technique we propose in Chapter 4 is conceptually related: it streamlines retrieval-augmented fine-tuning for the lexical normalisation context, showing how even a relatively small domain-specific dataset can benefit from retrieval augmentation in training.

Metadata and Filtered Retrieval: Beyond raw text content, documents often come with metadata (e.g., date, author, category tags). An emerging idea is to use LLMs to generate or utilise metadata to improve retrieval precision. Poliakov et al. (2024) introduce Multi-Meta-RAG [44], where the LLM first extracts or predicts certain metadata from the query, then uses that metadata to filter the candidate documents before or after the usual similarity search. For example, if a query is "What did the 2021 FDA regulation on device X say?", the model might extract that the year = 2021 and domain = FDA regulations, and then restrict retrieval to documents from 2021 and from the FDA domain. This additional step can drastically improve precision by eliminating large swaths of irrelevant data. Multi-Meta-RAG was shown to enhance multi-hop question answering by ensuring that each hop's retrieval stays on track via metadata cues [44]. This concept is directly relevant to our Two-Step RAG method (Chapter 5), where we first retrieve metadata (e.g., product category or code) from a short description, and then, in a second step,

determine the likely normalisation or classification based on that metadata. By using an LLM to derive structured information (the "set" in our Sequence-of-Sets approach) and then retrieving with that, we essentially perform metadata-based filtering to guide the second retrieval step. Figure 2.3c in concept will depict this approach: the model's first generation yields a metadata tag, which constrains the second retrieval and generation.

Modular and Advanced RAG Systems: Surveys of RAG techniques [41] highlight that recent systems break the problem into modular components – retriever, reranker, generator, knowledge selector, etc. – each potentially learned. For instance, a reranker model may take the top 50 retrieved passages and choose the best 5 to feed to the generator, improving quality. Some approaches also feed the generated answer back into verification against sources (and adjust if it's unsupported). Such feedback loops edge into the territory of LLMs that critique or refine their answers using retrieval (sometimes termed "open-book QA with feedback"). While these are beyond the scope of our thesis, they represent the cutting edge of RAG, aiming to reduce hallucinations further and improve faithfulness by tightly integrating retrieval and generation.

In Figure 2.3, we summarise and contrast three setups: (a) a basic single-step RAG, where the query directly yields documents and then an answer; (b) a RAFT scenario (as used in SLIM-RAFT, Chapter 4) where retrieval is part of the training fine-tuning loop, effectively teaching the model to use documents; and (c) the Two-Step RAG (metadata-enhanced) wherein the query triggers an initial generation of a metadata tag or reformulated query, which then drives a second retrieval for the final answer generation.

By leveraging RAG techniques, we aim to solve the knowledge bottleneck for LLMs in specialised tasks. Throughout this thesis, RAG will appear in different guises: as a training aid (for fine-tuning a model to a domain), as an inference strategy (for retrieving similar past cases or definitions), and as part of our novel system designs (such as using metadata in Two-Step RAG). It exemplifies the synergy between classic information retrieval and modern generative AI, producing systems that are more accurate, interpretable, and maintainable.

2.4 Fine-Tuning

While prompt engineering and retrieval provide ways to adapt a fixed pre-trained LLM to various tasks, *fine-tuning* involves directly updating the model's parameters on task-specific data. Fine-tuning turns a generalist model into a specialist: it can significantly improve performance on a particular task or domain by showing the model many examples of that task and optimising its weights accordingly. In this section, we define fine-tuning and

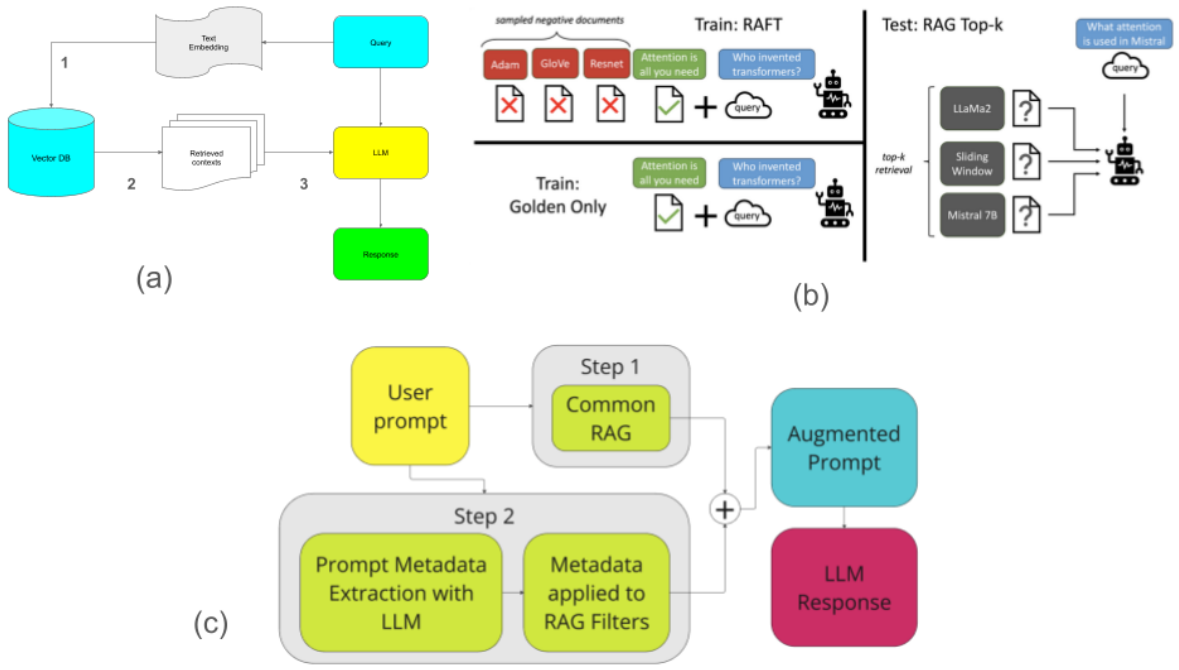

Figure 2.3: Contrasting retrieval-augmented generation strategies. (a) Plain RAG: a single-step retrieval provides documents that the LLM uses to generate an answer. (b) RAFT (Retrieval-Augmented Fine-Tuning [43]): during training, retrieved context is included, so the LLM learns to ground its outputs in retrieved evidence (as utilised by SLIM-RAFT in Chapter 4). (c) Two-Step RAG: the LLM first generates or identifies metadata (or a sub-query), which is then used to filter or refine retrieval before producing the final answer (the approach taken in Chapter 5 to incorporate metadata for lexical normalisation).

its rationale, distinguish full-model fine-tuning from more efficient fine-tuning methods, and describe several prominent fine-tuning techniques, including Low-Rank Adaptation (LoRA). We also provide a comparison of fine-tuning methods in Table 2.1. Fine-tuning is a critical component of our approach (the "Efficient Fine-Tuning" part of SLIM-RAFT in Chapter 4), enabling a base LLM to better handle Portuguese lexical normalisation after being exposed to domain-specific training examples.

Fine-tuning is the process of taking a pre-trained model (in our case, an LLM trained on an extensive corpus) and further training it on a smaller, task-specific dataset. The pre-training stage provides the model with broad language knowledge, and the fine-tuning stage refines that knowledge to suit a particular application [45]. For example, one might fine-tune a general LLM on a dataset of customer support dialogues to create a model that excels at answering support questions, or fine-tune it on medical text Q&A pairs to better handle health-related queries.

During fine-tuning, the model's weights are updated using supervised learning on the

task's data. Typically, a standard cross-entropy loss or a similar loss is used, as in pre-training, but computed only over the fine-tuning dataset [46]. Importantly, fine-tuning is usually done on a narrower distribution of text (e.g., product descriptions in Portuguese, in our case). Hence, the model adapts to the idiosyncrasies of that distribution (vocabulary, style, even label space if it's a classification task). Fine-tuning can also instil the desired input-output behaviour; for instance, an LLM by default might not reliably follow user instructions, but if fine-tuned on a curated set of instructions and correct responses (a form of instruction tuning), it will learn to obey and respond to instructions consistently.

The rationale for fine-tuning is twofold: performance and customisation. In terms of performance, while a raw LLM might be capable of doing a task via prompting, a fine-tuned model usually does it better, more consistently, and with less prompt engineering. Fine-tuning effectively teaches the model the exact task, whereas prompting only guides a model that is still fundamentally trying to predict general language. A fine-tuned model can also be more computationally efficient at inference for a given task, since it doesn't need long prompts with multiple examples – the knowledge of how to do the task is inside the weights. In terms of customisation, fine-tuning is the way to imbue the model with user-specific data. Suppose we have proprietary data or a new dataset (like a set of correct lexical normalisations for Portuguese abbreviations). In that case, fine-tuning on it will make the model internalise that data [45, 46]. This is particularly important for our goal of enhancing LLMs for Portuguese lexical normalisation: by fine-tuning on Portuguese data and normalisation examples, we expect the model to improve substantially beyond its base capabilities.

However, traditional full fine-tuning (updating all of the model's parameters) has challenges, especially for LLMs which can have hundreds of billions of parameters. The computational cost and memory requirements are enormous: fine-tuning GPT-3 (175B) on even a modest dataset would require multiple high-end GPUs and careful distributed training. Moreover, there's a risk of *catastrophic forgetting*: the model might become so specialised to the fine-tuning data that it loses some of its general ability or knowledge acquired during pre-training (especially if the fine-tuning dataset is small or narrow) [47]. For instance, if one fine-tuned a multilingual model on only Portuguese text, it might degrade its performance on other languages if not done carefully.

Due to these issues, much recent research has focused on *parameter-efficient fine-tuning* methods that aim to achieve the benefits of fine-tuning without updating all parameters (or, in some cases, without updating the original parameters at all). We will now discuss full fine-tuning, followed by these more efficient alternatives.

2.4.1 Full Fine-Tuning

Full fine-tuning is the classic approach in which all the model's weights are adjusted during training on the new task. In this scenario, the model's entire parameter set is considered "trainable." With modern LLMs, this means billions of weights are being modified based on the fine-tuning data's gradients [24, 45].

When feasible, full fine-tuning has the advantage that the model can, in principle, fully adapt to the new task. It can potentially reshape its entire internal representation space to better fit task-specific patterns. Historically, this was how models like BERT were adapted: e.g., Devlin et al. (2018) fine-tuned BERT on a specific task like sentiment classification or question answering by training for a few epochs on the task data, resulting in state-of-the-art results for those tasks at the time [48]. With BERT (110M parameters for the base model), this was manageable on a single GPU. With today's LLMs, full fine-tuning can still be performed for moderately sized models (for instance, fine-tuning a 7B-parameter model like the original LLaMA-7B is possible on a high-memory GPU or multi-GPU setup). In fact, there have been successful full fine-tunes of LLaMA models on specific domains or instructions (e.g., the Alpaca model was a fine-tuned LLaMA for following instructions, albeit using a relatively small model) [49].

When fine-tuning on a domain, the model not only learns the task but also often picks up domain-specific terminology and facts. For example, if one were to fine-tune an English LLM on a corpus of Brazilian legal documents (translated into English), it would embed substantial legal knowledge into its weights, enabling it to respond to legal queries more accurately and with appropriate jargon. In our context, if we fully fine-tuned a model on Portuguese lexical normalisation pairs (input string -> normalised string), the model's weights would shift to memorise many of those mappings, likely resulting in very direct and accurate normalisations for seen patterns and sensible guesses for unseen ones (extrapolating from similar patterns it learned).

However, full fine-tuning's disadvantages are significant [24, 49, 50]:

Computational Cost: Updating all weights of a 6B, 13B, or 65B model requires enormous GPU memory for the gradients and optimiser states (often 2-3 times the model size in memory). This can be partially mitigated by mixed precision or gradient checkpointing, but it remains heavy. For example, 65B parameters in 16-bit is 130 GB just to store the model, not counting gradients. Multi-node distributed training or memory offloading techniques become necessary.

Data Requirements: To fine-tune effectively without overfitting, one usually needs a reasonably large fine-tuning dataset. Fine-tuning on very few examples could lead

the model to over-adjust to them and lose generality. If only a few hundred examples are available, full fine-tuning might cause the model to essentially "memorise" those examples, reducing its ability to generalise. In such cases, other methods like few-shot prompting or parameter-efficient tuning might be preferred.

Overfitting and Forgetting: If the fine-tuning distribution is narrow, the model might "forget" some of what it knew. For instance, an extreme case: fine-tune a multilingual model only on English news articles for a long time – it might degrade on other languages or on conversational style because its weights drifted from those capabilities. Ideally, fine-tuning uses a small learning rate and maybe early stopping to avoid overfitting the model.

Despite these drawbacks, full fine-tuning is sometimes the best route for maximal performance on a high-value task if one has the resources. Notably, OpenAI's GPT-3.5 and GPT-4 underwent forms of full fine-tuning for alignment (reinforcement learning from human feedback, RLHF, can be seen as a kind of fine-tuning with a special objective) [17, 22]. Also, domain-specific large models like Codex (for programming) were essentially fine-tuned versions of GPT on code data [51]. These models show that targeted fine-tuning can yield substantial gains (e.g., Codex became very proficient at coding tasks). In our thesis work, we approach fine-tuning cautiously, preferring more efficient techniques for practicality. However, understanding full fine-tuning sets the stage for why those efficient techniques work and what trade-offs they navigate.

2.4.2 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) is a prominent parameter-efficient fine-tuning method. Introduced by Hu et al. (2021) [52], LoRA targets the problem of fine-tuning large models with limited computational resources by dramatically reducing the number of trainable parameters. The key idea is to approximate the update to the model's weight matrices with a low-rank factorisation.

In a Transformer, the most significant parameter chunks reside in the weight matrices of the self-attention and feed-forward layers. These are typically high-dimensional (for instance, a 2048-dimensional hidden state might go through a 2048×8192 weight matrix in a feed-forward layer). LoRA proposes that, instead of updating the entire matrix, we can restrict updates to a low-rank subspace. Concretely, LoRA adds two small rank-r matrices A and B to an existing weight W (of size $n \times m$). Initially, W is frozen to its pre-trained value and not directly updated. Instead, A (of size $n \times r$) and B (of size $r \times m$) are introduced, with AB added to W (often with a scaling factor). During fine-tuning, only A and B are learned. Because r is chosen to be much smaller than n or m (e.g.,

$r = 8$ or 16 in many cases, where n, m could be thousands), the number of new parameters $nr + rm$ is a tiny fraction of $n * m$. For example, if W has 100 million parameters, and we choose r such that A and B together have 1 million parameters, then LoRA is training just 1% of the parameters (while effectively keeping W fixed) [52].

Crucially, the combination $W + \Delta W$ (with $\Delta W = AB$) can still represent a wide range of transformations if r is chosen adequately, especially since each layer can have its own A, B. Empirically, LoRA has been shown to achieve performance comparable to full fine-tuning on many tasks, despite updating far fewer parameters [52]. This is because a low-dimensional subspace is often sufficient to capture the task-specific adjustments needed for the model. Intuitively, if the model's pre-trained weights are already a good starting point, fine-tuning doesn't need to move in all possible directions in weight space; it only needs to move in some salient directions, which can be spanned by a low-rank space.

One advantage of LoRA (and related methods) is that the original model weights remain unchanged. This means we can preserve the original model and store the lightweight LoRA matrices for each task. Deploying the fine-tuned model involves loading the base model and adding the LoRA "deltas" on top. One can also easily switch the LoRA modules to handle multiple tasks or domains with the same base model (akin to how "adapters" work, discussed shortly). The LoRA modules can even be merged into the base model weights after training if one desires a standalone model (since $W + AB$ can be computed and used as the new weight).

LoRA is particularly attractive when one needs to fine-tune huge models (like 30B+ parameters) on consumer-grade hardware [53]. By only training a small fraction of parameters, memory usage and compute (for backpropagation) are greatly reduced [54, 55]. The approach has been widely adopted in the community; for example, many fine-tuned versions of LLaMA-7B and 13B for chat or specific tasks used LoRA to avoid full model updates.

Following the introduction of LoRA, there have been further developments. One is QLoRA (Quantised LoRA) by Dettmers et al. (2023) [56]. QLoRA takes efficiency a step further by first quantising the base model's weights to 4-bit precision, which dramatically reduces memory footprint, and then applying LoRA fine-tuning on that compressed model. Dettmers et al. showed that QLoRA can fine-tune models as large as 65B parameters on a single GPU with 48GB memory – something that would be infeasible in 16-bit without multi-GPU setups [56]. Remarkably, the performance drop due to quantisation was minimal, and they achieved state-of-the-art results on several benchmarks with 33B- and 65B-sized models fine-tuned via QLoRA. Thus, QLoRA exemplifies combining quantisation and parameter-efficient fine-tuning to push the boundaries of accessible LLM

fine-tuning. In our project, we leverage such techniques to fine-tune models on our Portuguese data without needing a server farm of GPUs. For instance, fine-tuning a 7B or 13B model with LoRA on Portuguese lexical data can be done on a single high-end GPU or two, which was within our resource constraints.

Another family of parameter-efficient methods is Adapters (sometimes just called adapter layers). The idea, originally from computer vision and later applied to NLP by [57] and others, is to insert small trainable modules into each layer of the network while keeping the original network weights fixed. For example, an adapter in a Transformer layer might be a small feed-forward bottleneck: input goes into the adapter (a down-projection to a small dimension, then an up-projection back to the original size), and the adapter's output is added to the original layer's output. Only the adapter's weights are trained. Adapters typically add an extra 1-5% of parameters per task. They allow multi-task or multi-domain training by having separate adapter weights for each domain (with the rest of the model shared). Recent work by [58] has developed a comprehensive "adapter family" framework for LLMs, integrating various adapter techniques and demonstrating their effectiveness on large models [58]. Adapters and LoRA share the philosophy of keeping most weights frozen and introducing small trainable components; the difference is in how they are implemented (LoRA alters existing weight matrices with low-rank updates, whereas adapters introduce new paths in the network).

We can summarise a comparison of full fine-tuning vs LoRA vs QLoRA (and mention adapters) in Table 2.1. Here, we'll highlight aspects such as percentage of parameters updated, memory and compute costs, and typical use cases.

In Table 2.1, we see that parameter-efficient methods like LoRA and adapters drastically reduce the number of trainable parameters (often by two orders of magnitude or more) compared to full fine-tuning. This translates directly into lower memory usage and faster training times, as well as the convenience of storing only a few million parameters per fine-tuned model (instead of duplicating the entire 13B or 30B weight matrix for each task).

For our thesis, fine-tuning is a crucial tool. In Chapter 4, the SLIM-RAFT approach is essentially a fine-tuning regimen (with retrieval augmentation) that uses LoRA to achieve domain adaptation for lexical normalisation without exorbitant cost. We chose a parameter-efficient route both out of necessity (hardware limits) and because we didn't need to upheave the entire model for our task – a gentle but targeted nudging of the model's knowledge would suffice. The success of methods like LoRA in recent literature gives confidence that we can achieve near full fine-tuning performance for our task at a fraction of the cost, which aligns well with the "efficient adaptation" goal of this research.

Fine-Tuning Method	Trainable Params	Memory/Compute	Use Case and Flexibility
Full Fine-Tuning	~100% (all weights)	Highest (full model grads)	Best task performance but high cost; risk of overfit/forgetting; one model per task.
LoRA (Low-Rank)	< 1% (low-rank mats)	Low (adds small mats)	Efficient adaptation; minimal memory; can swap LoRA modules for multi-task; base model unchanged.
QLoRA (4-bit + LoRA)	< 1% + quantisation	Very low (4-bit base)	Fine-tune very large models on single GPU; slight perf loss from quantization; ideal for resource-limited tuning.
Adapters	1–5% (per layer)	Low (small bottlenecks)	Modular training; stackable for multi-domain; base model frozen; slight inference overhead from adapter layers.

Table 2.1: Comparison of fine-tuning approaches. Trainable parameters are given as a fraction of the model's parameters (approximate). Full fine-tuning updates the entire model and yields the highest performance, but at the cost of increased computation and potential overfitting. LoRA introduces negligible overhead and enables efficient multi-tasking. QLoRA enables huge models to be fine-tuned on limited hardware by combining 4-bit quantisation with LoRA. Adapter-based tuning adds small modules to each layer and similarly keeps base weights fixed, facilitating domain-specific tuning with minimal interference.

2.5 LLM Evaluation

Evaluating the performance of LLMs, especially on generative tasks, is a non-trivial endeavour. Unlike classification tasks where accuracy or F1 can be directly computed against ground truth labels, generative models produce free-form text, and the quality of that text can be subjective or multi-dimensional (e.g., correctness, fluency, relevance, coherence) [59, 60]. This section discusses the concepts and importance of LLM evaluation, surveys classical evaluation metrics and why they can fall short, describes more modern embedded evaluation methods, and finally examines the emerging trend of using LLMs themselves as evaluators of AI outputs. We also emphasise the need for statistically sound evaluation, introducing ideas that will be formalised in Chapter 6 (the IMMBA framework), such as bootstrap confidence intervals and mixed-effects models for robust comparison.

2.5.1 Concepts and Relevance

The concept of evaluation in the context of LLMs refers to how we measure and compare model outputs with respect to desired criteria [60]. Common criteria include:

Task success or correctness: Did the model produce the right answer or a satisfactory completion of the task? (e.g., did it translate correctly, did it answer the question correctly, did it follow the instruction given?)

Quality and fluency: Is the output well-formed language, natural, and fluent? Even if correct, a disfluent or awkward phrasing might be considered lower quality.

Relevance and coherence: Does the output stay on topic and make logical sense? For longer outputs, such as summaries or stories, coherence (no contradictions or non sequiturs) is essential.

Factual accuracy: Particularly for tasks like summarisation or QA, did the model state facts correctly (as opposed to hallucinating or fabricating)?

Safety and Bias: Does the model avoid toxic or biased language? (This might be an evaluation in alignment contexts.)

For a given application, some of these criteria will matter more than others. In our case (lexical normalisation and classification of short texts), the primary measure is task success: did the model correctly normalise the input according to a gold standard or known taxonomy? Since the outputs are short (often a corrected word or a code assignment), they can sometimes be evaluated using classical metrics (such as exact-match accuracy). However, as we integrate generation (the model might generate an intermediate reasoning or a justification), evaluating those parts becomes more complex.

The relevance of robust evaluation is high: without proper evaluation, we cannot be confident in the performance gains from various techniques. Moreover, with LLMs often being non-deterministic (they can produce different outputs on different runs unless the random seed is fixed), we need evaluation methods that account for variability [60]. If one model seems better than another, is that difference statistically significant or just due to random chance in sampling or a particular test set? These are important questions, especially in research where we must support claims of improvement with evidence.

Another consideration is that traditional single-number metrics may not capture everything we care about. For example, in summarisation, a model might score higher on ROUGE (an n-gram overlap metric) than another. Yet, humans might prefer the other model's outputs for being clearer or logically structured. So human evaluation is sometimes used as the gold standard—but it is time-consuming, expensive, and not scalable for rapid experimentation [60].

Thus, the field has been exploring better automated metrics and evaluation protocols. The remainder of this section breaks down evaluation approaches into three categories:

classical metrics, embedding-based (or learned) metrics, and LLM-as-evaluator methods. Each has its role, and in fact, our evaluation methodology in Chapter 6 (IMMBA) will incorporate aspects of all three, combined with statistical analysis to draw robust conclusions.

2.5.2 Classical Metrics

Classical evaluation metrics have their roots in earlier eras of NLP, and many are borrowed from machine translation and text summarisation research, where automatic evaluation was first heavily developed. These metrics compare the model's output to one or more reference outputs (ground truth answers or desired outputs), typically by measuring lexical overlap.

Some of the well-known classical metrics include [61, 62]:

Accuracy: For tasks where the output can be considered a label or a strictly correct/incorrect answer (e.g., multiple-choice questions, classification tasks, or something like an exact string match for a normalisation), accuracy (or its complement, error rate) is used. It's simply the percentage of outputs that exactly match the expected output. In lexical normalisation, if we have a test set of pairs (noisy string, gold normalised string), we could measure accuracy as the proportion of cases where the model's output equals the gold string. This is straightforward and unambiguous, but it gives no partial credit – near-misses are counted the same as completely wrong answers.

Precision, Recall, F1: These come from information retrieval/classification and are useful when outputs can have multiple elements or we care about balancing missing vs extra content. For example, if the task is to extract keywords, one might use precision and recall to evaluate the keywords. For generative tasks, these are less directly applicable unless the task is re-framed in terms of sets of items to include, etc.

BLEU (Bilingual Evaluation Understudy): Originally developed for machine translation, BLEU [63] measures the overlap of n-grams between the generated output and a reference text. It calculates the precision of n-grams up to a certain length (usually 4), with a brevity penalty to discourage outputs that are too short. BLEU scores range from 0 to 100 (though in practice seldom above, say, 30-40 for decent translations at the sentence level). BLEU was revolutionary in MT for providing an automated proxy to human judgment of translation quality. However, BLEU has well-known limitations: it doesn't account for semantic equivalence through different wording (e.g., "lift" vs "elevator" might

be considered totally different by BLEU if not in references), and it is not very sensitive to content ordering or coherence beyond n-gram matches.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Commonly used in summarisation, ROUGE, especially ROUGE-L and ROUGE-2, measures overlap between generated summary and reference summary, often emphasising recall (what percentage of the reference n-grams were covered by the output). ROUGE-L, for example, looks at the longest common subsequence. Like BLEU, ROUGE is easy to calculate and useful for rough comparison, but it fails to capture paraphrasing or whether the summary is logically faithful to the source.

METEOR: An MT metric that tries to improve on BLEU by considering synonymy and stemming. It aligns hypotheses and references at the word level, using not only exact matches but also matches based on stems or meanings via WordNet. It then computes a score based on the precision and recall of these alignments, with some weighting. METEOR often correlates better with human judgments than BLEU on certain tasks.

Perplexity: In language modelling tasks (where the goal is to fit a probability distribution over text), perplexity is a common metric. Perplexity is essentially the exponential of the negative log-likelihood of the test data, normalised by the number of words. Lower perplexity indicates that the model assigns higher probabilities to the test data, implying better predictive power. Perplexity is used to evaluate base LMs (like when OpenAI trained GPT-3, they tracked validation perplexity). However, perplexity doesn't directly tell you about task performance on, say, QA or summarisation; it's more about the model's general language predictive quality. Also, for comparing models with different vocabularies or tokenisations, perplexity can be tricky.

Exact Match / EM and Partial Credit: In some QA evaluations (like SQuAD reading comprehension), an exact match score is used (similar to accuracy), and a softer metric is also used: a macro-averaged F1 where the predicted and gold answer strings are treated as bags of tokens (ignoring stopwords, punctuation). This gives partial credit for overlapping answers. For instance, if the gold answer is "Barack Obama" and the model outputs "Obama", exact match is 0 but token overlap F1 would be high (precision $1/1 = 100\%$, recall $1/2 = 50\%$, F1 $= 67\%$). Such task-specific metrics demonstrate how classical metrics can be tailored to provide greater nuance.

Classical metrics are attractive because they are simple, fast to compute, and well-understood. They work best when there is a single correct answer or a limited set of correct answers. They struggle for open-ended generation tasks [64, 65]. In our project, certain

aspects can still use classical metrics. For example, suppose we treat lexical normalisation as a deterministic mapping (akin to translating slang into standard). In that case, we can use exact-match accuracy: did the model's normalisation exactly match the reference? If we assign taxonomy codes (like NCM codes) to products, they can be evaluated by accuracy or F1 (if multiple codes). Those will indeed be reported in Chapter 5's results as primary metrics. But we also might want to evaluate qualitatively: how well does the model justify its decision, or how coherent is its reasoning chain? Classical metrics won't cover that.

Another issue is the reliability of these metrics. Often, differences in BLEU/ROUGE scores of a few points are not statistically significant. It's common now to apply significance tests (like a paired bootstrap test) to check if one model's BLEU is genuinely better than another's. However, significance testing was not always reported consistently. In our evaluation chapter, we incorporate statistical significance via mixed models and hypothesis testing to ensure the differences we discuss are meaningful.

2.5.3 Embedded Methods

"Embedded" evaluation methods use vector representations or internal model states to assess similarity or quality, rather than relying on surface n-gram overlap. These methods often leverage the rich semantic information captured by neural networks.

One prominent example is BERTScore [66]. BERTScore computes similarity between a candidate output and a reference by taking each token in the candidate, finding the closest matching token in the reference in terms of cosine similarity in BERT's embedding space, and vice versa, and then calculating a precision, recall, and F1 from those matches. The idea is that if the candidate and reference use different words that mean the same thing, BERT's contextual embeddings for those words will be close, thus BERTScore will reward it. For instance, candidate "The boy is fast" vs reference "The child is quick": BERTScore would likely give a high score because "boy" and "child" have similar embeddings in context, as do "fast" and "quick". BERTScore tends to correlate better with human judgments on tasks like summarisation or translation than BLEU or ROUGE, because it's more flexible about wording.

Another category is metrics that involve learned models to predict a quality score. For instance, the BLEURT metric fine-tunes a BERT model on a dataset of human ratings to predict a score for a given candidate-reference pair, effectively learning a better automatic metric [67]. These "learned metrics" can capture subtle aspects of fluency or semantic fidelity that raw overlap metrics miss. However, they risk being domain-specific or overfitting to the kind of data they were trained on.

In terms of factual consistency (a major issue in tasks such as summarisation or open-ended generation), some embedded approaches involve retrieval or knowledge checking. For example, one can use an embedding-based similarity to check if all facts stated in a summary also appear in the source text. Or use a question-generation and question-answering approach: ask questions about the summary and see if a QA model can answer them correctly using the source article, as a way to detect hallucinations (this was an approach in some research on evaluating the faithfulness of summaries) [68].

Another interesting direction is using the model's own probabilities as a measure of confidence. For instance, if an LLM produces an answer with very low probability (it was an unlikely guess under its own model distribution), that might indicate it's hallucinating [69]. Conversely, suppose the model is very confident (assigns high probability to what it's saying). In that case, it might be more factual – though this is not a foolproof indicator, as models can be confidently wrong. But some evaluation methods harness model uncertainty: e.g., the "calibration" of a model can be measured by whether probabilities it assigns correlate with actual correctness frequencies [69].

Embedded methods also include reward models from reinforcement learning. In RLHF (reinforcement learning from human feedback), a model is trained to output a scalar "quality" score given a prompt and an output (this model is trained on human preference data) [70]. Such a reward model can serve as a metric: you feed in an answer and get a score indicating how good it is (according to what it was trained to value, e.g., helpfulness). In fact, OpenAI's GPT-4 development likely used GPT-4 itself or similar models to evaluate candidate models—a sort of automated judge (leading into the next section). However, a reward model is typically tuned to a specific notion of quality (e.g., user preferences for helpfulness), and might not generalise to other criteria unless explicitly trained for them [70].

In our thesis context, embedded methods could be useful for evaluating the similarity between a model's normalised output and a reference when exact matches aren't available. For example, if the model produces a synonym or a variant that wasn't the reference but is equally valid, an embedding similarity might catch that. Also, when evaluating reasoning chains, we might use embedding-based clustering to determine whether the reasoning covers the relevant concepts in a reference solution.

One should note, however, that embedding-based metrics can sometimes be "too forgiving" – they might consider an output acceptable because it's semantically similar, even if it misses a key detail. Therefore, often a combination of metrics is advised. Indeed, one approach in evaluation is to use multiple reference sentences and take the max of BLEU or something, or use both BLEU and BERTScore to ensure both precise and semantic matches.

Finally, any automated metric is only as good as its correlation with human judgment on the dimension of interest. For new tasks, it's common to do a small human evaluation to verify that, say, BERTScore or a particular metric correlates with what humans consider better output. We have to be mindful of metric-driven development: optimising solely for an imperfect metric can lead to models exploiting it (e.g., outputting common words to achieve more n-gram overlap without truly better content).

2.5.4 LLMs Evaluating LLMs

An intriguing and relatively recent development in the AI community is the use of Large Language Models (LLMs) themselves as evaluators or judges of AI-generated content. The premise is: if we have a competent model (like GPT-4), perhaps we can ask it to critique or score the outputs of other models (or even its own outputs in a self-play scenario) in a way that aligns with human preferences.

One concrete example is G-Eval by Liu et al. [71]. In G-Eval, the authors use GPT-4 with a carefully crafted prompt (including a chain-of-thought mechanism) to have the model score or rank candidate outputs. They found that GPT-4's evaluations correlated surprisingly well with human evaluations on tasks like summarisation and dialogue, often more so than traditional metrics. The GPT-4 evaluator was instructed to follow a rubric and sometimes even to provide a rationale before giving a score, to ensure it "thinks through" the evaluation (this is a chain-of-thought used for evaluation). The result was a Spearman correlation of 0.5 or higher with human judgments, exceeding that of metrics like BLEU or ROUGE, which might be 0.2–0.3 on the same data [71]. Essentially, GPT-4, as a judge, could notice nuances like factual inaccuracies, missing information, or stylistic issues that a simple metric would miss.

Another approach is Elo ratings or pairwise battles: the model is shown two outputs for the same prompt (from two different models) and asked which is better (and maybe why) [72]. By doing this across many prompts, one can derive an Elo rating or a win rate for each model. This pairwise comparison approach can be more stable than absolute scoring. OpenAI has used such methods internally, and some research (like the Chatbot Arena by LMSYS) has used GPT-4 as a referee to rank chatbots by having it choose winners in conversations [72, 73].

However, using LLMs as evaluators comes with caveats [71, 72, 73]:

Bias: An LLM might have particular biases or preferences. For example, if evaluating dialogue, a model might prefer verbose polite answers because its training data or prompt encourages that style, whereas humans might accept concise answers. If the LLM eval-

uator is of the same family as one of the models being evaluated, it might favour it (or penalise it if it recognises its own quirks).

Alignment of criteria: We must prompt the evaluator with a clear definition of "better." If we say "which summary is better?" GPT-4 might use its own guess of what makes a good summary. We might need to say "Better = more factually accurate and covering the important points without extra detail" or some explicit criteria.

Reliability: While top LLMs (GPT-4) are pretty good evaluators in many cases, they are not infallible. They can make mistakes or be inconsistent. Interestingly, research has found that prompting them to think step by step, to reference the input and output explicitly (and sometimes to act as a harsh critic), can improve evaluation consistency.

Given all that, LLM-based evaluation is a valuable tool to complement traditional metrics. In our project, for instance, if we develop a new normalisation method and want to evaluate it but lack an extensive labelled test set for every nuance, we could ask an LLM to judge outputs in some cases. For example, we could have an LLM verify if the model's normalised output is equivalent in meaning to the original phrase or if it fits the standard dictionary definition. Or we could have it evaluate the model's explanations or reasoning steps for correctness and clarity.

Chapter 6's IMMBA approach is not about using an LLM to evaluate outputs per se, but rather about statistically analysing multiple metrics and outputs. However, one could imagine using an LLM to generate a metric (score) and then plugging that into our statistical framework. Indeed, robust evaluation involves using multiple perspectives: e.g., obtain scores from a few different metrics and treat them as numerous observations of quality.

From a statistical standpoint, when we rely on LLM judgments, we should consider them as another source of measurement – likely one with noise and bias. That's where something like a mixed-effects model (as we propose in IMMBA) could incorporate these as observations and account for their variability. For example, we might model "true quality" as a latent variable, with the LLM's scores as one indicator and human scores as another (if available), but that's quite elaborate and probably beyond what we do here. Instead, we might do simpler things: for instance, use bootstrap to get confidence intervals on average GPT-4 scores for two models to see if one is significantly higher.

To connect to our future work chapters: In Chapter 6, we formalise evaluation using statistical inference methods (such as linear mixed models and bootstrap methods; see [74, 75, 76]). These methods ensure that when we compare model A vs model B, we account for variability across items (random effects for prompts or data points) and control

for false discovery when testing many metrics or conditions [76]. Even if we involve LLM-based evaluation, we would treat it as another metric or another "judge" and still apply these statistical guards.

Statistical Reliability: It's worth expanding a bit on why we need things like bootstrap and mixed models. A common practice is to use a paired bootstrap test to see if one model's metric is significantly better than another's [77]. The idea is to simulate many samples of the test set by resampling (with replacement) and computing the metric difference each time to get a confidence interval. If the 95% CI of the difference is entirely above 0, we say it's significant at $p<0.05$. This accounts for test set variability.

Mixed-effects models [75] go further by explicitly modelling the fact that each test question might have a different inherent difficulty, and each model has a certain ability. In a simple scenario, a linear mixed model could be:

$$Score_{model,question} = overall\ mean + model_{effect} + question_{effect} + error \tag{2.1}$$

Here $question_{effect}$ is a random effect (we assume questions are a sample from a population of possible questions). This approach would allow us to estimate, for instance, how much variance is due to question difficulty vs. model differences, and yield a possibly more reliable comparison when the number of questions is limited. It's akin to doing an ANOVA but with random effects.

In Chapter 6, IMMBA (Integrated Mixed Models with Bootstrap Analysis) likely uses such models to combine multiple metrics and produce an aggregate judgment or to analyse multi-metric evaluation in a principled way.

In summary, evaluating LLMs requires a toolkit approach: use classical metrics for certain objective aspects, use embedding-based or learned metrics for semantic understanding, possibly involve LLMs as sophisticated judges, and wrap all this in a sound statistical framework to ensure robust conclusions. We aim to adopt this comprehensive approach in our thesis to fairly assess the improvements brought by Tri-Layered Intelligent Framework for LLMs (TRINITY-LLM).

2.6 Lexical Normalisation

Lexical normalisation is the process of transforming non-standard or noisy text into a standard, canonical form. In essence, it involves mapping irregular word forms, misspellings, abbreviations, or colloquial expressions to their standardised equivalents [78]. This task is particularly relevant in domains such as social media (tweets, chats), short text messages, and user-generated content, where language use is informal and often diverges from standard orthography or vocabulary [79]. In our case, lexical normalisation plays a central

role as we normalise short product descriptions and other text snippets in Portuguese to enable more effective processing (e.g., classification into a standard taxonomy such as NCM codes).

In this section, we explain the concept of lexical normalisation and its importance, outline traditional approaches to the problem, and discuss how modern AI techniques (including LLMs) have been applied to lexical normalisation. We will also highlight the specific challenges in Portuguese and in domain-specific contexts, such as product names, and bridge to the motivation for our fine-tuning and retrieval-based approach (SLIM-RAFT) to tackle these challenges.

2.6.1 Concepts and Relevance

Human language, as used "in the wild," is often messy. People frequently use abbreviations ("msg" for "message"), intentional misspellings or phonetic spellings ("luv" for "love", "u" for "you"), omit letters ("tmrw" for "tomorrow"), use slang that might not appear in dictionaries ("brb" for "be right back"), or make typographical errors. In certain contexts, such as product listings, one might see domain-specific shorthand ("blk hdphn" for "black headphone") or concatenated words due to tagging systems. Lexical normalisation aims to automatically convert such variants into a consistent, standard form.

For example:

"ur gr8" would normalise to "you are great".

"naija" (slang for Nigeria) would normalise to "Nigeria".

"iphone6s" might normalise to "iPhone 6s".

In Portuguese, "vc tbm" (common shorthand in chats for "você também") should normalise to "você também". Or "qdo" -> "quando", "td bem" -> "tudo bem".

The relevance of lexical normalisation is that many NLP tools (parsers, translation systems, classification models) are trained on or expect well-formed text. If you feed them raw social media text or noisy inputs, performance drops drastically. Normalisation can be a pre-processing step that improves downstream tasks. It's been shown to help, for instance, enhance part-of-speech tagging or dependency parsing of social media text when applied as a preprocessing step [79].

In our specific thesis, lexical normalisation is critical because we are dealing with short texts (e.g., product descriptions or names) that are often non-standard. For example, in a product catalogue, the same item might be written in multiple ways by different people: "celular c/ câmera", "cel c/ cam", "telefone celular com camera", etc., all meaning "cell phone with camera". To classify these properly under a unified taxonomy (like HS or NCM codes for customs), we benefit from normalising them to a consistent form ("celular com câmera"). Normalisation reduces the sparsity of vocabulary – different surface forms

that mean the same thing get mapped to one form, which in turn can be linked to the correct category. Additionally, if we want to use an LLM primarily trained on standard text, normalising the input will help it better recognise the content.

Lexical normalisation can be seen as a form of translation: from a non-standard dialect/register to the standard form of the language. Unlike complete machine translation, however, normalisation usually preserves the language (e.g., Portuguese to Portuguese) and mostly concerns individual words or slight rephrasing, not complete reordering or syntactic changes.

One of the early systematic definitions of the task was provided by Han et al. [79], who focused on social media. They highlighted that out-of-vocabulary (OOV) words in tweets often correspond to alternate spellings or abbreviations that have standard equivalents. Their approach involved first identifying which tokens need normalisation (since not every token is out-of-vocab or in need of change) and then generating or looking up the candidate normalisations.

The task can be broken down into sub-problems:

Detection: Decide which words (tokens) in the input are non-standard and should be transformed. For instance, in "novos fones_ouvido c/ mic", you might identify "fones_ouvido" (with underscore or missing preposition) and "c/" as needing normalisation.

Mapping/Generation: For each such token or phrase, find the correct normalised form. This could be a one-to-one mapping or a one-to-many mapping (sometimes a single token expands to multiple tokens, like "c/" -> "com"). Conversely, numerous tokens might be compressed into one (e.g., by removing spaces in certain languages), though that's less common.

Contextual validation (if needed): Some words might be ambiguous. E.g., "gd" could be "good" or "grand" depending on context. So sometimes you need the context to choose the proper normalisation.

The importance of context and domain: Lexical variants are often domain-specific. Internet slang evolves rapidly; domain jargon in product listings might have unique abbreviations (like "NWT" in apparel = "New With Tags"). In our use case, product listings and possibly addresses (e.g., CEP and postal addresses) have their own shorthand conventions. E.g., "Apto" might mean "apartamento" (in addresses) or "apt.", etc. So a normaliser might need some domain knowledge.

The next sections will address how these were tackled in the literature and then how AI, especially recent neural models, approach it.

2.6.2 Traditional Approaches

Traditional approaches to lexical normalisation mostly date to the 2000s and early 2010s, and they were often grounded in rules, dictionaries, or relatively simple statistical models [80].

Dictionary-based substitution: A straightforward method is to compile a dictionary of common slang/abbreviations to standard form mappings. For example, a lexicon mapping “u”->“you”, “ur”->“your/you’re” (maybe multiple options), “brb”->“be right back”, etc. Then, normalisation is just a lookup-and-replace. This is high-precision for known abbreviations but has low recall, since language is creative and new slang emerges, and it doesn’t handle misspellings that aren’t in the dictionary.

Rule-based transformations: These could include heuristic rules like “if a word has repeated characters (looooove), reduce them to two (loove) or one (love) depending on what yields a known word.” Or phonetic substitution rules (like maybe “f” <-> “ph” or certain leetspeak, “4” -> “for” or “ate” depending). Han et al. (2013) [79] noted patterns like inserting vowels where they were omitted, expanding shorthands, etc. These rules often need to be language-specific. For English, a typical rule is to remove repeated letters beyond 2 (since English sometimes elongates letters for emphasis, e.g., "cooool").

Noisy Channel model: Some works treated text normalisation like a spell correction problem, using a noisy channel or edit distance approach. Essentially, they model that the non-standard text is a “noisy” version of some “hidden” correct text, and try to find the most likely hidden text given the observed one. This involves a model of the likelihood of certain edits (insertion, deletion, character substitution) and a language model for the likelihood of the corrected text. For instance, Cook et al. (2014) [81] used such an approach for SMS normalisation. This can handle unseen misspellings: e.g., if someone writes “tommoroww”, an edit distance approach might find “tomorrow” as the closest dictionary word. However, purely character-based corrections might falter in more semantic or abbreviation cases (e.g., “np” -> “no problem” isn’t a few-character edit away).

Statistical Machine Translation (SMT): Another approach is to treat normalisation as a translation problem and apply MT techniques. If one can gather a parallel corpus of noisy text and its normalised version, one can train an MT system (like phrase-based SMT in the old days, or even modern neural MT) to translate from noisy to normalised [82]. Back in the mid-2010s, some researchers did use phrase-based SMT for this [83]. The

challenge is gathering training data; often, it has to be done by crawling social media and then manually or semi-automatically creating normalised references. Shared tasks such as W-NUT 2015 and W-NUT 2021 (Workshop on Noisy User-generated Text) provided datasets for normalisation across multiple languages, spurring work in this area [84, 85].

Multilingual and cross-lingual issues: Most early work was on English, but normalisation for other languages has its own challenges and was less studied until shared tasks like MultiLexNorm (W-NUT 2021), which included languages like Spanish, French, Portuguese, etc [84, 85]. Each language has its texting abbreviations and orthographic quirks. Portuguese, for instance, often drops diacritics in informal text (e.g., "voce" for "você"), which is a normalisation issue. Also, Portuguese has clitic contractions that might be incorrectly spaced or not, etc.

In summary, traditional methods often rely on string similarity and knowledge bases. They can work well for relatively frequent or straightforward transformations, but struggle with the long tail of creative slang and error patterns. They also often treat words in isolation, sometimes failing when context is needed (e.g., "read" vs "red" depending on the context of a misspelling).

One more application of lexical normalisation is in the area of taxonomy classification (like HS code classification of products). While not exactly traditional "social media" normalisation, in product descriptions, one might have synonyms or various ways to name a product. The HSCodeNet paper by Du et al. (2021) [86] aimed to classify commodity descriptions to HS codes. They introduced a neural network combining local (word-level, sequential) and global (text-level, spatial) information. Implicitly, part of what such a model would learn is to normalise synonyms or variants to the exact internal representation. For example, if "couch" vs "sofa" appear, the model might learn that both indicate the same category. While not explicit normalisation, it is related: the model's embedding layer could be seen as normalising different tokens that mean the same thing to similar vectors. However, one might also handle that by pre-normalising (like mapping all synonyms to a canonical name before classification).

It is worth noting that the line between lexical normalisation and entity/alias resolution can blur. For instance, mapping "Coca Cola" vs "Coke" to the same canonical form is akin to normalisation (though one could call it alias resolution if it's brand names). In our project's context (NCM classification), normalising product descriptions might involve expanding certain abbreviations (e.g., "PC" -> "computador pessoal" if needed) or standardising measurement units ("cm" vs "centímetro"). Those can be vital for correct classification.

2.6.3 The Use of AI for Lexical Normalisation

As with many NLP tasks, the last decade has seen a shift from rule-based or statistical methods to neural network approaches for lexical normalisation. The availability of annotated datasets (e.g., from the W-NUT shared tasks [84, 85]) enabled researchers to train end-to-end normalisation models.

Approaches include:

Sequence-to-sequence (seq2seq) models: Treat the entire input text (or each word) as a sequence to be transduced into another sequence. For instance, one could have a character-level seq2seq model that reads a word like "qdo" and outputs "quando". Or a word-level seq2seq that reads a whole tweet and outputs the whole tweet in normalised form, akin to translation. Early on, RNN-based seq2seq with attention (like those used in MT around 2015) were applied. They can naturally handle things like one-to-many mappings (inserting missing words) or many-to-one (merging tokens), as long as the training data covers those patterns [87]. A challenge is that training data may be limited, but techniques like data augmentation (creating synthetic noisy versions of clean text) have been used [88].

Character-level Transformers: Models like ByT5 [89] and other byte-level or character-level models have been found effective for normalisation tasks, since the problem often revolves around character edits. T5, in particular, is a variant of the T5 model that operates on UTF-8 bytes directly and has been shown to excel at tasks such as misspelling correction and even the MultiLexNorm 2021 shared task. In fact, the winning system of MultiLexNorm 2021 [90] for many languages used a ByT5-based approach, because it can naturally model character deletions/insertions in a sequence-to-sequence framework without worrying about unknown tokens.

Adapter or multi-task approaches: Sometimes normalisation is used as a pre-processing but one could integrate it into a pipeline or multi-task model. For example, one could train a language model or classifier with an auxiliary objective to normalise the text. However, typically a separate normaliser is applied [91].

BERT-based tagging or masking: Another method could be to use a masked language model (like BERT) to fill in or correct words. Imagine replacing each out-of-vocab token with a [MASK] and letting BERT predict a likely word. However, BERT might not know the slang at all (because it's outside its vocabulary), so this could be hit-or-miss. But one could fine-tune BERT for normalisation by treating it as a sequence tagging

problem, where each character is labelled as "keep/delete/modify," etc., or by using edit labels [92].

In Portuguese specifically, resources like BERTimbau [4], a Portuguese BERT model, could be used to help normalisation by scoring candidate replacements in context. There's also bert-base-multilingual, which covers Portuguese and might be used if monolingual is not available. But now, with large multilingual LLMs, one could also prompt a model like GPT-4 in Portuguese: "Normalise the following text: ..." and see how well it performs (this would be an interesting test; GPT-4 presumably could handle a lot of Portuguese internet slang due to broad training).

Additionally, the context of our research deals with structured classification in a fiscal domain (NCM code assignment, postal code standardisation). Lexical normalisation here might involve things beyond everyday social media slang:

- Normalising numeric formats, units (like converting "1,5kg" to "1.5 kg" or a standard format).
- Handling multi-word term variations (like "Notebook" vs "Laptop").
- Possibly normalising words to a specific vocabulary (like mapping colloquial product names to official names in a taxonomy).

Finally, lexical normalisation is not an end in itself in our case; it is a means to improve classification. In Chapters 4 and 5, we will see how we incorporate normalisation into a retrieval-augmented fine-tuning approach (SLIM-RAFT) and into a retrieval pipeline (Two-Step RAG). The idea is that, by normalising (or at least accounting for) lexical variants, our model can retrieve the correct references and make the correct NCM code classification.

To make the bridge explicit: Suppose we have a retrieval system that fetches similar past cases to help classify a new product description. If the new description says "cel c/ cam" and our database of past cases has only examples phrased as "celular com câmera", a naive retrieval might miss the connection. But if we normalise "cel c/ cam" -> "celular com camera" (maybe still missing the accent, but close enough), the vector representations or keyword overlap improve, and the correct examples can be retrieved. Similarly, in fine-tuning, if we augment training data with normalised forms or explicitly teach the model that "qdo" and "quando" are equivalent by showing both in context, the model becomes more robust.

In summary, AI-based lexical normalisation has largely shifted towards neural sequence-generation or tagging models that can capture more complex transformations than earlier rule-based systems. For our purposes, leveraging an LLM's knowledge (via fine-tuning

or prompting) is a promising direction, since a large model could implicitly learn many normalisation patterns from data. The challenge is ensuring it focuses on the specific variants in our domain (which might be underrepresented in generic training data) – hence the need for fine-tuning and retrieval strategies tailored to our domain.

With the background now covered from LLM fundamentals to evaluation and normalisation, we have set the stage for the subsequent chapters. In Chapter 4, we will see how these concepts come together in the design of the SLIM-RAFT fine-tuning method, which extends retrieval-augmented training to better handle lexical variations in Portuguese. In Chapter 5, we will examine the empirical results of using Two-Step RAG for enhanced retrieval and the gains from fine-tuning. And in Chapter 6, we will discuss how the evaluation methodologies mentioned (including LLM-based evaluation and statistical analysis) are applied to rigorously assess our models' performance improvements.

2.7 Chapter Summary

This chapter provided the conceptual basis for this research. It introduced LLMs, prompt engineering techniques including Chain-of-Thought, and the structure of RAG systems. It also reviewed fine-tuning strategies such as LoRA, evaluation methods, and approaches to lexical normalisation. These foundations support the methodological innovations and evaluation frameworks presented in Chapters 4 through 6.

Chapter 3

Related Works and State of the Art Review

In Chapter 2, we provided foundational background on Large Language Models (Section 2.1), Retrieval Augmented Generation (Section 2.3), prompt engineering (Section 2.2), fine-tuning techniques (Section 2.4), evaluation metrics (Section 2.5), and lexical normalisation (Section 2.6). We now build on that groundwork by examining recent literature (primarily 2020–2025) relevant to our research problem. In particular, this chapter reviews: (a) retrieval-augmented fine-tuning (RAFT) methods that integrate retrieval into the training of LLMs (Section 3.2); (b) approaches to lexical normalisation, especially for Portuguese short-text domains with structured label taxonomies (Section 3.3); and (c) the state-of-the-art advancements in prompt engineering for fine-tuning and for retrieval-augmented generation, as well as statistical methodologies for robust LLM evaluation (Section 3.5 and sub-sections therein). Throughout, we contrast competing approaches, highlight their requirements and limitations, and position our work in the context of this literature. This discussion directly underpins the tri-layered *TRINITY* framework introduced in Chapter 1, wherein Stage 1 (SLIM-RAFT) will leverage insights from RAFT and parameter-efficient tuning, Stage 2 (Two-Step RAG) will draw on advances in retrieval and prompting, and Stage 3 (IMMBA) will build upon modern evaluation practices. The following search engines were used to find academic papers related to our research: SCOPUS, Web of Science, Google Scholar and Scholar GPT. The first section of this chapter shows how these search engines were used.

3.1 Scientific Research Repositories and Web Search

As part of the development of Chapter 3, a systematic bibliographic search was conducted using the SCOPUS and Web of Science databases to identify recent and relevant studies

aligned with the themes explored in this review. The goal was to gather authoritative publications addressing large language models, fine-tuning strategies, retrieval-augmented generation techniques, and statistically grounded evaluation methods. The search and selection procedures adopted are described below.

3.1.1 SCOPUS and Web of Science

The SCOPUS and Web of Science databases were searched in the same way. We limited searches to the keywords indicated below and the publication period from 2020 to the present day.

- large language models AND fine tuning AND retrieval augmented generation (497 documents on SCOPUS and 308 on Web of Science);
- large language models AND evaluation AND statistical analysis (399 documents on SCOPUS and 455 on Web of Science).

The results were sorted in descending order by the number of citations for each search. The first 50 papers were reviewed by reading the titles and abstracts; then, studies deemed relevant were selected by reading their introductions and conclusions. Finally, papers related to our research were reviewed and included in this thesis in chapters 2 and/or 3.

Figure 3.1 shows the plots of the aggregated values of the total number of academic papers observed. In total, 497 documents were found using the following filtering rule: (TITLE-ABS-KEY (large AND language AND model) AND TITLE-ABS-KEY (fine AND tuning) AND TITLE-ABS-KEY (retrieval AND augmented AND generation)) AND PUBYEAR $>$ 2019.

3.1.2 Scholar GPT and Google Scholar

The traditional bibliographic search presented in the previous section was complemented with a search in the *Scholar GPT* application, where we used the following prompt:

```
-----------------
Find and summarise recent academic articles (2020-2025) related to
the lexical normalisation of short texts in Portuguese, particularly
using large language models (LLMs) or other deep learning techniques.

For each article, extract and clearly present:
    Problem addressed;
    Proposed method or model;
```

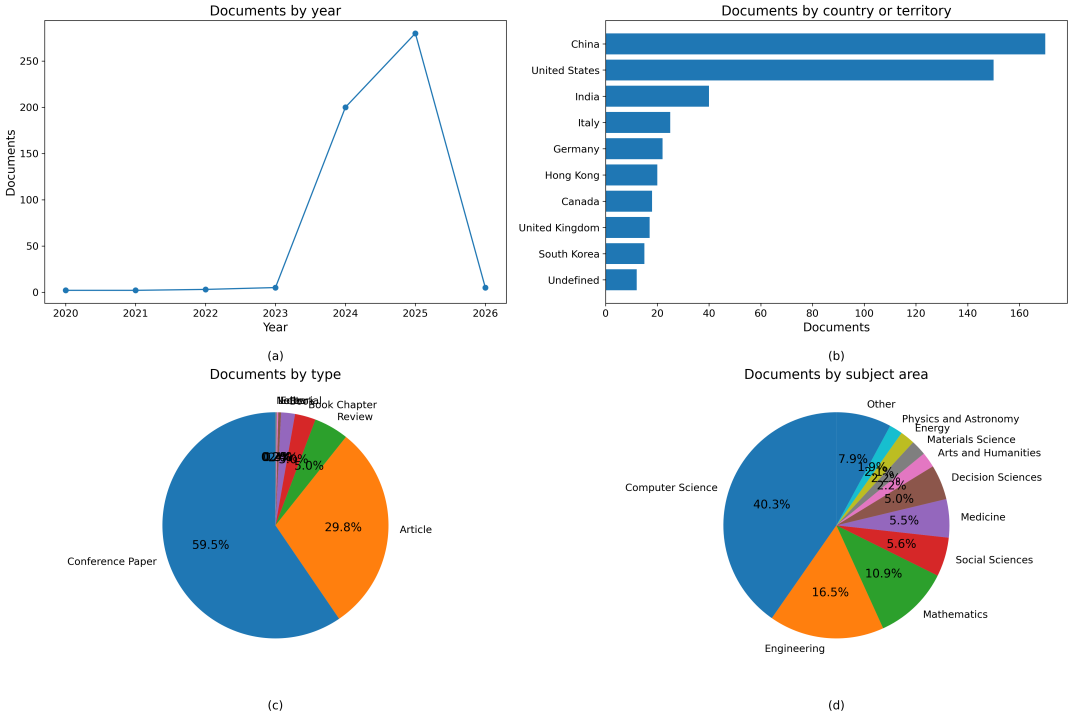

Figure 3.1: Comprehensive analysis of documents found in the SCOPUS database.

```
    Main results or achievements;
    Statistical evaluation techniques (if used).

Prioritise peer-reviewed publications or arXiv preprints in NLP, AI,
or applied linguistics. Focus on works that involve:
    Portuguese or multilingual language models (e.g. BERTimbau, BoDe,
    TeenyTinyLLaMA);
    Hierarchical classification or label taxonomies (e.g, NCM/HS
    codes);
    Retrieval-based or prompt-based learning (e.g. RAG, RAFT,
    instruction tuning);
    Evaluation frameworks that apply statistical rigour (e.g. LMMs,
    bootstrap resampling, FDR control, and inter-rater agreement).

Prefer sources that include datasets, benchmarks, or comparative
studies relevant to product classification or language standardisation
tasks.
-----------------
```

We review the output and use the *Google Scholar* website to reach out and download

the articles' PDF versions, if available. This review was done similarly to the work on searches in the previous subsection.

The combined search strategies across SCOPUS, Web of Science, Scholar GPT, and Google Scholar provided a broad and up-to-date view of the current research landscape. The selected studies informed the theoretical foundations and comparative analyses presented in Chapters 2 and 3, ensuring that this thesis is grounded in the most relevant and recent developments in the field.

3.2 Retrieval Augmented Fine-tuning (RAFT)

Large pretrained language models can be adapted to new knowledge either by retrieving relevant information at inference time (retrieval-augmented generation, or Retrieval Augmented Generation) or by fine-tuning the model's parameters on domain-specific data. *Retrieval Augmented Fine-Tuning* (RAFT) is a recently proposed strategy that combines these paradigms by integrating a retrieval step into the model fine-tuning process [43]. In RAFT, instead of fine-tuning on $(\text{input}, \text{output})$ pairs alone, the model is fine-tuned on triplets (q, D, a) where q is a query (e.g. a question), D is a set of documents retrieved for q, and a is the desired answer. The key idea is to train the LLM not just to produce the correct answer a, but also to effectively utilise D — ignoring irrelevant "distractor" documents and focusing on the pertinent facts [43]. During RAFT fine-tuning, the model may be explicitly encouraged to *cite* or copy supporting evidence from the relevant document in D that contains the answer, thereby grounding its output in the source text. Moreover, RAFT often adopts a chain-of-thought style training signal: the model is trained to generate an explainable, step-by-step reasoning path that leads to the answer, including references to the source material. By learning to filter out noise and highlight supporting snippets, RAFT aims to endow the model with improved reasoning and factual accuracy in open-book settings [43].

The RAFT main guideline is based on the question, "How best to prepare for an Examination?". That strategy is shown in Figure 3.2 (up) and is explained as follows: (a) Fine-tuning-based approaches implement "studying" either through direct "memorisation" of the input documents or by responding to practice question-and-answer sessions without referring back to the documents. (b) Conversely, in-context retrieval methods do not fully exploit the learning opportunities the fixed domain provides and are akin to taking an open-book examination without prior study. While these methods capitalise on in-domain learning, they do not adequately prepare for open-book examinations. In contrast, our approach (c) RAFT utilises fine-tuning with question-answer pairs while referencing the documents within a simulated imperfect retrieval environment—thereby effectively

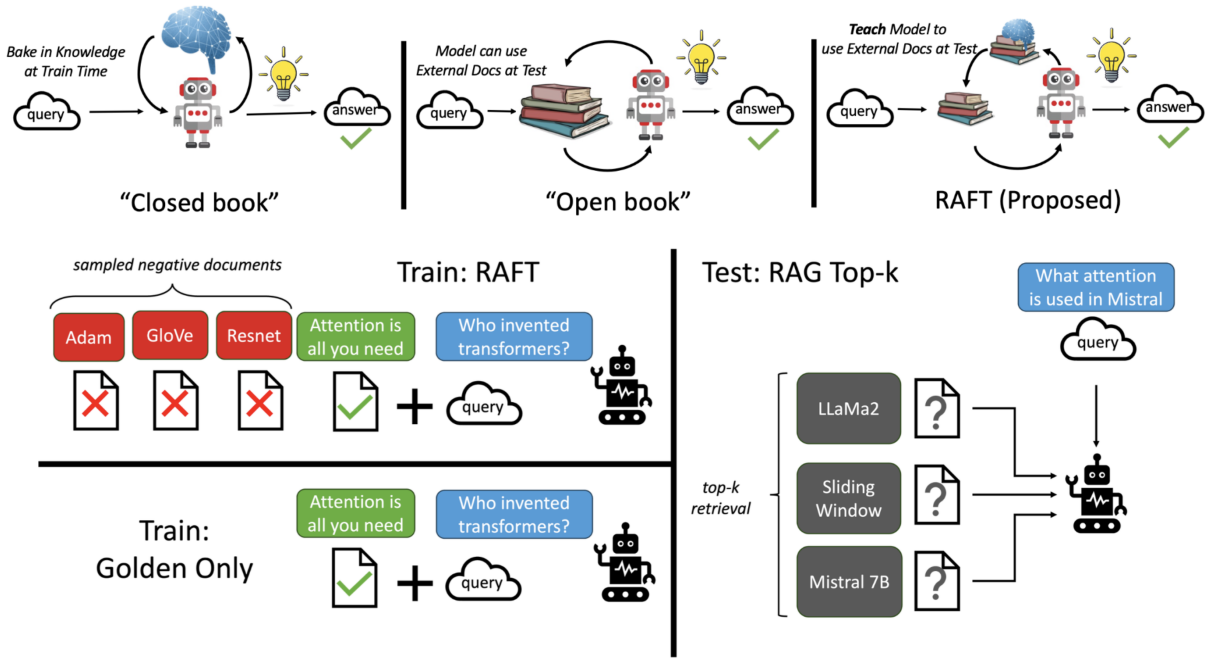

Figure 3.2: RAFT: Paralel between "How to prepare for a test?" and the RAFT fine-tuning method [43].

preparing for the open-book exam scenario. The same Figure 3.2 (bottom) presents an overview of the RAFT method: The bottom-left figure illustrates the approach of adapting Large Language Models (LLMs) to derive solutions from a set of both positive and negative documents, in contrast to the standard RAG setup.

Zhang et al. (2024) introduced RAFT in the context of domain-specific QA tasks and reported consistent performance gains across several benchmarks when compared to standard fine-tuning or vanilla RAG prompts [43]. For example, on a PubMed QA dataset (biomedical domain) and the multi-hop HotpotQA, RAFT-tuned models outperformed models fine-tuned without retrieval by effectively leveraging domain documents and ignoring misleading data. Notably, RAFT even improved over RAG baselines where the model had access to a retriever at inference but was not trained to deal with distractors. These results suggest that baking domain knowledge into model parameters via retrieval-augmented training can yield better in-domain reasoning ability than relying on retrieval at inference alone. RAFT represents a post-pretraining adaptation recipe to efficiently inject new knowledge into an LLM, which is especially useful for private or time-sensitive data that cannot be covered in general pretraining.

It is instructive to compare RAFT with alternative adaptation approaches along several dimensions, as summarised in Table 3.1. Classic RAG (as formulated by Lewis [37]) leaves the LLM parameters unchanged and instead prepends retrieved text to the input

at inference time, relying on the model's attention mechanisms to incorporate external knowledge. This has the benefit of minimal training (often only the retriever, such as a DPR model, is trained [37]) and allows up-to-date information to be fetched at runtime. However, pure RAG can struggle when the model's internal knowledge and the retrieved evidence conflict, or when many retrieved passages are irrelevant. Since the model was not explicitly taught how to ignore or handle false passages, it may get distracted or propagate errors from them. RAFT addresses this by using training signals that differentiate useful vs. distractor documents: essentially, RAFT provides supervised examples of how to ground answers in relevant text and disregard the rest [43]. The trade-off is that RAFT requires curated training data with queries and relevant documents, which can be costly to assemble for specialised domains.

Approach	**Data Requirement**	**Domain Drift Handling**	**Knowledge Grounding**
Classic RAG [37]	Moderate (needs Q–A for retriever; no LM fine-tune)	High flexibility (update docs without retraining LM)	External only (relies on retrieved text; may hallucinate if retrieval fails)
RAFT [43]	High (needs curated Q–doc–A triples with relevance labels)	Medium (model tuned to specific corpus; needs re-tune if corpus changes)	Both (grounds answers in documents and encodes domain info in parameters)
PEFT (e.g. LoRA [52])	Low–Medium (can leverage general instruction data; fewer parameters to train)	Low–Medium (specialised adapters per domain; easy to swap out, but each adapter static)	Parametric (all knowledge in tuned weights; prone to hallucination if gaps)

Table 3.1: Comparison of knowledge adaptation approaches: classical Retrieval-Augmented Generation (RAG) vs. Retrieval-Augmented Fine-Tuning (RAFT) vs. Parameter-Efficient Fine-Tuning (PEFT).

Another axis of comparison is full-model fine-tuning versus Parameter-efficient Fine-Tuning (PEFT) approaches. Full fine-tuning (updating all model weights) on domain data can achieve strong results but is often infeasible for very large models due to computational and memory constraints. Parameter-efficient fine-tuning techniques like LoRA [52] and QLoRA (quantised LoRA) [56] offer a lightweight alternative: they freeze the original model weights and train only a small number of additional parameters (for example, low-rank adaptation matrices in each transformer layer for LoRA). These methods drastically reduce the number of trainable parameters (LoRA can cut it by 10,000× for GPT-3 175B [52]) and avoid full model deployment for each domain. RAFT can in fact be combined with PEFT: instead of full-model RAFT fine-tuning, one can apply RAFT training objectives while using LoRA updates. This was implicit in Zhang 's experiments, which fine-tuned a Gorilla LLM on API documentation QA with RAFT – such a procedure could

be made more memory-efficient by injecting LoRA layers. The downside of PEFT (and any fine-tuning) is that the model's parameters become domain-specialised, potentially losing some generality, and need maintenance as knowledge evolves. RAG, by contrast, can update the index of documents without retraining the model, making it flexible to domain drift.

Finally, RAFT relates to *retriever training regimes*. In a typical RAG pipeline, a neural retriever like DPR [40] or ColBERT [93] is trained (often with contrastive learning) to fetch relevant passages given a query. Retriever training uses positive passages (ground-truth evidence) and negatives (distractors) to learn vector embeddings. RAFT does not train a retriever per se, but it can be seen as complementary: one could use a high-recall retriever to supply D for RAFT training, then fine-tune the model to focus on the truly relevant piece within D. Interestingly, a recent approach called REPLUG [94] closes the loop between retriever and LLM: it treats the LLM as a black box and fine-tunes the retriever to maximise the LLM's final task performance. REPLUG demonstrated that using the LLM itself as the reward signal for retrieved evidence (i.e. boosting documents that help the model get the answer right) can significantly improve end-to-end accuracy. This philosophy aligns with RAFT's objective of tailoring the entire system (retrieval + generation) to the target-domain task. However, RAFT goes one step further by updating the generator model weights rather than just the retriever. In summary, RAFT can be seen as a specialised fine-tuning that teaches an LLM how to *use* retrieved knowledge effectively, as opposed to either leaving the model unchanged (RAG) or updating it without context (standard fine-tuning). This approach is particularly apt for our purposes, as Stage 1 of the TRINITY framework (Chapter 4) implements a domain-specific RAFT variant (dubbed *SLIM-RAFT*) to fine-tune a Portuguese LLM for lexical normalisation efficiently.

This thesis proposal, SLIM-RAFT, as shown in Chapter 4, differs from the original RAFT approach by prioritising efficiency, controllability, and domain specificity over large-scale instruction tuning. While RAFT relies on instruction-heavy fine-tuning with externally retrieved examples to improve general instruction following, SLIM-RAFT introduces a lightweight, retrieval-aware fine-tuning strategy that integrates curated, domain-specific signals directly into the training process. This design significantly reduces computational and financial costs, enables fine-grained control over the training data and prompts, and is better suited for specialised tasks such as Portuguese product description classification.

3.3 Language Models for Lexical Normalisation

Lexical normalisation is the process of transforming noisy or non-standard text into a normalised, standard form. In our case study, this involves mapping short, often ill-formed product descriptions to standardised codes in a taxonomic inventory (the Mercosul Common Nomenclature, NCM). This task is challenging because the input "lexicon" (e.g. a product name as written by a layperson) may include spelling variations, abbreviations, domain-specific jargon, or missing context, yet it needs to be normalised to the correct official description or code. Traditional approaches to lexical normalisation (particularly in contexts like social media text) used pipeline architectures: e.g., spell-correct the text, expand abbreviations using a dictionary, and then apply rules or statistical models to choose the closest standard term. Such pipelines, however, are brittle and language-specific, and require substantial maintenance for each new domain.

In the context of product description normalisation for tax codes, earlier efforts treated it as a text classification problem: given a description, select the correct code from a fixed taxonomy. Classical machine learning approaches ranged from Naïve Bayes and SVMs with handcrafted features, to early neural networks using word embeddings. Those approaches struggled with the scale of the label space and the hierarchical nature of the codes [95]. The NCM (like the HS code system worldwide) is hierarchical: codes have chapters, headings, sub-headings, etc., reflecting a tree of commodity categories. Simply treating each of the thousands of leaf codes as independent classes ignores valuable structured information. Recent research has proposed specialised architectures to handle this. For example, Du et al. (2021) introduced *HScodeNet*, a neural model that combines a sequential encoder for the text with a graph-based module to capture the hierarchical structure of HS codes [86]. By incorporating parent–child relationships between codes and even modelling interactions between different segments of the description, HScodeNet achieved superior accuracy on HS classification tasks. This demonstrates the benefit of infusing domain ontology knowledge into the model architecture or loss function (e.g. via a label correlation loss as in HScodeNet).

A parallel thread in recent years is the rise of transformer-based language models for text classification in specific languages. For Portuguese, a milestone was the release of BERTimbau [4], a BERT-base model pretrained on large Brazilian Portuguese corpora. BERTimbau and similar models capture the general language patterns and can be fine-tuned for tasks like our NCM classification. Indeed, Lima et al. (2022) [96] found that a Portuguese-trained BERT model slightly outperformed a multilingual BERT on an NCM auto-classification benchmark (MCC 0.85 vs 0.83) by better understanding domain-specific vocabulary and Portuguese semantics. The implication is that for lexical normalisation in Portuguese, using a model with native-language pretraining (or at least some

training in Portuguese) yields gains in dealing with nuances like plural/singular forms, synonyms, or common misspellings in Portuguese product names. Multilingual models (e.g. mBERT or XLM-R) can handle Portuguese to some extent but may be slightly less attuned to its morphology and orthographic quirks [96].

However, even BERT-sized models (110M parameters) may fall short on highly ambiguous cases or when presented with phrasing not seen in training. Larger LLMs bring superior generalisation and world knowledge that could help infer the correct normalisation from context. For example, GPT-4 can often decipher an obscure abbreviation or a misspelled word in a product description by drawing on its vast training data, as can be seen in the results of our experiment in Chapter 5. The downside is cost and latency; calling a 175B+ parameter model or an API at scale for every classification can be impractical. An interesting middle ground has emerged in the form of smaller, domain-tuned LMs. Some recent works have produced compact Portuguese LMs specifically for limited-resource settings. For instance, Corrêa et al. (2024) [24] released *TeenyTinyLlama*, a pair of open-source Brazilian Portuguese LLMs with only a few hundred million parameters, demonstrating that with the right training data even models an order of magnitude smaller than BERTimbau can achieve strong results on language tasks. Similarly, open initiatives like *Bode* and *GemBode* fine-tuned smaller LLaMA-2 models (7–13B) on Portuguese data to create specialist LLMs for Portuguese [50]. These compact models are attractive for lexical normalisation: they can be locally deployed, have faster inference, and still benefit from some of the instruct-following and reasoning abilities of larger LLMs (especially when fine-tuned on instruction datasets).

Another consideration is whether to frame lexical normalisation as a generation task or a classification task. A large model can be prompted to "rewrite the description in standard form" or even to predict the likely NCM code by generating it. Prompt-based generation has the advantage of possibly explaining its reasoning (in an interactive setting), but ensuring consistency with the official taxonomy can be hard. Classification, on the other hand, guarantees a valid code as output, and the hierarchical structure can be exploited. Some recent systems adopt a hybrid two-step approach: first use an LLM to extract or infer structured information (like product category, materials, usage), then use a rules engine or smaller model to map that to a code [97]. This is akin to how a human expert might first identify key attributes of the product then consult the tariff schedule. Data augmentation is also commonly used to bolster training in this domain: e.g., generating synthetic product descriptions (perhaps via an LLM prompt) to increase coverage of rare categories, or back-translating descriptions (English–Portuguese) to introduce lexical diversity [98].

In summary, the state of the art for lexical normalisation in Portuguese (and similar languages) is trending towards leveraging pre-trained language models fine-tuned on domain data, with attention to taxonomy structure and low-resource constraints. Small-to-medium LMs like BERTimbau or TeenyTinyLlama can be effectively fine-tuned for classification, while larger LLMs can be used in a retrieval-augmented or few-shot manner to supplement those models for the hardest cases. In Chapter 4, we will explore a RAFT-based fine-tuning approach (SLIM-RAFT) that uses a combination of retrieval and parameter-efficient tuning to specialise a Portuguese LLM for the short-text normalisation task. Our approach will also draw on ideas of hierarchical classification (ensuring that the model's predictions remain consistent with the NCM code hierarchy) and data augmentation to cover the wide range of lexicon variations in product descriptions.

3.4 Portuguese LLMs Evolution

This subsection analyzes the progression of large language models (LLMs) for Portuguese language applications, focusing on six notable models: LLaMA, LLaMA 2, Cabrita, Sabiá, Bode, and TeenyTinyLlama. The review tracks the evolution from early models, LLaMA and LLaMA 2, to the subsequent advancements brought by Cabrita, Sabiá, Bode, and TeenyTinyLlama, highlighting their contributions to the field of natural language processing for Portuguese.

LLaMA [21], introduced as an open and efficient foundation language model, set the stage for subsequent developments in language processing. With models ranging from 7B to 65B parameters, LLaMA demonstrated the feasibility of training state-of-the-art models using publicly available datasets exclusively. LLaMA-13B even outperformed GPT-3 on various benchmarks, showcasing its competitive performance despite its smaller size. Building upon this foundation, LLaMA 2 [99] refined the models, particularly in dialogue use cases, by introducing fine-tuned LLMs optimised for chat interfaces. These early models laid the groundwork for subsequent advancements in NLP. Next, we will present the latge language models, pre-trained in Brazilian Portuguese, that preceded the TeenyTinyLlama model used in this study.

Cabrita: Closing the Gap for Foreign Languages Cabrita [49] addressed the challenges existing LLMs face, particularly in scenarios involving conversational memory resources and specific domain problems. By proposing a methodology that successfully enhanced performance and efficient tokenization for Portuguese, Cabrita offered a cost-effective solution without compromising quality. This advancement significantly contributed to bridging the gap for foreign languages like Portuguese in LLMs.

Sabiá: Portuguese Large Language Models Sabiá [100] focused on the significance of monolingual pretraining for improving the performance of LLMs in Portuguese. By extending the pretraining of an English-centric model using Portuguese corpora, Sabiá demonstrated notable improvements in performance across various datasets. This approach enhanced language proficiency and injected domain-specific knowledge, particularly beneficial for tasks requiring cultural or linguistic nuances specific to Portuguese.

Bode: A Fine-Tuned Large Language Model for Portuguese Prompt-Based Tasks Bode [50] addressed the challenges of prompt-based tasks in Portuguese by introducing fine-tuned models based on the LLaMA 2 architecture. With versions optimized for Portuguese prompts, Bode offered satisfactory results in classification tasks, filling a crucial gap in Portuguese NLP applications. This model's tailored approach provided a promising solution for leveraging LLM capabilities in Portuguese-specific contexts.

TeenyTinyLlama: Open-Source Tiny Language Models Trained in Brazilian Portuguese TeenyTinyLlama [24] aimed to democratize LLMs for low-resource languages like Brazilian Portuguese. By developing compact models under permissive licenses, TeenyTinyLlama provided accessible solutions for NLP tasks in low-resource settings. This initiative contributed to fostering open-source development and extending the reach of LLMs to diverse linguistic communities.

3.5 State of the Art Review

Having examined RAFT and lexical normalisation in isolation, we now review broader state-of-the-art developments that inform our research. We focus on three areas: prompt engineering techniques as applied to fine-tuning paradigms (Section 3.5.1), prompt engineering in the context of retrieval-augmented generation (Section 3.5.2), and advanced statistical methods for evaluating LLM performance (Section 3.5.3). These topics correspond to cross-cutting concerns in modern LLM systems: how to best elicit knowledge or reasoning (prompting), how to ground model outputs in external data (RAG prompting and filtering), and how to reliably measure success (evaluation metrics and statistics). Each subsection below surveys key concepts and recent proposals, setting the stage for the methodologies we develop in Chapters 4–6.

3.5.1 Prompt Engineering for Fine-Tuning

Instruction tuning and prompt design: One major trend in the last few years is *instruction tuning*, wherein LLMs are fine-tuned on a dataset of prompts and responses to

better follow human instructions. Models like GPT-3.5/GPT-4, LLaMA-2, and FLAN-T5 [101] have all benefited from an additional stage of supervised fine-tuning on instruction–response pairs, often curated from crowd-workers or mined from the web. The result is that the model becomes adept at understanding an input query's intent when phrased as a natural instruction and producing a helpful answer. From a prompt engineering perspective, this means that the model's behaviour can be shaped by how the fine-tuning prompts are phrased. For example, including system-level directives ("You are an expert assistant. . . ") or exemplars in the fine-tuning data can later allow those patterns to be used at inference [1]. Open-source efforts like Dolly and OpenAssistant have released instruction-tuned models and datasets, confirming that even relatively small fine-tuned models can exhibit substantially improved instruction following. The implication for fine-tuning tasks is that one should carefully design the prompt format during training. In our context, if we fine-tune a model for lexical normalisation.

Chain-of-thought and reasoning prompts: Prompt engineering has also been used to enhance the reasoning capabilities of models during fine-tuning. The approach [102] demonstrated that by providing step-by-step explanations in few-shot prompts, models can solve math word problems and other reasoning tasks more accurately [103]. However, a risk is *contamination* or spurious reasoning – if the chain-of-thought is inconsistent or if training rationales are not carefully checked, the model might learn to generate convincing-sounding but flawed explanations. Ensuring high-quality rationales (or using techniques like verification to confirm the reasoning) is therefore important [31, 102].

A related prompting innovation is *least-to-most prompting*, which explicitly trains or prompts the model to tackle a complex problem by reducing it to simpler sub-problems [104]. For instance, an LLM can be prompted: "What sub-problem should we solve first?" and then later: "Now given that intermediate result, solve the next part," etc. This idea has been extended in fine-tuning by providing multi-turn dialogues in the training set where the model effectively talks itself through a problem. Soifer even introduced self-refine or self-ask strategies in fine-tuning: the model first generates a question about the input, then answers it, to simulate an internal QA reasoning process. These structured approaches (sometimes dubbed "/SoS-style" reductions, e.g., Skeleton-of-Thought prompting by Ning , 2023) help with complex tasks by dividing them, and can reduce latency by enabling parallel solution of subparts [104]. In practice, incorporating such patterns in fine-tuning data (or as a predetermined prompting template) can make the model more reliable on multi-step problems.

Self-consistency and data augmentation: Wang (2022) proposed a *self-consistency* decoding strategy for chain-of-thought prompts [33]. Instead of relying on a single reasoning path, one samples multiple diverse reasoning paths from the model and then aggregates their final answers (e.g., via majority vote) [103]. While self-consistency is an inference-time method, it has inspired training-time techniques as well. One can generate multiple reasoning paths for a given question (using the model itself or another) and include those as additional training examples, to encourage the model to consider different approaches to a problem. Another approach augments the fine-tuning dataset by perturbing prompts or using paraphrased instructions, to ensure the model's performance is robust to prompt wording. For example, if our task is lexical normalisation, we might augment training prompts with variations like "Which category does this item belong to?" vs "Classify the product described as…", so that the fine-tuned model is not overly sensitive to one prompt phrasing.

Data curation and quality: Prompt engineering for fine-tuning is not only about clever templates; it's also about selecting and crafting the right data. The quality of fine-tuning data is perhaps the most critical factor. There have been cases where seemingly impressive instruction-tuned models were later found to have test answers leaked into training (contamination), or to have overfit on popular benchmark questions. As an ethical and scientific imperative, researchers must audit fine-tuning datasets for such leakage and clean them. Moreover, balancing the dataset (ensuring a diverse coverage of topics and avoiding bias) can be seen as a form of prompt engineering at the dataset level. Recent surveys stress that a model fine-tuned on data drawn from only a narrow distribution of prompts will perform poorly on others, highlighting the need for breadth in instructions and contexts [105].

In conclusion, prompt engineering for fine-tuning revolves around shaping the model's behaviour via the training interface: using instructions, exemplars, and rationales in training prompts to imbue the desired capabilities. In our thesis, when fine-tuning the model (Chapter 4), we apply these insights by using instruction-style prompts for the training examples and by including chain-of-thought rationales for complex normalisation decisions (when available). We also take care to avoid training on any evaluation items (to prevent leakage) and to construct prompts that generalise, as opposed to overly specific ones that could bias the model.

3.5.2 Prompt Engineering for RAG

When using Retrieval Augmented Generation systems, prompt engineering extends beyond just phrasing the question – it involves orchestrating how the query interacts with

the retrieval component and how retrieved results are incorporated into the model's input. State-of-the-art RAG setups therefore include sophisticated prompt designs and pre-/post-processing techniques to maximise relevant information retrieval and usage. Here we outline several such techniques:

Query rewriting and expansion: A well-known challenge is that user queries may be underspecified or use terminology mismatched to the knowledge corpus. Prompt engineering can help by rewriting the query into a form more likely to retrieve good evidence. For example, a question "Does this product require an export license?" might be automatically expanded with context ("export license for [product type] according to [country] regulations") before hitting the retriever. Some systems use the LLM itself to generate a refined query: the model is prompted with, "Rewrite the query in a way a database of regulations would understand," leveraging its semantic knowledge [1]. This is often done in a zero-shot or few-shot manner as a first step. Query expansion can also add synonyms or related terms; classic IR techniques like adding WordNet synonyms have been supplanted by LLM-based expansion, which tends to be more context-aware. A recent approach called *BlendFilter* [106] employs a "query generation blending" strategy: it prompts the LLM to create multiple versions of the query, some focusing on different aspects of the question, and uses all of them to retrieve documents. By blending these results, it ensures comprehensive coverage of the information need. BlendFilter then applies a learned knowledge filtering step (also using the LLM in a verification role) to eliminate irrelevant passages. This two-stage prompt pipeline (query rephrasing in diverse ways, followed by LLM-based filtering of retrieved text) significantly improved accuracy on open-domain QA benchmarks, highlighting how careful prompt-driven query augmentation and result filtering can overcome retrieval noise.

Multi-hop reasoning and decomposed prompts: For questions that require reasoning across multiple documents (multi-hop queries), prompting techniques have been developed to break the query into subqueries. One line of work uses the LLM to decide the next search query based on what was found so far – sometimes called *self-ask* or recursive RAG. For instance, the system might prompt itself: "To answer the user's question, first I need to find X," then after retrieving X, ask the next question Y that fills the gap, and so on. This can be done with a fixed script or dynamically. Multi-Meta-RAG by Poliakov et al. (2024) [44] took a related but distinct approach: it uses LLM-extracted metadata to perform database-style filtering for multi-hop questions. Essentially, the LLM reads documents and extracts structured metadata (e.g. identifies the document's topic, entities, or provenance) and stores these in a database. Then, for a multi-hop query that might

require joining information, the system first filters documents via metadata (like "only consider documents from source A and about topic B"), reducing the search space [44]. This *prepare-then-retrieve* workflow improved performance on a multi-hop QA benchmark by ensuring that irrelevant branches are pruned early [44]. In terms of prompting, the LLM is prompted to generate a metadata summary for each document offline, and at query time it may also be prompted to suggest relevant metadata filters from the user query (e.g., "the question involves Person X and Person Y, filter for documents mentioning both"). Such meta-knowledge-driven prompts effectively inject a priori knowledge of what to retrieve.

Use of schema and structured data: RAG prompting need not be restricted to plain text. Many domains have accompanying structured data (tables, knowledge graphs) that can be harnessed. The DAQu framework stands for *Database-Augmented Query representation* [107]. It enriches query embeddings by linking the query to entries in a relational database and encoding those in the retrieval representation. While DAQu's core is in the retriever architecture, one can view it through a prompt engineering lens as well: the LLM (or a pre-processing step) effectively asks the database for relevant info about the query (like user profile, product attributes) and appends that to the query before retrieval [107]. For example, given a short query "import duty for HS 8471," a DAQu approach might retrieve from a trade database that HS 8471 is "Automatic data processing machines... (computers)" and enrich the query with those keywords. Experiments have shown that database-augmented queries significantly improve recall in scenarios where pure text similarity was insufficient [107]. Prompt-wise, implementing DAQu might involve writing a prompt for the LLM like: "Here is the user query. Here is the relevant database row about that query. Reformulate a detailed query using both." The output is then used for document retrieval.

Retrieval with metadata and knowledge summaries: Another cutting-edge idea is to generate semantic summaries or clusters of the knowledge base to guide retrieval. Mombaerts (2024) propose adding a *Meta Knowledge* layer to RAG [108]. They have the LLM pre-read each document (or cluster of documents) and produce: (a) a set of potential question–answer pairs that the document could answer, and (b) a concise "meta summary" of the document's contents [108]. Then, given a user query, instead of directly retrieving full documents, the system can retrieve relevant QA or summary pieces first (which is faster and more targeted), and then only fetch the full documents for those. In effect, the prompt to the LLM at query time becomes: "Given the user question, choose from these synthesized pieces of knowledge the ones that might contain the answer, then compose the

answer." By preparing queries with augmented information (like "synthetic QA matches") and even letting the LLM score or rank them, they achieved higher precision [108]. This workflow, described as "prepare-then-rewrite-then-retrieve-then-read", relies on prompt-driven generation of meta-knowledge that enriches both the retrieval and the final answer formulation. The cost is upfront: generating QAs and summaries for thousands of docs (though Mombaerts report doing this for 2000 research papers for under 20 with an API [108]). The benefit is a more informed retrieval process and an output that can cite specific facts (since the LLM was effectively "trained at runtime" on the summaries).

Negative example mining and re-ranking: When training retrievers or filtering retrieved passages, the use of hard negatives (documents that look relevant but are not) is essential. Prompt engineering can assist here by having the LLM generate challenging negative queries or passages. For instance, given a genuine QA pair, one could prompt the LLM: "Produce a confusing question that is similar to the original but has a different answer," and then ensure that the retriever does not retrieve the same documents for both [94]. For re-ranking, a strong cross-attention model, or even the LLM itself, can be used in a second stage to re-read the top-k results and sort them by relevance to the query. Many RAG pipelines now use a two-stage retrieve-then-rerank, where the reranker is often a fine-tuned MiniLM or an MPNet model. Alternatively, one can prompt an LLM: "Here are 5 passages. Which two are most likely to contain the answer?" and use its judgment (this is a form of LLM-as-reranker, akin to the REPLUG approach's iterative retrieval [94]).

In Table 3.2, we summarise some of these RAG prompting/meta techniques, comparing their applicability within the pipeline, whether they require a specific schema or external knowledge sources, how they handle ambiguous queries, and their computational overhead. The general trend is clear: simple one-shot RAG is being augmented with more intelligent querying and filtering strategies, often leveraging the LLM in auxiliary roles (rewriter, planner, verifier). In our Stage 2 (Chapter 5), we will incorporate some of these ideas by designing a two-step RAG system that first extracts key metadata via prompting (using an LLM to identify salient attributes in the query or partial answer) and then filters the retrieval based on that. We will also utilise LLM-based re-ranking to ensure the final context given to the generator is as relevant as possible. These enhancements aim to improve both the precision of retrieved information and the clarity of the model's responses, particularly for complex, domain-specific queries in our evaluation.

Despite significant advances in prompt engineering, most state-of-the-art techniques are designed for inference-time conditioning and treat prompts as linear, single-sequence inputs, offering limited support for structured or relational supervision during fine-tuning.

This creates a gap between prompt-based task specification and parameter-level adaptation, particularly for domain-specific problems with heterogeneous evidence sources. The proposed Set-of-Sequences approach, shown in Chapter 4, addresses this gap by enabling fine-tuning over structured collections of semantically related sequences, preserving contextual diversity while avoiding rigid input linearization. By bridging prompt engineering and efficient fine-tuning, this approach allows models to internalise prompt structure during training, leading to more robust task adaptation under constrained computational budgets.

Method	**Filtering/ Enhancement Stage**	**Use of Schema/ Structured Data**	**Robustness to Ambiguous Query**
Vanilla RAG Prompting	No special filtering; retrieve top-k based on query as-is	None (text only)	Low – relies on query matching; may retrieve irrelevant info if query is vague
Query Rewriting / Expansion	Pre-retrieval: LLM rephrases or expands query before retrieval	Not required (though can use ontology for expansion)	Medium – captures more aspects of query, mitigating ambiguity
BlendFilter [106]	Pre & post: multiple query variants + LLM-based result filtering	None (uses LLM's own knowledge)	High – multiple query aspects covered; LLM filters out off-target hits
Multi-Meta-RAG [44]	Pre-retrieval: metadata filtering via DB query	Yes – requires domain-specific metadata schema	High for multi-hop – narrows search to relevant sub-collection
DAQu [107]	Pre-retrieval: query embedding includes DB info	Yes – uses relational DB (e.g. user or item data)	Medium – enriches query context; handles underspecified queries better
Meta Knowledge RAG [108]	Pre- and mid-retrieval: offline meta QA pairs and summaries guide retrieval	Not a fixed schema, but requires generating structured metadata (Q&A, summary) for each doc	High – clarifies query via synthetic QA patterns; broader matching for vague queries
LLM-based Re-Ranking (Self-RAG)	Post-retrieval: LLM verifies or re-sorts retrieved passages	None (other than LLM's own comprehension)	Medium – can discard obviously irrelevant passages after initial retrieval

Table 3.2: Overview of advanced prompting and metadata techniques for Retrieval-Augmented Generation (RAG). This table compares strategies used at different stages of the RAG pipeline—before, during, or after retrieval—to improve the relevance and robustness of retrieved evidence. It highlights whether each method leverages structured schema information (e.g., metadata, databases) and evaluates how well it handles ambiguous or underspecified user queries. These methods are especially relevant to our Two-Step RAG system design in Chapter 5, where query disambiguation and domain-specific retrieval are critical.

3.5.3 The Use of Statistics for LLM Evaluation

Evaluating the performance of LLMs, especially on complex language tasks, is non-trivial. Traditional metrics (accuracy, BLEU, F1, etc.) provide a rough aggregate measure but often fail to capture nuances, and differences between models can be within statistical noise, as discussed in Chapter 2. Moreover, as LLM outputs become more variable (due to stochastic sampling, or differences in prompt phrasing), we need robust statistical tools to compare systems and to account for evaluator variability. Here we outline the state of the art in applying statistical methods to LLM evaluation:

Human-in-the-loop and hybrid evaluation: Despite advances in automatic metrics, human judgment remains the gold standard for many open-ended tasks (e.g., summarisation, open QA, dialogue helpfulness). However, human evaluation is expensive and introduces its own variability – different annotators may disagree, and the same annotator may score differently at different times [71]. Best practice today often uses a hybrid approach: automatic metrics to filter or quickly evaluate in certain dimensions, and targeted human evaluation for things like factual correctness or preference tests [109]. For example, one might use an automatic metric to ensure a generated text covers X% of reference facts, but then have humans rate fluency and correctness on a smaller sample. Another trend is using LLMs as proxies for humans (so-called "AI judges"), which we will discuss shortly [110]. In any case, whenever human ratings are involved, statistical analysis is needed to aggregate them reliably. Techniques like *MACE* [111] (estimating annotator competence and bias) or simply measuring inter-rater agreement are commonly applied. If multiple annotators score each output, one can use correlation measures (Spearman's ρ, Kendall's τ) to assess how well the rankings align, or Krippendorff's alpha for overall agreement. High agreement gives confidence in the scores; low agreement may indicate the task is ill-defined or the rubric unclear.

Bootstrap confidence intervals: A powerful, model-agnostic tool for understanding metric uncertainty is the bootstrap [74]. In NLP evaluation, we often treat the set of test examples as a sample from a larger population of possible examples. The bootstrap can simulate this by resampling with replacement from the test set and computing the metric (e.g., accuracy or ROUGE) many times. The variation across these resamples provides an empirical distribution for the metric, from which we can derive a confidence interval (CI). For instance, we might find that Model A's accuracy is 85% with a 95% CI of [80%, 89%]. If Model B's accuracy is 88% with CI [84%, 92%], these intervals overlap, indicating no statistically significant difference at 95% confidence. In contrast, if B's CI were [86%, 93%], mostly above A's, we'd conclude B is likely better. The bootstrap is particularly

useful when the test set is not extremely large and when we want to avoid assumptions of normality or independent errors. It has been used in recent LLM evaluations to give a sense of how reliable reported improvements are. Without CIs or significance testing, it is easy to overclaim significance from a 1-2 point improvement which might be within chance variation [112].

Inter-rater reliability and correlations: As mentioned, when human scores are used, one must assess agreement. Spearman's rank correlation is often used to compare an automatic metric to human rankings (to judge how well the metric correlates with "ground truth" human judgment). For example, if an embedding-based metric has Spearman $\rho = 0.45$ with human scores and BLEU has $\rho = 0.30$, we infer the embedding metric better aligns with human perception. Kendall's tau is another rank correlation that is more sensitive to small differences in ranking. These correlations can also have confidence intervals (computed via bootstrap over the dataset of scores) to tell if differences in correlation are significant. High-quality evaluations in recent papers include such analysis to back up claims like "Metric A is significantly more correlated with human judgment than Metric B on this task" [113]. Additionally, for categorical agreement (like "win/tie/lose" comparisons of model outputs by humans), statistics like Cohen's kappa or Fleiss' kappa are reported.

Controlling false discoveries: When many comparisons are made, such as evaluating a model on 10 different metrics or across 5 datasets, the chance of a false-positive finding increases [114]. The *False Discovery Rate* (FDR) procedures, like Benjamini–Hochberg (1995) [76], are used to adjust p-value thresholds when doing multiple significance tests. For example, if we test a model on 10 benchmarks, instead of requiring $p < 0.05$ for each (which yields up to 50% chance of a Type I error across them), we might use a tighter criterion such as $p < 0.005$ for each or adjust the p-values. Contemporary works, especially extensive evaluations like HELM [115], adopt such corrections to highlight which improvements are significant. In our thesis experiments, where applicable, we will employ multiple-comparison corrections to ensure we are not just cherry-picking statistically fluky results.

LLMs as evaluators – critiques: A notable recent development is using LLMs themselves to evaluate other LLMs' outputs (often called "GPT-as-a-judge"). For instance, given two answers, prompt GPT-4 with: "Which answer is better with respect to correctness and clarity?" This approach scales cheaply and often shows high agreement with average human preferences [116]. However, studies by Shen at al. [117] have pointed out pitfalls: LLM judges can exhibit positional bias (preferring the second answer, say), can

be inconsistent, and can fail on subtle criteria like factuality if both answers seem plausible. Moreover, if two models being evaluated share similar styles, an LLM judge might not discern differences that a human would care about (or, conversely, it might pick up on issues humans miss, such as mild toxicity). That research concluded that while GPT-4 as a judge correlates better with humans than automated metrics like BLEU, it is "not yet" a replacement for careful human evaluation [117]. It is best used with calibration (e.g., by prompting it with known-good versus known-bad answers to assess its accuracy) and in combination with human spot checks.

In summary, the state of the art in LLM evaluation is moving toward more rigorous, statistically grounded practices. Point estimates of accuracy or win-rate are no longer sufficient; researchers provide confidence intervals, use statistical models to analyse results, and carefully examine reliability of evaluators. Our work embraces this: Chapter 5 and especially Chapter 6 will introduce an evaluation framework (*IMMBA*) that leverages bootstrap resampling and Linear Mixed Models to assess model improvements across multiple metrics robustly. By doing so, we aim to address many of the concerns raised above – ensuring that any reported gains in our specialised LLM are statistically significant, practically meaningful, and not an artefact of evaluation bias.

Despite the growing sophistication of Large Language Models, their evaluation is still primarily dominated by point estimates of accuracy or aggregate scores computed over fixed test sets, often ignoring variability induced by prompts, sampling, and data heterogeneity. This reliance on single-run metrics limits statistical validity and hampers principled comparison across models and configurations. The proposed IMMBA approach, as shown in Chapters 4 and 5, addresses this gap by introducing a statistically grounded evaluation framework that combines mixed-effects modelling with bootstrap resampling, thereby explicitly capturing prompt-level and run-level variability.

3.6 Chapter Summary

This chapter critically reviewed the most relevant methods and findings that inform our research. We began by analysing Retrieval-Augmented Fine-Tuning (RAFT), highlighting its advantages over classic RAG and parameter-efficient tuning approaches in grounding knowledge and handling domain drift (Section 3.2). We then examined current strategies for lexical normalisation in Portuguese, noting the importance of language-specific pretraining, taxonomy-aware classification, and the trade-offs between model size and generalisation (Section 3.3). Lastly, we reviewed recent advances in prompt engineering and LLM evaluation (Section 3.5), focusing on techniques that improve model reasoning, retrieval precision, and statistical robustness. These insights shape the design of

our TRINITY framework. This chapter already identified key limitations in current approaches to domain adaptation, prompt engineering, and evaluation of Large Language Models, particularly under resource and reproducibility constraints. To address these gaps, the thesis introduces in the next chapter a cost-efficient, retrieval-aware fine-tuning strategy designed for controlled, domain-specific adaptation rather than large-scale instruction tuning. It further proposes a new fine-tuning paradigm that bridges the gap between inference-time prompt engineering and parameter-level adaptation by enabling structured, multi-sequence supervision during training. Finally, the evaluation framework proposes advances the state of the art in LLM assessment by replacing single-run, point-estimate metrics with statistically grounded inference that explicitly models prompt- and run-level variability. Together, these contributions form a coherent methodology for efficient, robust, and reproducible application of LLMs to specialised tasks.

Chapter 4

Proposed Model: TRINITY-LLM

Tri-Layered Intelligent Framework for LLMs: This research project was developed through the sequential integration of three interdependent studies, each addressing a critical stage in constructing, deploying, and evaluating domain-specific Large Language Models (LLMs). The first, *Efficient Fine-Tuning for Domain-Specific Language Models: SLIM-RAFT*, defines the methodological foundation for cost-effective fine-tuning and lexical normalisation of short-text data in Portuguese. The second, *Two-Step RAG for Metadata Filtering and LLM Evaluation Using Bootstrap Multivariate Mixed Model*, extends the framework by introducing an advanced Retrieval-Augmented Generation (RAG) architecture capable of contextual metadata extraction. The third, *IMMBA: Integrated Mixed Models with Bootstrap Analysis*, completes the pipeline with a statistically principled framework for multivariate evaluation that combines Linear Mixed Models (LMMs) with bootstrap resampling. Together, these studies form an integrated architecture for domain-adapted reasoning, retrieval, and evaluation.

4.1 Conceptual Overview

The overall research design follows a modular but cumulative logic. Each component builds upon the preceding one to address successive challenges encountered in the application of LLMs to structured classification tasks in the fiscal domain, particularly the lexical normalisation of short product descriptions for classification according to the *Nomenclatura Comum do Mercosul* (NCM) and postal code standardisation (CEP - *Código de Endereçamento Postal*).

The first stage, SLIM-RAFT, provides the fine-tuned base model through a simplified retrieval-augmented fine-tuning (RAFT) process. The second stage, Two-Step RAG, enhances retrieval precision by integrating dynamic metadata extraction and contextual filtering. Finally, the IMMBA framework applies statistical inference techniques to evalu-

ate model performance with reproducibility and interpretability. Figure 4.1 conceptually summarises this architecture as a unified research pipeline comprising training, retrieval, and evaluation stages. This will be explained in detail throughout the following sessions.

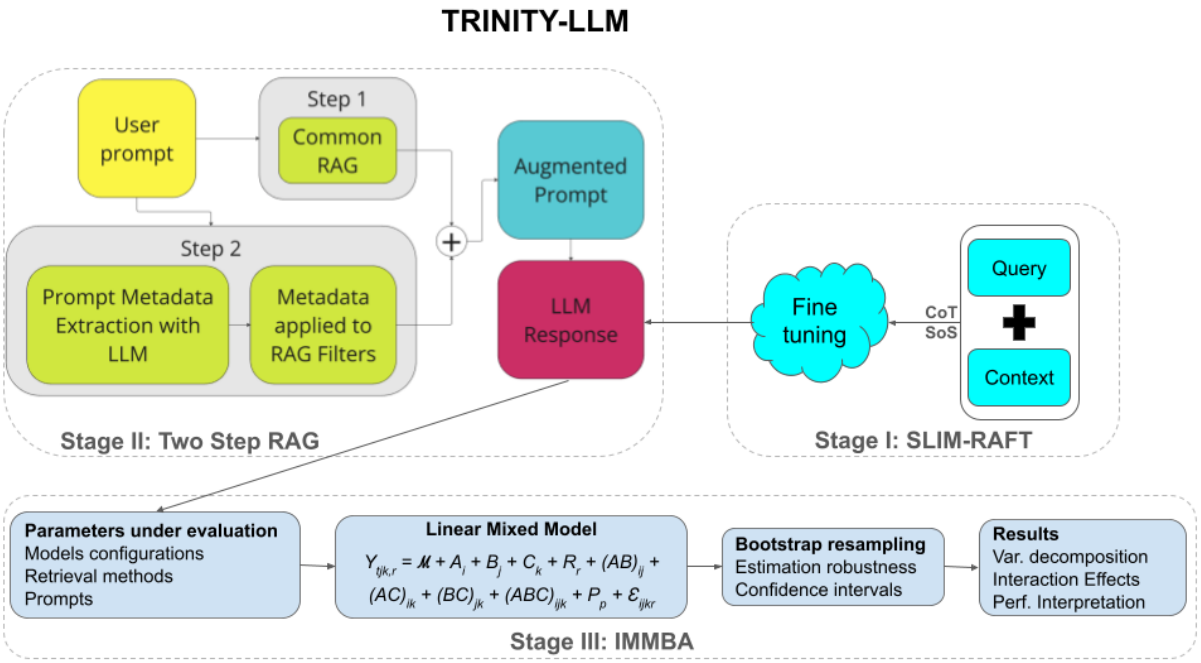

Figure 4.1: Tri-Layered Intelligent Framework for LLMs (TRINITY-LLM): Integrated research pipeline combining SLIM-RAFT fine-tuning, Two-Step RAG retrieval, and IMMBA statistical evaluation.

4.2 Stage I: Fine-Tuning and Lexical Normalisation (SLIM-RAFT)

This stage introduces the *Simplified Logical Intelligent Model – Retrieval-Augmented Fine-Tuning* (SLIM-RAFT), a lightweight framework conceived to overcome the linguistic and computational limitations observed in conventional fine-tuning pipelines for large language models (LLMs) when applied to Portuguese short-text classification. The SLIM-RAFT model extends the principles of the original *Retrieval-Augmented Fine-Tuning* (RAFT) approach while substantially reducing its complexity and cost. Its primary goal is to enable accurate lexical normalisation of short, noisy descriptions—such as those found in fiscal documents—through a simplified, reasoning-based fine-tuning strategy. Figure 4.2 depicts the overall architecture.

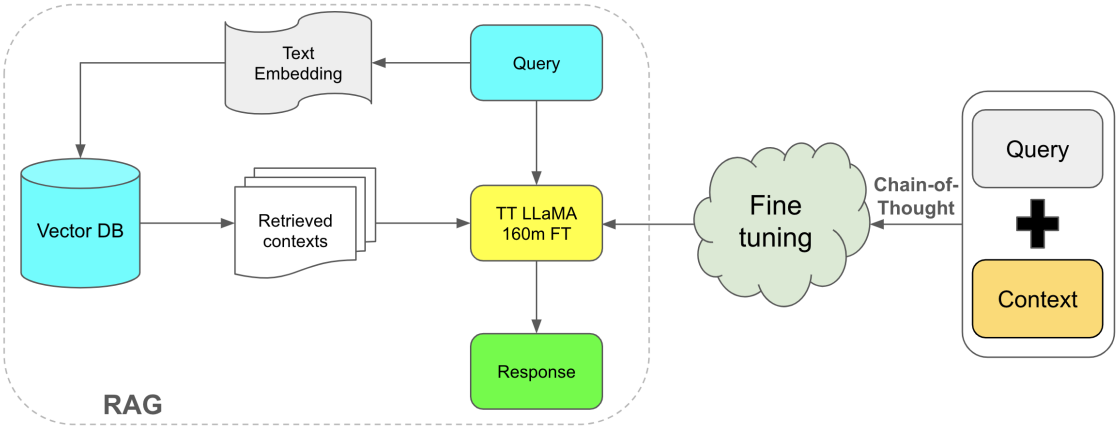

Figure 4.2: The Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning (SLIM-RAFT) diagram. Figure by the author.

4.2.1 Motivation and Context

The rapid dissemination of Generative AI systems has advanced natural-language technologies to unprecedented levels of sophistication [1, 2]. Proprietary multilingual LLMs such as ChatGPT exhibit strong reasoning abilities across modalities, yet they still depend heavily on prompt-engineering skills to reach their full potential [3]. Meanwhile, open-source models—exemplified by the LLaMA 3 family—allow local fine-tuning for privacy and flexibility but remain largely Anglocentric: approximately 90 % of their pre-training corpus is in English, leaving less than 10 % for other languages, including Portuguese, Spanish, and French. This linguistic imbalance constrains performance in technical and domain-specific contexts [4].

To mitigate these disparities, smaller and language-specific LLMs have emerged. Among them, the *TeenyTinyLLaMA* (TTL) model [24]—trained natively in Brazilian Portuguese and available in 460 M and 160 M-parameter versions—offers an optimal compromise between resource efficiency and representational power. The SLIM-RAFT framework builds directly upon this model, exploiting its compact architecture to achieve cost-effective domain adaptation through fine-tuning on curated Portuguese datasets.

4.2.2 From RAG to RAFT to SLIM-RAFT

Retrieval-Augmented Generation (RAG) [37, 41] addresses key weaknesses of LLMs' hallucination, factual drift, and opaque reasoning—by coupling language generation with external document retrieval. RAFT [43, 118] extends this paradigm by incorporating relevant knowledge during fine-tuning rather than inference, training the model to recognise and ignore distractor documents and to expose its reasoning chain. However, RAFT's dependency on another high-capacity LLM to construct a chain-of-thought (CoT)-based

training corpus renders it computationally expensive and, for most public institutions, infeasible.

SLIM-RAFT preserves RAFT's logical coherence but simplifies its data-generation stage. By replacing extensive document-level reasoning with compact, transitive logic sequences, it drastically reduces dataset-creation cost while retaining interpretability. The result is a fine-tuned model capable of structured inference without the overhead of full CoT supervision.

Application Domain: The NCM Challenge. The model is applied to the *Mercosur Common Nomenclature* (NCM) system—an eight-digit extension of the Harmonised System (HS) maintained by the World CustomsOrganisationn. NCM codes are mandatory for all Brazilian import, export, and domestic sales operations, demanding exceptional terminological precision. Given the high heterogeneity and abbreviation frequency in invoice descriptions, standard translation or embedding-only approaches (e.g., vanilla TeenyTinyLLaMA or ChatGPT) fail to capture the semantic relations required for accurate classification. The task thus exemplifies a real-world problem of lexical normalisation under noisy conditions.

Logical Simplification: Sequence-of-Sets (SoS) prompt The methodological novelty of SLIM-RAFT lies in the re-conceptualisation of chain-of-thought prompting into what we term a **Sequence-of-Sets (SoS)** reasoning paradigm. Instead of training on long narrative explanations, SoS represents reasoning as ordered logical relations among sets:

$$a \in A, \tag{4.1}$$

$$A \subseteq B, \tag{4.2}$$

$$\therefore a \in B. \tag{4.3}$$

This transitive pattern mirrors deductive inference while remaining computationally lightweight. During fine-tuning, prompts are constructed from minimal contextual fragments—each representing one set relation—followed by a concise question requiring the model to infer the hierarchical inclusion. This preserves the interpretability of CoT reasoning yet compresses training examples to a fraction of RAFT's typical size. Figure 4.3 shows the **Sequence-of-Sets (SoS)** logic in a composition of sets.

Prompt Engineering and Dataset Construction. The fine-tuning corpus was automatically generated through three iterative stages:

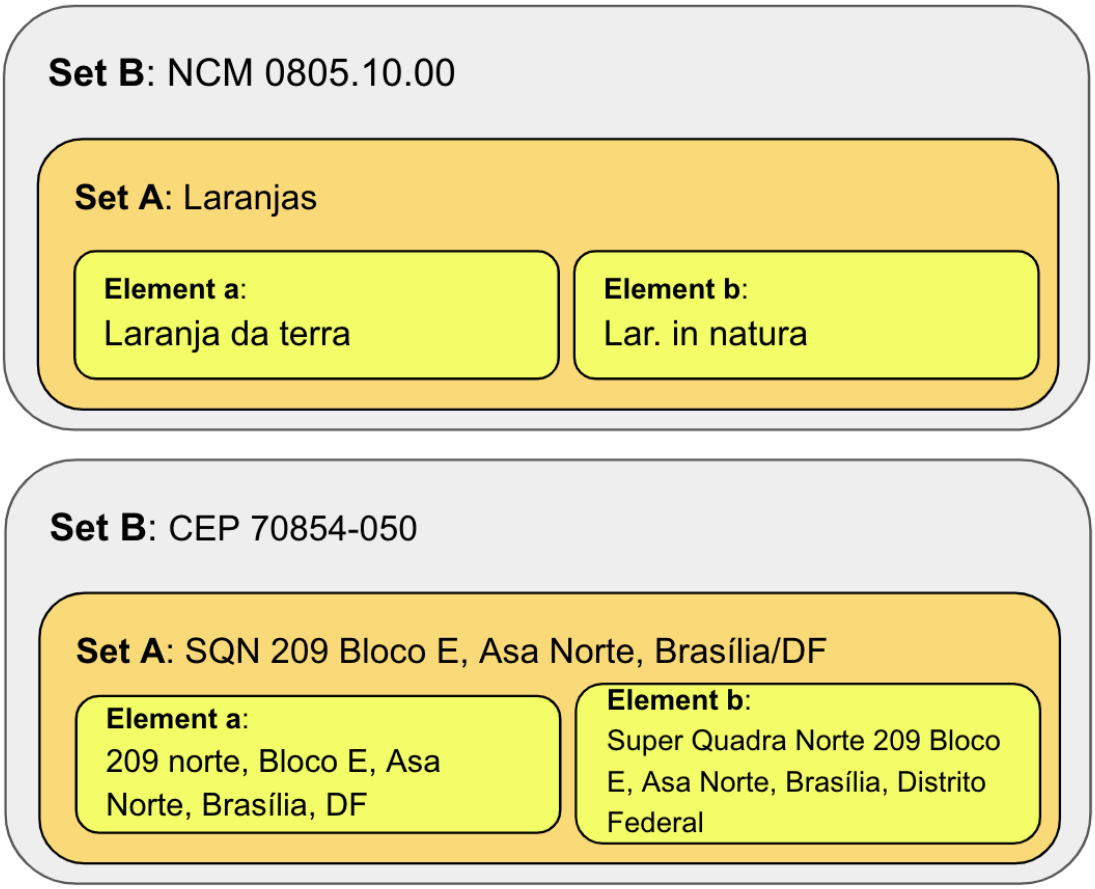

Figure 4.3: The Sequence-of-Sets (SoS) logic diagram.

1. Domain experts defined a small set of prototypical question–answer pairs (e.g., "What is the category of the product X?").
2. Variations of each pair were produced via ChatGPT 3.5 to enrich linguistic diversity.
3. A Python script populated the templates with real entries from the NCM database, yielding thousands of structured samples following the JSON-based SoS format.

The resulting number of records is estimated by

$$N = q \times v \times n, \tag{4.4}$$

where q is the number of expert-defined pairs, v the number of paraphrastic variations, and n the number of sampled NCM instances.

Two datasets were ultimately derived:

- **NCM-Dataset** (240 000 records) – used to train the TTL 460 M model with a standard LLaMA-2 Q/A format (`https://huggingface.co/datasets/yurifacanha/`

NCM-dataset);

- **NCM-RAFT-Small** (100 000 records) – built using the SoS prompting scheme to fine-tune TTL 160 M, with an overall generation cost below USD 100 (https://huggingface.co/datasets/yurifacanha/NCM-RAFT-samall).

An illustrative record in Portuguese is shown below (abbreviated for clarity):

```
[o codigo da categoria do VIN. PORT. ... 2014 750ML é: 22041010]
[a categoria 22041010 possui a seguinte descrição: Bebidas ... champanha]
Pergunta: O produto VIN. PORT. ... 2014 750ML está classificado em qual
categoria NCM?
Resposta: o produto ... possui categoria: Bebidas, líquidos alcoólicos ...
```

4.2.3 Source Model and Training Procedure.

Both TTL models (160 M and 460 M parameters) were fine-tuned using the `Transformers` library with low-rank adaptation (LoRA) [52]. The training was executed in Google Colab Pro using an A100 GPU over 11 hours, following the open-source notebook available at https://github.com/vinidiol/descmerc/blob/main/fine_tuning_teenytinyllama_RAFT.ipynb. The complete implementation and replication resources are publicly accessible in the SLIM-RAFT GitHub repository [1].

Results and Comparative Performance. The fine-tuned TTL 160 M model achieved a mean accuracy of 8.67 / 10 in NCM classification, compared with 4.5 / 10 for ChatGPT-4 under the same evaluation protocol. This confirms that the simplified SoS reasoning is sufficient to capture domain-specific logic even in extremely compact models. The approach thus demonstrates that reliable domain adaptation does not necessarily require billion-parameter architectures but can be attained through methodical dataset engineering and logical prompting.

Summary. In summary, SLIM-RAFT introduces an interpretable, resource-efficient fine-tuning methodology that logically extends RAFT while removing its cost-prohibitive dependencies. Its transitive SoS reasoning schema enables robust lexical normalisation of Portuguese short texts and establishes the foundation for the subsequent retrieval and statistical evaluation stages described in the following sections. The algorithm and details are presented in Section 4.3; the prompts and examples of use are in Section 4.3.5; and the evaluation method is in Section 4.4.

[1] https://github.com/yurifacanha/ncmrag

4.3 Stage II: Advanced Retrieval and Metadata Filtering (Two-Step RAG)

The Two-Step RAG method addresses a central limitation of conventional RAG systems: their heavy reliance on semantic similarity alone, which often yields off-target results under vague or underspecified prompts. Prior approaches such as Multi-Meta-RAG [44] apply metadata filtering sourced from large language models (LLMs) to improve document selection, but they tend to struggle with domain adaptability and schema rigidity. In contrast, Two-Step RAG organises contextual retrieval into two explicit phases (Figure 4.4):

1. **Common Context Retrieval:** a broad, unconstrained search using a standard RAG pipeline, ensuring high recall even when metadata are absent, noisy, or inconsistent. This step returns an initial set of candidates without prematurely restricting the search space.

2. **Metadata Extraction and Application:** an LLM extracts structured attributes directly from the original prompt (e.g., NCM, label, item, product). These attributes are then used to refine retrieval through targeted, post-hoc filtering over the semantic candidates. Unlike DAQu [107], which expands queries over relational databases, Two-Step RAG combines conventional retrieval with embedding-based semantic filtering guided by dynamically inferred metadata, thereby balancing breadth with precision.

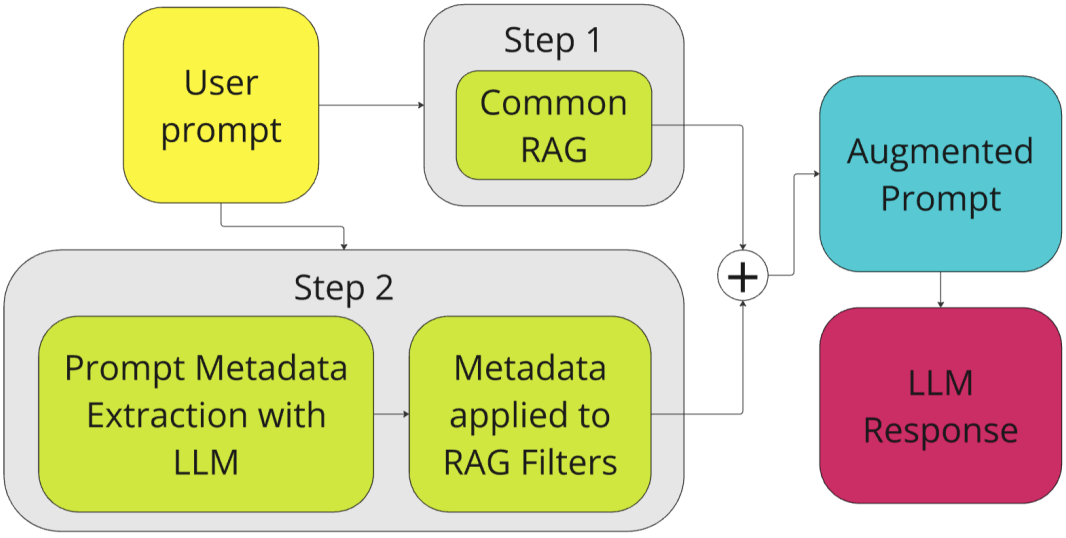

Figure 4.4: Illustration of the Two-Step RAG methodology [10].

The distinctive feature of Two-Step RAG is the deliberate separation of semantic retrieval and metadata filtering. While BlendFilter [106] blends query generation with heuristic filters and Meta Knowledge RAG [108] summarises clusters to guide retrieval, both rely on preset heuristics that may limit flexibility across domains. By contrast, Two-Step RAG first secures broad contextual coverage and only then applies metadata-driven constraints inferred by an LLM, enabling precise responses without sacrificing initial recall. This design directly addresses the recall–precision trade-off typical of RAG, improving accuracy while maintaining robustness to prompt ambiguity.

4.3.1 Mathematical Formulation of Two-Step RAG

The Two-Step RAG pipeline formalises retrieval as a two-phase process. Briefly, an embedding model first performs an unconstrained semantic search to produce initial candidates. Next, a language model extracts structured metadata from the prompt to generate a refined, augmented query that filters these candidates. Now we will detail it.

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ be a set of documents, where each document d_i contains textual information along with associated metadata to be extracted. Now we have a sequence of tokens representing a given Original Prompt:

$$P = (w_1, w_2, \ldots, w_m), \tag{4.5}$$

where w_j belongs to a vocabulary $\mathcal{V}$. To enable retrieval, we define an embedding function φ that maps a prompt to a d-dimensional vector space:

$$\varphi : P \to \mathbb{R}^d. \tag{4.6}$$

The knowledge base is indexed using document embeddings, forming a structured set:

$$\mathcal{K} = \{(\varphi(d_1), d_1), (\varphi(d_2), d_2), \ldots, (\varphi(d_N), d_N)\}. \tag{4.7}$$

4.3.2 Step 1: Common Retrieval (R_1)

The first stage, called **common retrieval** (R_1), searches for relevant documents based on the cosine similarity between the prompt embedding $\varphi(P)$ and the document embeddings stored in $\mathcal{K}$. The following equation defines the retrieval process:

$$R_1(P, \mathcal{K}) = \underset{d_i \in \mathcal{D}}{\operatorname{argmax}} \ \cos\big(\varphi(P), \varphi(d_i)\big). \tag{4.8}$$

This function returns a subset of relevant documents, denoted as $D_1 \subseteq \mathcal{D}$, based on semantic similarity. The subset size can be configured.

4.3.3 Step 2: Metadata Extraction (M) and Metadata-Driven Filtering (R_2)

In the second step, a language model extracts structured attributes from the prompt, known as **metadata extraction** (M). The following equation defines the extraction function:

$$M : P \to \{(k_1, v_1), (k_2, v_2), \ldots, (k_n, v_n)\}, \tag{4.9}$$

Where each pair (k_i, v_i) represents a metadata key and its corresponding value. The process is executed by a Large Language Model (LLM) as follows:

$$M(P) = \text{LLM}_{\text{metadata}}(P). \tag{4.10}$$

If no metadata are extracted, D_1 is used as-is. Otherwise, the metadata-driven filter is applied. The internal step, called metadata-driven filtering (R_2), refines the results from D_1 by applying the extracted metadata. The following equation defines the filtered document set D_2 as:

$$R_2(D_1, \mathcal{M}) = \left\{ d_i \in D_1 \;\middle|\; \forall (k, v) \in \mathcal{M},\ d_i[k] = v \right\}, \tag{4.11}$$

Here, $d_i[k]$ represents the metadata value k associated with document d_i. This filtering step ensures that only the most relevant documents are included according to the extracted metadata.

4.3.4 Augmented Prompt

Finally, the augmented prompt (A) combines the original prompt with the refined document set D_1, D_2 to provide a more precise context. The augmentation function is given by:

$$A(P, D_1, D_2) = \text{Concat}\big(P, \text{Context}(D_1, D_2)\big), \tag{4.12}$$

Where $\text{Context}(D_1, D_2)$ represents the structured formatting of the filtered documents. This model formalises the Two-Steps RAG mechanism, highlighting its ability to balance broad initial retrieval with refined filtering based on dynamically extracted metadata.

This approach improves retrieval precision without prematurely restricting the search space, thereby enabling an adaptive trade-off between generality and specificity in retrieval.

Algorithm 1 Two-Step RAG with Metadata Filtering

Require: Prompt P, Knowledge Base $\mathcal{K} = \{(\varphi(d_i), d_i)\}_{i=1}^{N}$

Ensure: Augmented Prompt $A(P, D_1, D_2)$

Compute prompt embedding: $\varphi(P)$

Step 1: Common Retrieval

Retrieve initial candidates by semantic similarity

$D_1 \leftarrow R_1(P, \mathcal{K}) = \arg\max_{d_i \in \mathcal{D}} \cos\big(\varphi(P), \varphi(d_i)\big)$

Step 2: Metadata Extraction

Extract metadata from the prompt

$\mathcal{M} \leftarrow M(P) = \mathrm{LLM}_{\mathrm{metadata}}(P)$

if $\mathcal{M} \neq \varnothing$ **then**

    Step 2: Metadata-Driven Filtering

    Filter documents in D_1 using metadata $\mathcal{M}$

    $D_2 \leftarrow R_2(D_1, \mathcal{M}) = \{d_i \in D_1 \mid \forall (k, v) \in \mathcal{M},\ d_i[k] = v\}$

else

    $D_2 \leftarrow D_1$

end if

Step 3: Prompt Augmentation

Combine P with filtered context

$A(P, D_1, D_2) \leftarrow \mathrm{Concat}(P, \mathrm{Context}(D_1, D_2))$

return Augmented Prompt $A(P, D_1, D_2)$

Two-Step RAG Comparative Advantages

Two-Step RAG contrasts with contemporaries such as Multi-Meta-RAG [44], BlendFilter [106], and DAQu [107]. Multi-Meta-RAG increases recall by issuing parallel, metadata-conditioned searches; BlendFilter reduces noise by fusing query generation and filters; DAQu relies on SQL over structured schemas. By comparison, Two-Step RAG explicitly decouples broad semantic retrieval (R_1) from subsequent metadata extraction (M) and targeted filtering (R_2), thereby avoiding early restriction of the search space while still enabling precise, schema-light refinement. Because metadata are inferred directly from natural-language prompts, the method avoids strong ontology dependencies and generalises across heterogeneous knowledge bases (e.g., NCM taxonomies). The modular operators $\varphi(P)$, $R_1(P, \mathcal{K})$, $M(P)$, and $R_2(D_1, \mathcal{M})$ also improve interpretability and extensibility. Table 4.1 summarises key differences and the contributions of Two-Step RAG.

Table 4.1: Comparison between Two-Step RAG and related methods.

Criterion	**Two-Step RAG**	**Multi-Meta-RAG** [44]	**BlendFilter** [106]	**DAQu** [107]	**Two-Step RAG Contributions**
Retrieval Steps	Two steps: broad (unfiltered) + refined with extracted metadata	Multiple parallel searches with different metadata	Query with embedded filters	Single SQL query	Avoids early bias by applying filters only after broad context retrieval
Metadata Extraction	Yes – dynamically via LLM	Yes – generally predefined	Implicit in the prompt	Not available	Flexible, does not rely on a fixed schema
Metadata Application	Post-retrieval, as filter	Pre-retrieval, defines multiple searches	During query generation	Pre-retrieval via SQL	Maintains high coverage before filtering, reducing loss of useful documents
Schema/Ontology Dependency	Low – automatic inference	Medium – requires defined attributes	High – rigid filters	High – requires relational mapping	Does not require specific data structure, enabling easier adaptation
Formal Model	Modular: R_1, M, R_2	Parallel: $\bigcup R_i$	Composite: $Q(P) \rightarrow R(Q)$	Structured: $Q_{SQL} \rightarrow R(Q)$	Clear and extensible structure, eases analysis and implementation
Domain Adaptability	High	Medium	Low to medium	Low	Works across domains with minimal configuration
Robustness to Ambiguous Prompts	High – broad retrieval ensures context	Medium – multiple variations help partially	Medium – depends on well-formed prompts	Low – structure requires well-defined query	Performs well even with vague or incomplete prompts
Primary Objective	Balance recall and precision with post-retrieval semantic filtering	Maximize recall via diverse queries	Reduce noise using integrated filters	Precise queries on structured databases	Ensures precision without compromising initial coverage

4.3.5 Case Study

To illustrate the method in practice, we consider NCM classification queries. The objective is to show how Two-Step RAG, when coupled with metadata filtering, retrieves and grounds the answer more precisely than a single-stage RAG baseline. We examine the prompt *"What is the official description of the NCM code for the goods with the following description: {product description}?"*. Notably, even with an explicit description, common RAG fails to recover the correct record, whereas Two-Step RAG succeeds with high reliability.

Original prompt

> *{Portuguese}* Qual a descrição oficial do código NCM para a mercadoria com a seguinte descrição: VASSOURAO CERPA PLAST FIO RIGIDO 370X63MM MAX
>
> *{English}* What is the official description of the NCM code for the goods with the following description: BROOM CERPA PLAST WIRE HARD 370X63MM MAX

Step 1 — Common Retrieval (R_1)

- *NCM: 96039000, Rótulo: vassouras, rodos, cabos e afins, Item: 39, Produto: ESCOV ACO B SEM CABO MAX MADEIRA 40700 1 UN.* English: NCM:

96039000, Label: brooms, squeegees, handles and the like, Item: 39, Product: BRUSH, STEEL, NO HANDLE, MAX, WOOD 40700 1 UN

- *NCM: 96039000, Rótulo: vassouras, rodos, cabos e afins, Item: 39, Produto: ESCOVA PARA BANHO.* English: Product: BATH BRUSH
- *NCM: 96039000, Rótulo: vassouras, rodos, cabos e afins, Item: 39, Produto: SM ESCOVA TANQUE OVAL CX 36UN.* English: Product: OVAL TANK BRUSH (BOX 36)
- *NCM: 96039000, Rótulo: vassouras, rodos, cabos e afins, Item: 39, Produto: RODO MAXIMA ADERENCIA MEDIO CONCEPT.* English: Product: SQUEEGEE, MEDIUM, MAX GRIP
- *NCM: 96039000, Rótulo: vassouras, rodos, cabos e afins, Item: 39, Produto: LIMPADOR SPONGUI.* English: Product: SPONGE CLEANER

Step 2 — Metadata Extraction (M)

The metadata extraction prompt is customised to the case study domain; other domains will require adaptation to their specific characteristics.

Extraction prompt

> *Tente identificar e extrair os possíveis metadados NCM, rótulo, Item e Produto da seguinte pergunta. Retorne um JSON. Se não encontrar a informação retorne null. Importante: rótulo é igual ao produto; se encontrar um deles, preencha automaticamente o outro com o conteúdo. Pergunta: {Original Prompt}.*
> *English: Identify and extract possible metadata from the following question, including NCM, Label, Item, and Product. Return a JSON. If no relevant information is found, return* `null`*. Important: the Label is equivalent to Product; if one is found, populate the other automatically. Question: {Original Prompt}.*

Step 2 — Metadata-Driven Filtering (R_2)

> NCM: 96039000 | Rótulo: vassouras, rodos, cabos e afins | Produto: VASSOURAO CERPA PLAST FIO RIGIDO 370X63MM MAX

Augmented Prompt

> *Você é um assistente especializado em responder de forma objetiva e clara com base em informações relevantes extraídas de uma base de conhecimento.*
> *English: You are an assistant specialised in objective, clear answers grounded in a knowledge base.*
>
> Relevant context (Portuguese):
>
> - NCM: 73269090 | Rótulo: Abraçadeiras | Produto: CABO PARA ROLO DE PINTURA 23 CM
> - NCM: 73261900 | Rótulo: Abraçadeiras | Produto: ESTICADOR PARA FIO FE
> - NCM: 73269090 | Rótulo: Abraçadeiras | Produto: CABO PARA ROLO DE PINTURA 10 CM

- NCM: 96039000 | Rótulo: vassouras, rodos, cabos e afins | Produto: RODO VAI E VEM MAIOR ADEREN. MEDIO
- NCM: 96039000 | Rótulo: vassouras, rodos, cabos e afins | Produto: RODO VAI E VEM MAIOR ADERENCIA PEQU
- NCM: 96039000 | Rótulo: vassouras, rodos, cabos e afins | Produto: VASSOURAO CERPA PLAST FIO RIGIDO 370X63MM MAX

Pergunta: "Qual a descrição oficial do código NCM para a mercadoria com a seguinte descrição: VASSOURAO CERPA PLAST FIO RIGIDO 370X63MM MAX." Caso o contexto não seja suficiente, indique explicitamente a insuficiência de dados. English: Question: "What is the official description of the NCM code for the merchandise with the following description: CERPA PLAST BROOM RIGID WIRE 370X63MM MAX." If context is insufficient, state this explicitly.

Applying Mathematical Layers

Let P be the tokenised prompt and $\varphi(P) \in \mathbb{R}^d$ its embedding. Consider a knowledge base $\mathcal{K}$ indexed by $\varphi(d_i)$ for $d_i \in \mathcal{D}$. The retrieval–filtering workflow proceeds as follows.

Step 1: Common Retrieval (R_1)

$$D_1 = R_1(P, \mathcal{K}) = \underset{d_i \in \mathcal{D}}{\operatorname{argmax}} \ \cos\big(\varphi(P), \varphi(d_i)\big).$$

This yields candidates purely by semantic similarity:

- d_1: NCM: 96039000, Produto: ESCOV ACO B SEM CABO MAX MADEIRA 40700 1 UN
- d_2: NCM: 96039000, Produto: ESCOVA PARA BANHO
- d_3: NCM: 96039000, Produto: SM ESCOVA TANQUE OVAL CX 36UN
- d_4: NCM: 96039000, Produto: RODO MAXIMA ADERENCIA MEDIO CONCEPT
- d_5: NCM: 96039000, Produto: LIMPADOR SPONGUI

Although all share the same NCM, none precisely matches the target product, revealing the limitations of R_1 alone.

Step 2: Metadata Extraction (M) Metadata are inferred from the prompt:

$$\mathcal{M} = M(P) = \mathrm{LLM}_{\mathrm{metadata}}(P) = (\mathit{NCM}, 96039000),$$

$$(\mathit{Produto}, \mathit{VASSOURAO\ CERPA\ PLAST\ FIO\ RIGIDO\ 370X63MM\ MAX})\}.$$

Filtering then applies:

$$D_2 = R_2(D_1, \mathcal{M}) = \{d_i \in D_1 \mid \forall (k, v) \in \mathcal{M},\ d_i[k] = v\}.$$

Only one candidate satisfies all constraints (grey-highlighted):

- NCM: 96039000, Produto: RODO VAI E VEM MAIOR ADERENCIA PEQU
- NCM: 96039000, Produto: RODO VAI E VEM MAIOR ADEREN. MEDIO
- NCM: 96039000, Produto: VASSOURAO CERPA PLAST FIO RIGIDO 370X63MM MAX

Augmented Prompt (A) The final augmented prompt is

$$A(P, D_1, D_2) = \text{Concat}\big(P, \text{Context}(D_1, D_2)\big),$$

which supplies the model with both the original question and the constrained, metadata-consistent evidence.

Research Questions on Two-Steps RAG Performance and Effectiveness

To evaluate the effectiveness of the Two-Step RAG, the reliability of responses generated by Large Language Models (LLMs) and the influence of key experimental variables, we define the following research questions that form the foundation of our study:

- **RQ1:** The Two-Steps RAG method improves document retrieval precision compared to a conventional RAG approach.
- **RQ2:** Applying extracted metadata in the second step of Two-Steps RAG reduces retrieval noise without compromising the diversity of results.
- **RQ3:** The performance of Two-Steps RAG varies depending on the domain of documents, being more effective in domains with well-structured semantic patterns (e.g., NCM, CEP).

4.4 Stage III: Statistical Evaluation and Robust Inference (IMMBA)

This stage introduces the *Integrated Mixed Models with Bootstrap Analysis* (IMMBA), a statistical framework designed to provide rigorous, interpretable, and reproducible evaluation of Large Language Models (LLMs). While preceding stages (Sections 4.2 and 4.3) focused on improving model training and retrieval, IMMBA addresses the equally critical

challenge of *evaluation*. Specifically, it decomposes observed performance variability into systematic and random sources, employing multivariate Linear Mixed Models (LMMs) combined with non-parametric bootstrap resampling to ensure robustness against distributional assumptions. This integration bridges classical psychometric analysis with modern LLM experimentation.

4.4.1 Linear Mixed Model Formulation

To quantify variability in LLM performance, IMMBA models evaluation scores through a multivariate Linear Mixed Model (LMM), allowing simultaneous estimation of fixed and random effects:

$$\begin{aligned} Y_{ijk,pr} = \mu + A_i + B_j + C_k + R_r + (AB)_{ij} + (AC)_{ik} \\ + (BC)_{jk} + (ABC)_{ijk} + P_p + \epsilon_{ijkr}, \end{aligned} \tag{4.13}$$

where:

- $Y_{ijk,pr}$ is the observed score under configuration (A_i, B_j, C_k, R_r) for prompt P_p.
- μ denotes the overall intercept (grand mean).
- A_i, B_j, C_k, and R_r correspond to fixed effects associated with the model architecture, decoding temperature, top-p sampling, and retrieval method, respectively.
- $(AB)_{ij}$, $(AC)_{ik}$, $(BC)_{jk}$, and $(ABC)_{ijk}$ represent two- and three-way interactions among fixed factors.
- P_p denotes a random effect capturing prompt-induced linguistic variability, modelled as $P_p \sim \mathcal{N}(0, \sigma_P^2)$.
- ϵ_{ijkr} is the residual error, assumed $\epsilon_{ijkr} \sim \mathcal{N}(0, \sigma_e^2)$.

This hierarchical formulation decomposes the total variance as follows:

$$\mathrm{Var}(Y_{ijk,pr}) = \sigma_f^2 + \sigma_P^2 + \sigma_e^2, \tag{4.14}$$

where σ_f^2 denotes variance explained by fixed effects (model, temperature, sampling, retrieval), σ_P^2 represents variability due to prompt phrasing, and σ_e^2 captures residual, unstructured noise. Such decomposition provides interpretable insight into the contribution of each experimental factor and the stochastic nature of linguistic variability—addressing a key limitation of traditional point-based metrics.

4.4.2 Bootstrap Resampling for Robust Estimation

To enhance statistical reliability and reduce dependence on parametric assumptions, IMMBA integrates non-parametric bootstrap resampling within the LMM fitting process. The bootstrap procedure unfolds as follows:

1. Generate B resamples of the original dataset by sampling with replacement.
2. Fit the LMM independently to each bootstrap sample, yielding empirical distributions of parameter estimates and variance components.
3. Derive confidence intervals, standard errors, and bias estimates from these empirical distributions.

This approach mitigates potential violations of normality and homoscedasticity in LLM score distributions. In this study, $B = 1000$ bootstrap iterations were employed—a balance between estimator stability and computational feasibility. The outcome is a robust inferential layer capable of quantifying uncertainty around each variance component and fixed-effect coefficient.

4.4.3 Evaluation Metrics and Scoring Protocol

Model responses were evaluated along four principal dimensions, each rated on a 0–10 ordinal scale by a robust LLM evaluator using a standardised protocol:

- **Quality** – assesses clarity, conciseness, and structural coherence.
- **Agreement** – measures semantic alignment with a reference or gold-standard answer.
- **Accuracy** – quantifies factual correctness based on official NCM taxonomies.
- **Hallucination** – penalises unsupported or fabricated content; lower scores indicate higher reliability.

Each dimension reflects a complementary aspect of generative model competence. Multiple raters independently scored responses, with disagreements resolved by adjudication. Inter-rater reliability was evaluated using Spearman's ρ, confirming high consistency among assessors. Figure 4.5 illustrates the multi-metric evaluation workflow integrated with the bootstrap pipeline.

The LMM and bootstrap estimations were applied separately to each of the four metrics, producing dimension-specific variance decompositions and cross-metric correlation structures. This multivariate treatment enables the identification of interdependencies, for example, whether higher Quality is systematically associated with reduced Hallucination variance, offering a psychometric view of LLM reliability.

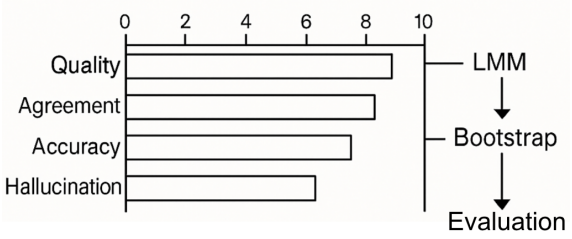

Figure 4.5: Flow of evaluation metrics through the bootstrap estimation process.

4.4.4 Experimental Design and Statistical Controls

The experimental design ensures rigorous control of fixed and random factors, facilitating valid inference across the full configuration space. The evaluation dataset derives from the Eleven Dataset [119], which contains Brazilian Electronic Invoice (NF-e) product descriptions annotated with their corresponding NCM codes. All analytical scripts and configuration files are openly available in the project repository[2], supporting full reproducibility.

The study tested the following research questions:

- **RQ1**: Fixed experimental factors (model architecture, retrieval method, temperature, and top-p) significantly influence evaluation scores.
- **RQ2**: Prompt phrasing induces measurable random variability across metrics.
- **RQ3**: Bootstrap resampling stabilises parameter estimates and confidence intervals, enhancing inference reliability.

Factorial Structure. A full-factorial design was implemented, encompassing:

- **Models** (A_i): five LLMs—`gpt-4o-mini`, `deepseek-chat`, `TeenyTinyLLaMA`, `gemini-2.0-flash`, and `Mistral-7B`;
- **Temperature** (B_j): three decoding temperatures (0.1, 1.0, 1.9);
- **Top-p** (C_k): three nucleus-sampling thresholds (0.1, 0.5, 0.9);
- **Retrieval Method** (R_r): conventional RAG versus metadata-enhanced Two-Step RAG.

The resulting $5 \times 3 \times 3 \times 2 = 90$ configurations were independently tested, each replicated across multiple prompt samples to ensure balanced randomisation and adequate statistical power.

[2] `https://github.com/pcbrom/immba`

Random Effects and Prompt Sampling. Prompts were treated as random effects within the LMM to capture stochastic linguistic influence. Each prompt represents a semantically distinct formulation of the same underlying query (e.g., different phrasings of an NCM classification request). This design choice isolates prompt-induced variance (σ_P^2) from model configuration effects, revealing the proportion of total uncertainty attributable to linguistic variability.

Sample Size and Power Analysis. Sample size determination followed conventional power analysis guidelines. Assuming a medium effect size ($f = 0.25$), significance level $\alpha = 0.05$, and statistical power $1 - \beta = 0.8$, a minimum of 196 replicates per condition was required, totalling approximately $21,000$ evaluated responses. This ensured robust parameter estimation and sensitivity to cross-factor interactions.

4.4.5 Rationale and Theoretical Implications

The integration of LMM and bootstrap estimation within IMMBA provides several methodological advances:

- **Hierarchical variance decomposition:** disentangles systematic (fixed) and linguistic (random) sources of variability, allowing transparent attribution of model performance fluctuations.
- **Interaction detection:** quantifies non-additive effects between parameters (e.g., temperature–retrieval interactions) that conventional mean-based metrics overlook.
- **Robust inference:** bootstrap distributions produce empirical confidence intervals resilient to non-normal, skewed, or heavy-tailed data.

In high-stakes applications—such as fiscal data classification or compliance auditing—these properties are essential. They provide a statistically grounded means to evaluate reliability, quantify uncertainty, and guide model selection with explicit confidence bounds. The resulting analytical framework establishes a reproducible bridge between linguistic evaluation and inferential statistics, setting a new standard for empirical LLM assessment.

4.5 Integrated Research Workflow

The three methodologies—SLIM-RAFT, Two-Step RAG, and IMMBA—compose a coherent pipeline for constructing, deploying, and evaluating LLMs in domain-specific contexts. The integrated workflow follows this sequence:

1. **Fine-tuning:** Training of the compact Portuguese-language LLM using the SLIM-RAFT method on the ELEVEN dataset to create a domain-specialised model capable of lexical normalisation and classification.
2. **Retrieval Enhancement:** Application of the Two-Step RAG methodology for context enrichment through dynamic metadata extraction and semantic filtering.
3. **Evaluation and Analysis:** Statistical decomposition and inference using the IMMBA framework to quantify fixed versus random sources of performance variation and assess reproducibility.

This triad of models forms a comprehensive framework that not only advances task performance but also establishes a transparent and statistically sound methodology for evaluating LLMs in sensitive, high-stakes applications such as fiscal classification and regulatory auditing.

4.6 Chapter Summary

This chapter introduced the **TRINITY-LLM** framework, which integrates three complementary components: Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning, Two-Step RAG for Advanced Retrieval and Metadata Filtering, and Integrated Mixed Models with Bootstrap Analysis. Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning provides a lightweight fine-tuning strategy using the *Sequence-of-Sets* (SoS) prompting method to enhance Portuguese text classification. Two-Step RAG for Advanced Retrieval and Metadata Filtering improves retrieval by combining broad semantic search with metadata-based filtering, balancing recall and precision. Finally, Integrated Mixed Models with Bootstrap Analysis establishes a rigorous statistical foundation through Linear Mixed Models and bootstrap resampling, decomposing variance into fixed, random, and residual components. Together, these stages form a unified and reproducible pipeline for efficient fine-tuning, adaptive retrieval, and robust evaluation of domain-specific language models.

The next Chapter will explain the details of the data characteristics and their preparation, as well as the architecture and implementation of the **TRINITY-LLM** model.

Chapter 5

Results and Discussion

This chapter reports the empirical evidence underpinning **TRINITY-LLM**, integrating results from (i) Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning for lexical normalisation and classification in the NCM and CEP tasks; (ii) Two-Step RAG for Advanced Retrieval and Metadata Filtering for metadata-guided retrieval; and (iii) Integrated Mixed Models with Bootstrap Analysis for statistically rigorous evaluation through bootstrap-enhanced Linear Mixed Models.

5.1 Experimental Setup and Datasets

We organise the scripts for clarity and reproducibility in the following repositories.

- ELEVEN Dataset: `https://github.com/vinidiol/descmerc`

  This is a database made up of short labeled texts composed of descriptions of goods extracted from electronic invoices. The descriptions were classified by specialist tax auditors and are presented following the Mercosur Common Nomenclature.

- SLIM-RAFT: `https://github.com/yurifacanha/ncmrag`

  The collection of codes developed for the execution of the first stage of this experiment, the classification of merchandise descriptions by NCM code.

- Two-Step RAG: `https://github.com/pcbrom/2-Steps-RAG`

  This repository provides all the scripts, prompts and data required to reproduce the experimental design described in the second stage, especially the RAG procedure.

- IMMBA: `https://github.com/pcbrom/immba`

  This repository reproduces the statistical results reported in the third stage, including confidence intervals and significance tests. It contains the scripts used to conduct bootstrap resampling, linear mixed models, and variance decomposition.

5.1.1 Models and Conditions

We evaluate four configurations for NCM classification:

- **Model 1**: TeenyTinyLLaMA (tiny460M) without domain fine-tuning (**TTL**).
- **Model 2**: ChatGPT 4.0 (**GPT**).
- **Model 3**: TeenyTinyLLaMA with NCM fine-tuning (**NCM-TTL**).
- **Model 4**: TinyLLaMA fine-tuned on the NCM dataset with the **SLIM-RAFT** methodology.

For the CEP task, we compare **SLIM-RAFT-CEP** with **TTL** and **GPT-4.o mini**. All models are evaluated on held-out questions (100 Q/A pairs per task) that were not seen during fine-tuning. Unless stated otherwise, *ChatGPT-4* conducts blind judging on a 0–10 scale; model identities are masked during assessment to mitigate evaluator bias.

5.1.2 Data, Protocol, and Reproducibility

For NCM classification, prompts are drawn from Brazilian Electronic Invoice (NF-e) item descriptions aligned to the Mercosur Common Nomenclature (NCM). The NCM data source in the ELEVEN Dataset [119]. For CEP, prompts refer to Brazilian postal code contexts. We follow a standard protocol: present 100 unseen questions to each model; record outputs; ask an external LLM judge for 0–10 scores (ChatGPT 4.0); compute means and dispersion. All CEP application code is available on GitHub[1]. Additional model- and metric-level analyses for Two-Step RAG and IMMBA use factorial designs and bootstrap procedures described in Sections 4.3 and 4.4.

5.2 Results for SLIM-RAFT (NCM and CEP)

5.2.1 NCM: Aggregate Performance and Illustrative Q/A

Aggregate Scores. The evaluation uses 100 unseen Q/As. Table 5.1 summarises average scores, Standard Deviation (SD), and ranges (minimum and maximum scores). **SLIM-RAFT** (Model 4) attains the highest mean (8.63) with SD 2.30, substantially outperforming **GPT 4.0** (4.5) and **TTL** (0.2).

Qualitative Comparison. Tables 5.2 and 5.3 provide side-by-side Q/A examples (Portuguese and English). **SLIM-RAFT** aligns more closely with the expected NCM description, evidencing domain reasoning benefits of prompting.

[1] `https://github.com/pcbrom/webist_cep`

Model	Mean	SD	Min.	Max.
TTL	0.2	0.98	0	5
NCM-TTL	4.71	3.53	0	10
GPT 4.0	4.5	1.39	0	5
SLIM-RAFT	8.63	2.30	0	10

Table 5.1: NCM Score results between the four models [9].

Q/A and Models	Sample question - Portuguese
Prompting	Qual a categoria NCM correta para o produto : LIMPADOR VEJA PERF. NAT PERF.SUAVE 1LT? Responda apenas a descrição da categoria. Caso não saiba responda a categoria que mais se aproxima. Não adicione qualquer qualquer outro comentário à sua resposta.
Expected Answear	Outros agentes orgânicos de superfície (exceto sabões); preparações tensoativas, preparações para lavagem (incluídas as preparações auxiliares para lavagem) e preparações para limpeza (inclusive multiuso e limpadores), mesmo contendo sabão, exceto os produtos descritos nos CEST 11.001.00, 11.004.00, 11.005.00 e 11.006.00; em embalagem de conteúdo inferior ou igual a 50 litros ou 50 kg.
TeenyTinyLLaMA	Não consigo responder com base nas informações fornecidas. No entanto, posso fornecer uma explicação geral de como categorizar produtos NCM, incluindo categorias específicas e suas respectivas categorias.
ChatGPT 4.0	Preparações para limpeza
TeenyTinyLLaMA FT PT NCM	A categoria NCM correta é : Agentes orgânicos de superfície (exceto sabões); preparações tensoativas, preparações para lavagem (incluindo as preparações auxiliares para lavagem) e preparações para limpeza, mesmo que contenham sabão, exceto as da posição 34.01. - Preparações acondicionadas para venda a retalho
SLIM-RAFT	o produto LIMPADOR VEJA PERF.NAT PERF. SUAVE 1LT possui categoria: Sabões, agentes orgânicos de superfície, preparações para lavagem, preparações lubrificantes, ceras artificiais, ceras preparadas, produtos de conservação e limpeza, velas e artigos semelhantes, massas ou pastas para modelar, "ceras para odontologia" e composições para odontologia à base de gesso. - Agentes orgânicos de superfície (exceto sabões); preparações tensoativas, preparações para lavagem (incluindo as preparações auxiliares para lavagem) e preparações para limpeza, mesmo que contenham sabão, exceto as da posição 34.01. - - - Preparações acondicionadas para venda a retalho

Table 5.2: Comparison of Q/A results of four models in Portuguese [9].

Q/A and Models	Sample question - English
Prompting	What is the correct NCM category for the product: CLEANER VEJA PERF. NAT PERF. SOFT 1LT? Respond only to the category description. If you don't know, answer the category that comes closest. Do not add any other comments to your answer.
Expected Answear	Other organic surface agents (except soaps); surfactant preparations, washing preparations (including auxiliary washing preparations) and cleaning preparations (including multipurpose preparations and cleaners), whether or not containing soap, except the products described in CEST 11.001.00, 11.004.00, 11.005.00 and 11.006.00; in packaging with a content of less than or equal to 50 liters or 50 kg.
TeenyTinyLLaMA	I am unable to respond based on the information provided. However, I can provide a general explanation of how to categorize NCM products, including specific categories and their respective categories.
ChatGPT 4.0	Cleaning preparations
TeenyTinyLLaMA FT PT NCM	The correct NCM category is: Organic surface agents (except soaps); surfactant preparations, washing preparations (including auxiliary washing preparations) and cleaning preparations, whether or not containing soap, other than those of heading 34.01. - Preparations packaged for retail sale
SLIM-RAFT	The CLEANER VEJA PERF.NAT PERF. SOFT 1LT product has a category: Soaps, organic surface agents, washing preparations, lubricating preparations, artificial waxes, prepared waxes, conservation and cleaning products, candles and similar articles, modeling masses or pastes, "dental waxes" and plaster-based dental compositions. - Organic surface agents (except soaps); surfactant preparations, washing preparations (including auxiliary washing preparations) and cleaning preparations, whether or not containing soap, other than those of heading 34.01. - - - Preparations put up for retail sale

Table 5.3: Comparison of Q/A results of four models in English [9].

5.2.2 CEP: Aggregate Results and Interpretation

The evaluation of SLIM-RAFT for the CEP code model was conducted in the same manner as previously described for SLIM-RAFT. One hundred new questions were asked of the model, and its results were compared with those of TTL and ChatGPT 4.0 mini, as shown in Table 5.4. The score achieved by the SLIM-RAFT model was comparable to that of OpenAI's 4.0 Mini model. We highlight that our model uses an open-source LLM with 160M parameters, while the 4.o mini is a state-of-the-art LLM with billions of parameters. Therefore, SLIM-RAFT was successfully applied in both domains tested.

Table 5.4 shows that **SLIM-RAFT-CEP** performs comparably to **GPT-4.o mini**, despite using a compact 160M-parameter model.

Model	Aver.	St. Dev.	Min.	Max.
TTL	0.39	0.063	0	2
GPT 4.o mini	1.90	0.067	0	3
SLIM-RAFT-CEP	1.92	0.039	0	2

Table 5.4: CEP Score results between the three models [9].

Due to resource constraints, we did not evaluate the fine-tuned TeenyTinyLlama model solely on the CEP. However, we believe this did not compromise the study, as the composite proposal with SLIM-RAFT was compared with the reference models (TTL and GPT).

Proof of generalisation: **SLIM-RAFT** generalises beyond NCM to CEP with competitive performance relative to a much larger proprietary model, underscoring the value of Sequence of Sets (SoS) prompting and targeted fine-tuning.

5.3 Results for Two-Step RAG

To demonstrate the applicability and flexibility of the Two-Step RAG method, we conducted the study with five different models:

- Mistral-7B (`Mistral-7B-Instruct-v0.3`);
- TeenyTinyLlama (SLIM-RAFT version: `TeenyTinyLlama-160m-NCM-ft`);
- Deepseek-chat (`deepseek-chat`);
- Gemini-2.0-flash (`gemini-2.0-flash`); and
- GPT-4.0-mini (`gpt-4o-mini-2024-07-18`).

The choice of models was based on a weighing of several factors, including model relevance, frequency of use in other studies, size, cost, and availability for inclusion in our application. The results and discussion of the experiment are presented in the following sections.

5.3.1 Research Questions and Outcomes

As summarised in Table 5.5, the study tested three key hypotheses to evaluate the performance of the Two-Step RAG approach in comparison to the conventional RAG method. Specifically, research question RQ1 predicted improved document retrieval precision with the Two-Step RAG, which was confirmed by higher average quality, precision and lower hallucination rates. Research question RQ2, regarding the reduction of retrieval noise without compromising result diversity through metadata application, was also confirmed. Meanwhile, research question RQ3 was partially confirmed, highlighting that the method's effectiveness varies with metadata quality across different document domains.

Research Questions	**Obtained Result**	**Interpretation**
RQ1: The 2-Step RAG improves document retrieval precision compared to conventional RAG.	Confirmed — higher average quality (5.30 vs. 2.73), better precision/accuracy (4.70 vs. 2.12) and lower hallucination (3.93 vs. 4.67).	Metadata-guided refinement enhances contextual retrieval.
RQ2: Applying metadata reduces retrieval noise without compromising diversity.	Confirmed — lower variability with 2-Step RAG, no significant loss of breadth.	Filters applied *after* broad retrieval maintain coverage.
RQ3: Performance gains vary by domain.	Partially confirmed — bigger in domains with structured metadata (NCM), but the difference was less pronounced in domains lacking clear metadata (CEP). Effectiveness depends on the quality and structure of metadata.	

Table 5.5: Tested research questions and corresponding results.

5.3.2 Descriptive Comparisons Across Models

Two-Step RAG improves Quality, Agreement, and Accuracy, while reducing Hallucination (metrics defined in Subsection 4.4.3) across models, with the most substantial gains

observed for *gpt-4o-mini-2024-07-18* and *deepseek-chat*. Tables 5.6 and 5.7 summarise means, spreads, and improvement ratios.

The descriptive statistics reveal the performance distribution of the evaluated models. The mean metric values indicate moderate performance: quality at 4.01, agreement at 3.98, accuracy at 3.29 and hallucination at 3.57. Significant variability, especially in agreement (SD 3.01) and hallucination (SD 3.45)[2], suggests inconsistent model performance, undermining reliability in NCM coding applications where precision and stability are vital to avoid errors. The 75th percentile quality score of 6 indicates many models perform below this level. A median accuracy of 2 further shows that many models do not classify codes as expected. Table 5.7 presents a consolidated comparison of models across all metrics, including means, SD, Coefficient of Variation (CV) and the improvement factor of Two-Step RAG over Common RAG. The results confirm that Two-Step RAG consistently outperforms the baseline in terms of quality, agreement, and accuracy. Improvements range from × 1.02 to × 4.54, with GPT-4o-mini and deepseek-chat showing the most stable and improved performance.

Two-Step RAG outperforms Common RAG across all metrics, showing higher quality (5.30 vs 2.73), agreement (5.56 vs 2.41) and accuracy (4.70 vs 2.12), with a lower hallucination rate (3.93 vs 4.67). This results in more accurate and consistent classifications, essential for reducing NCM classification discrepancies. The *gpt-4o-mini-2024-07-18* model achieved the best results with 2-Step RAG, making it ideal for practical application.

The model evaluation showed *gpt-4o-mini-2024-07-18* and *deepseek-chat* excelled with high-quality scores (4.58, 5.10), agreement (4.49, 4.94), accuracy (3.76, 4.41) and low hallucination rates (2.37, 2.33), indicating reliability for NCM coding. In contrast, *Mistral-7B-Instruct-v0.3* had lower quality (2.34), lower accuracy (1.91), and higher hallucinations (5.50), making it unsuitable for precise tasks.

TeenyTinyLlama-160m-NCM-ft is moderately performing with a quality of 3.73 and an accuracy of 2.77, but a high hallucination rate (5.36) may impact classification accuracy. *gemini-2.0-flash* offers a balanced profile, with a quality of 4.32, agreement of 4.47, accuracy of 3.58 and a low hallucination rate (2.31), indicating better stability.

Temperature and top-p parameters do not significantly affect quality, agreement, accuracy or hallucination metrics. However, a lower temperature (0.1) reduces hallucinations and ensures consistent output, making it suitable for high-reliability tasks such as NCM coding. The top-p variation shows no significant impact on these metrics.

The Two-Step RAG approach, paired with high-performance models like *gpt-4o-mini-2024-07-18* and *deepseek-chat*, best suits NCM classification, offering higher accuracy,

[2]Higher values for Quality, Agreement and Accuracy indicate better performance, while lower values for Hallucination are preferred.

Category	Model	Quality	Agreement	Accuracy	Hallucination
Best Performance	gpt-4o-mini	4.6 (2.4) [51.3]	4.5 (3.0) [66.2]	3.8 (2.8) [75.8]	2.4 (2.7) [114.8]
	deepseek-chat	5.1 (2.4) [46.6]	4.9 (3.2) [64.6]	4.4 (3.1) [69.9]	2.3 (2.7) [116.0]
Moderate Performance	TeenyTinyLlama	3.7 (2.2) [58.7]	3.7 (2.4) [64.9]	2.8 (1.9) [68.2]	5.5 (3.1) [55.9]
	gemini-2.0-flash	4.3 (2.3) [52.8]	4.5 (2.8) [63.4]	3.6 (2.8) [78.1]	2.3 (2.5) [109.5]
Lower Performance	Mistral-7B	2.3 (2.6) [111.1]	2.3 (2.9) [122.9]	1.9 (2.6) [134.3]	5.5 (4.2) [75.8]

Table 5.6: Descriptive performance statistics for each model across evaluation metrics, reported as mean score (Standard Deviation), and [Coefficient of Variation in %].

Model	Group	Quality	Agreement	Accuracy	Hallucination
Mistral-7B	2-Step	3.64 (2.81) [77.19]	3.73 (3.23) [86.60]	3.13 (2.99) [95.53]	4.44 (3.75) [84.46]
	Common	1.03 (1.47) [142.72]	0.95 (1.47) [154.74]	0.69 (1.13) [163.76]	6.56 (4.30) [65.55]
	Ratio	× 3.53	× 3.93	× 4.54	× 0.67
TeenyTinyLlama	2-Step	3.77 (2.21) [58.62]	3.74 (2.40) [64.17]	2.80 (1.90)	5.38 (3.08) [57.24]
	Common	3.68 (2.17) [58.96]	3.65 (2.39) [65.47]	2.73 (1.87) [68.50]	5.33 (3.10) [58.16]
	Ratio	× 1.02	× 1.02	× 1.03	× 1.01
deepseek-chat	2-Step	6.76 (2.07) [30.62]	7.20 (2.69) [37.36]	6.45 (2.90) [67.86]	3.01 (3.20) [106.31]
	Common	3.43 (1.20) [35.00]	2.68 (1.70) [63.43]	2.38 (1.52) [63.86]	1.65 (1.86) [112.72]
	Ratio	× 1.97	× 2.69	× 2.71	× 1.82
gemini-2.0-flash	2-Step	5.95 (2.01) [33.78]	6.48 (2.51) [38.73]	5.44 (2.71) [49.82]	3.33 (2.67) [80.18]
	Common	2.69 (1.01) [37.55]	2.45 (1.28) [53.24]	1.73 (1.22) [70.52]	1.29 (1.89) [146.51]
	Ratio	× 2.21	× 2.64	× 3.14	× 2.58
gpt-4o-mini	2-Step	6.35 (1.93) [30.39]	6.67 (2.54) [36.73]	5.69 (2.74) [48.15]	3.49 (2.88) [82.52]
	Common	2.81 (1.03) [36.65]	2.31 (1.30) [56.27]	1.83 (1.11) [60.65]	1.24 (1.98) [152.42]
	Ratio	× 2.26	× 2.89	× 3.11	× 2.81

Table 5.7: Mean (SD) [CV%] by model and RAG type, with 2-Step/Common ratios.

consistency and fewer errors. Avoid models like *Mistral-7B-Instruct-v0.3* due to high hallucination rates and low accuracy. Using conservative parameter settings, such as low temperature, leads to more reliable classifications. These findings support future enhancements of AI models for NCM coding, stressing the importance of consistency and minimal errors.

5.3.3 Scatter Matrix and Correlation Structure

The Scatter plots in Figure 5.1 indicate strong positive monotonic associations among Quality, Agreement, Accuracy ($\rho \in [0.955, 0.977]$), and weak negative correlations with Hallucination ($\rho \approx -0.28$).

The scatter matrix analysis reveals connections between quality, agreement, accuracy and hallucination, supporting Spearman's correlation analysis. Histograms show a multimodal distribution for quality and hallucination, influenced by RAG type. The Two-Step RAG yields higher values for these metrics, while the Common RAG results in more varied, lower values.

Unlike Pearson's, which assumes linearity, Spearman's correlation captures monotonic relationships without making that assumption. It focuses on rank consistency, making it ideal for complex datasets where relationships might not be linear and crucial for eval-

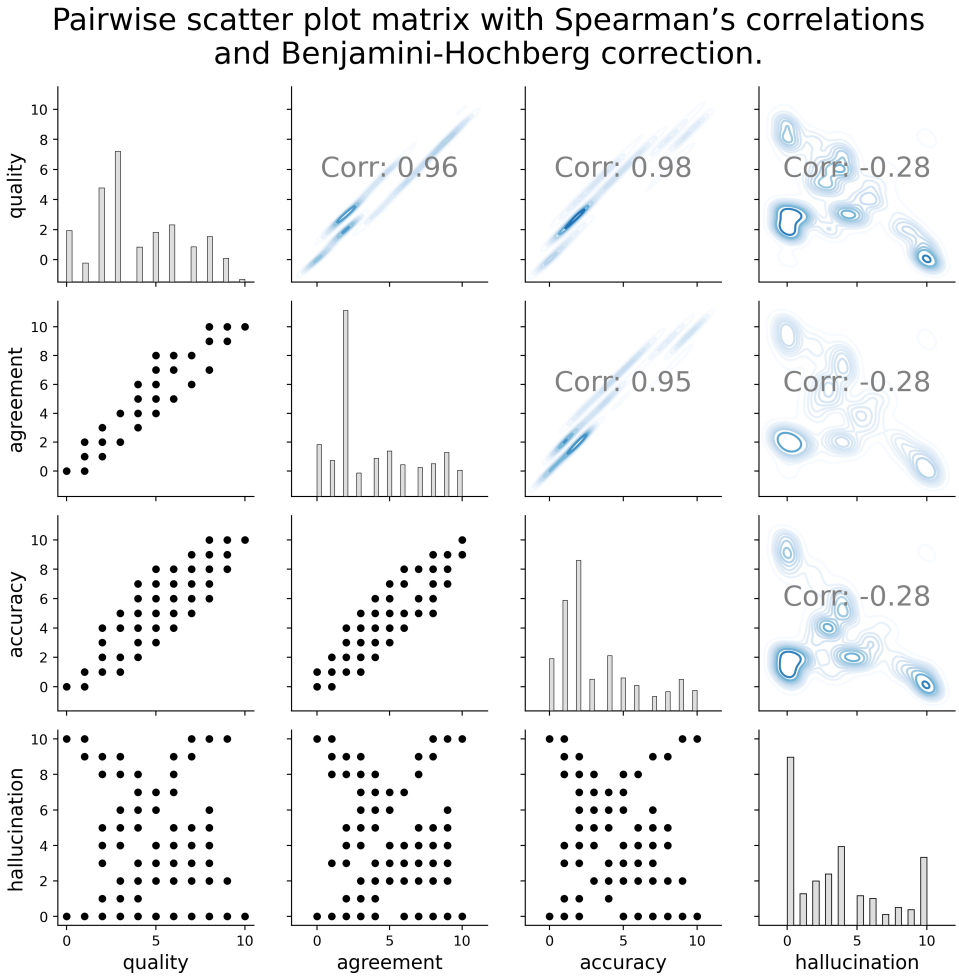

Figure 5.1: Pairwise scatter plot matrix with Spearman correlations between quality, agreement, accuracy and hallucination. Diagonal: histograms; lower triangle: scatter plots; upper triangle: density estimates. All correlations are significant (adjusted $p < 0.05$).

uating model performance where improvements aren't always proportional. The scatter and Spearman correlation matrices reveal the interdependence among quality, agreement, and accuracy, whereas hallucination shows weak inverse correlations. The Benjamini-Hochberg correction [76] boosts statistical robustness by reducing false positives, thereby strengthening the reliability of these relationships in assessing model performance.

Scatter plots in Figure 5.1 show a strong positive Spearman correlation between quality, agreement and accuracy, with coefficients of 0.959, 0.977 and 0.955. Diagonal bands highlight a robust monotonic link: higher quality corresponds to higher agreement and accuracy. Kernel density plots emphasise concentrated value regions, reinforcing these structured relationships. Hallucination presents a weak but consistent negative Spearman correlation with quality (-0.277), agreement (-0.284) and accuracy (-0.284). In other words, higher-quality, agreement, and accuracy scores are generally associated with lower

hallucination rates.

These results align with the Coefficient of Variation reported in Table 5.7, where Two-Step RAG not only improves performance but also reduces variability in models such as Deepseek-chat and GPT-4o-mini, indicating more stable behaviour across prompts. Figure 5.2 provides a complementary view of the distribution of these metrics across different RAG types, reinforcing the observed multimodal patterns.

Also, figure 5.2 contrasts distributions across models and RAG types, showing Two-Step RAG's higher central tendency and reduced dispersion in most cases. The Two-Step RAG yields higher values for these metrics, whereas the Common RAG yields more varied, lower values, as shown in Figure 5.1. Each combination of model and RAG type exhibits a distinct spread of values for Quality, Agreement, Accuracy and Hallucination. In contrast, when *Mistral-7B-Instruct-v0.3* is combined with the 2-Step RAG, the method yields markedly lower Quality and Accuracy, alongside higher Hallucination scores, yet with a narrower overall spread (the plot is "low and tight"). Meanwhile, under Common RAG, the same model shows more dispersed boxplots, implying greater unpredictability. Some samples achieve moderately better results, but a cluster of negative outliers also appears.

In other words, the figure underscores how each model responds differently to the choice of RAG technique. Even if textual results alone might obscure these nuances, the distributions here clearly reveal the relative stability (or variability) of the metrics across model–RAG pairings.

5.4 Results for IMMBA

This research employs descriptive statistics to evaluate quality, agreement, accuracy and hallucination across language models and RAG strategies. It scrutinises parameters such as temperature, top-p, and RAG type. The results provide insights into the reliability and consistency of models for classifying the Mercosur Common Nomenclature (NCM), demonstrate performance variations, and evaluate the suitability of each approach for this task. Using n = 1000 bootstrap replications, confirm that our confidence intervals and standard errors reflect the true variability in the dataset, leading to more robust and generalizable conclusions. It demonstrated that Mixed Linear Models could detect significant interactions between temperature, top-p and the retrieval method. A sampling human check was performed to validate this method; more details of this procedure will be presented in the next chapter.

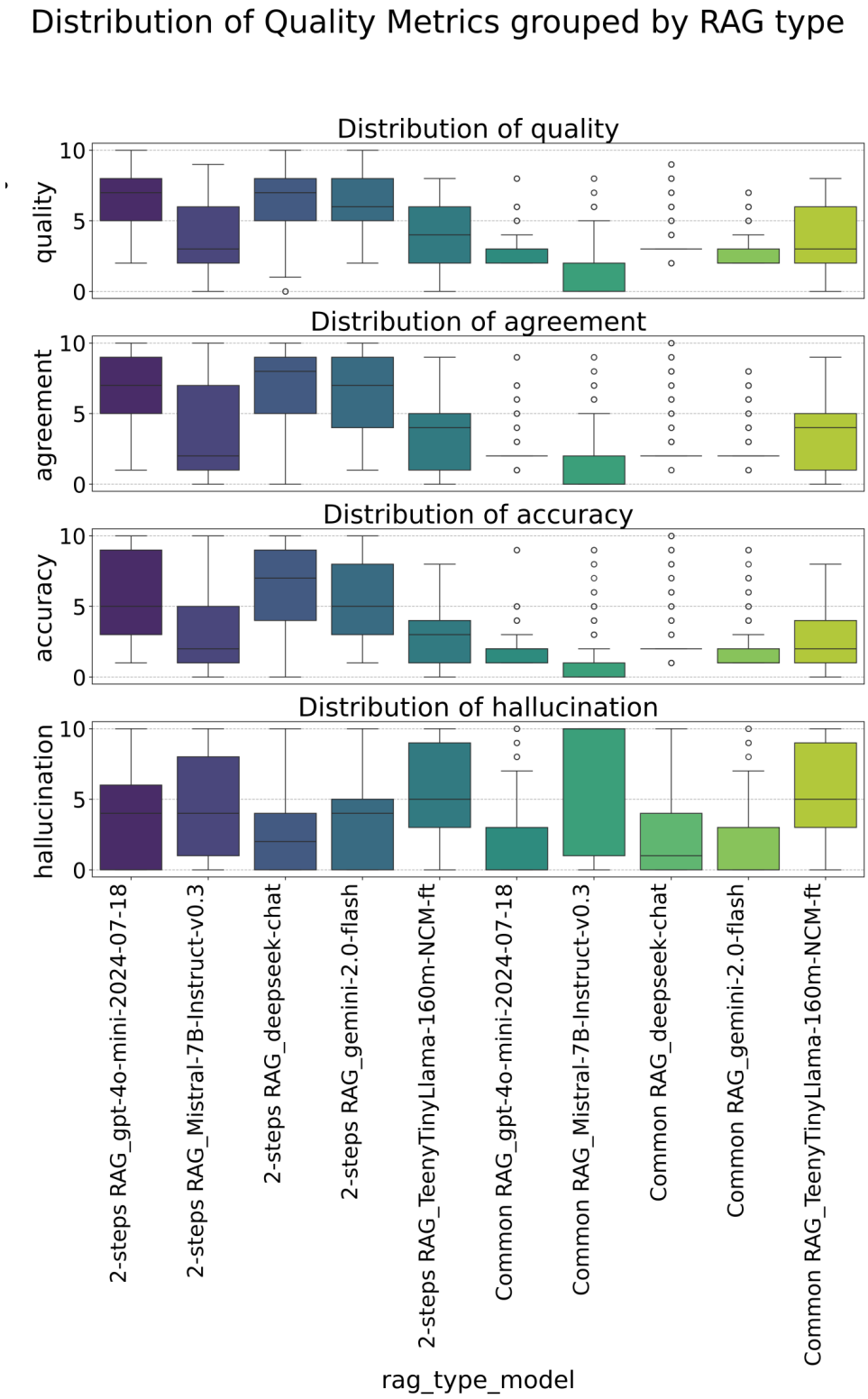

Figure 5.2: Distribution of quality, agreement, accuracy and hallucination across models and RAG type.

5.4.1 Bootstrap MLMM Estimates and Confidence Intervals

MLMM bootstrap confirms the superiority of Two-Step RAG (large negative coefficient for `Common RAG`), and highlights model-specific sensitivity to decoding parameters (e.g., `TeenyTinyLlama-160m-NCM-ft` interacting with temperature). In this section, we will delve into the results to demonstrate these findings.

Coefficient	**Coef Mean (SE)**	**95% CI (Lower, Upper)**
Intercept	3.9103 (0.0993)	(3.7219, 4.1138)
model[T.TTL-160m-NCM-ft]	2.0183 (0.1179)	(1.7984, 2.2549)
model[T.deepseek-chat]	1.3930 (0.0985)	(1.1971, 1.5855)
model[T.gemini-2.0-flash]	0.8835 (0.0924)	(0.6960, 1.0619)
model[T.gpt-4o-mini-2024-07-18]	0.9936 (0.0944)	(0.8092, 1.1784)
temp.[T.1.0]	0.1253 (0.0351)	(0.0579, 0.1973)
temp.[T.1.9]	0.3559 (0.0410)	(0.2789, 0.4427)
top p[T.0.5]	0.1665 (0.0370)	(0.0916, 0.2427)
top p[T.0.9]	0.3131 (0.0466)	(0.2238, 0.4020)
rag type[T.Common RAG]	-2.3191 (0.0836)	(-2.4858, -2.1500)
model[T.TTL-160m-NCM-ft]:temp.[T.1.0]	-0.5922 (0.0558)	(-0.7020, -0.4909)
model[T.deepseek-chat]:temp.[T.1.0]	-0.0451 (0.0452)	(-0.1363, 0.0396)
model[T.gemini-2.0-flash]:temp.[T.1.0]	-0.0565 (0.0434)	(-0.1522, 0.0258)
model[T.gpt-4o-mini-2024-07-18]:temp.[T.1.0]	-0.0638 (0.0415)	(-0.1514, 0.0132)

Table 5.8: Bootstrap Coefficients with Standard Errors (SE) and 95% Confidence Intervals (CI)

We used a bootstrap-based multivariate linear mixed model to enhance the robustness of coefficient estimates, accounting for both fixed and random effects. Since classical assumptions such as normality and homoscedasticity may not hold in our data, the bootstrap provides a nonparametric method for generating confidence intervals and standard errors by resampling from the data, independent of any assumed distribution. This method flexibly accounts for variance components, including those from prompts (σ_P^2) and residual experimental noise (σ^2). The results are presented in Table 5.8.

The models *gpt-4o-mini-2024-07-18* and *deepseek-chat* showed the highest regression coefficients, reinforcing their superior performance in the descriptive analysis. These models achieved the highest mean values for quality (4.58 and 5.10, respectively) and accuracy (3.76 and 4.41) while maintaining lower hallucination rates. The *TeenyTinyLlama-160m-NCM-ft* model also presented a high positive coefficient (+2.015), suggesting an above-average performance. However, interaction terms are sensitive to hyperparameter variations, leading to significant performance degradation at higher temperatures and top-p values. This aligns with its high hallucination rate (5.38), as observed in the descriptive statistics, Table 5.7.

The negative coefficient for the RAG type confirms that Common RAG significantly reduces response quality. This is consistent with the lower mean values associated with this method, including lower quality (2.73 vs. 5.30), lower accuracy (2.12 vs. 4.70) and higher hallucination rates (4.67 vs. 3.93). While Mistral-7B shows the largest relative gain in accuracy (×4.54), its absolute performance remains below the others, reinforcing the limitations of applying this architecture even under improved retrieval conditions. These results strongly support the Two-Step RAG approach, which improves consistency and classification accuracy in NCM applications.

Regarding hyperparameter influence, the regression analysis suggests that temperature and top-p have a minor positive effect, with coefficients close to zero. A slight increase in performance is observed for higher temperatures, but exploratory analysis suggests that lower temperatures effectively reduce hallucination rates, a factor for reliability. Similarly, while higher top-p values showed positive coefficients, exploratory results indicate no substantial improvement in classification quality, suggesting that these hyperparameters should be carefully adjusted rather than universally increased.

The interaction effects reveal a strong dependence of the *TeenyTinyLlama-160m-NCM-ft* model on hyperparameter settings. Specifically, increasing the temperature and the top-p significantly degrade its performance. In contrast, models such as *deepseek-chat* and *gpt-4o-mini-2024-07-18* did not show statistically significant interactions, suggesting their performance remains stable across different hyperparameter configurations.

5.4.2 Variance Decomposition

The variance decomposition analysis reveals that the total variation in model performance can be partitioned into three distinct components: variance from fixed effects, variance attributable to prompt design and residual variance. Precisely, the variance from fixed effects ($\sigma_f^2 = 2.1394$) reflects the contribution of the model's inherent structure.

Notably, the variance associated with prompts ($\sigma_P^2 = 0.6598$) indicates that the specific design and formulation of prompts shape model outputs. This finding suggests that even subtle variations in prompt construction can affect performance, thereby providing valuable insights into how input phrasing may modulate model behaviour.

Consequently, this analysis can serve as an indirect measure of prompt quality; however, it does not directly assess the clarity or semantic relevance of the prompts but rather the model's sensitivity to their formulation. Therefore, a comprehensive evaluation of prompt quality should be complemented with qualitative analyses and additional performance metrics.

Nevertheless, the most significant portion of the overall variance is due to residual effects (σ_e^2 = 6.4209), which likely arise from experimental noise and other unaccounted-for factors.

$$\text{Var}(Y_{ijk,pr}) = \sigma_f^2 + \sigma_P^2 + \sigma_e^2 \tag{5.1}$$

These results, shown in Table 5.9, indicate that both fixed effects and prompt design contribute to model performance, but the significant residual variance underscores the impact of experimental variability and model inconsistencies, providing empirical evidence that bootstrap-based estimation allows for reliable inference despite violations of classical assumptions. Unlike standard parametric methods, bootstrap resampling does not require normality, homoscedasticity or independence assumptions, making it a more appropriate method for assessing coefficient stability in complex real-world datasets such as the NCM code descriptions.

Component	**Symbol**	**Absolute Value**	**Percentage of Var(Y)**
Fixed Effects	σ_f^2	2.14	23.2%
Prompt Variability	σ_P^2	0.66	7.2%
Residual Error	σ_e^2	6.42	69.6%
Total	Var(Y)	9.22	100%

Table 5.9: Variance Decomposition in the IMMBA Model

Performance Interpretation: Decomposing variability clarifies the limitations of aggregate metrics in capturing LLM behaviour:

- Fixed effects explain 23.2% of total variance, highlighting the influence of model architecture and configuration.
- Prompt phrasing contributes 7.2% of variance, confirming the substantial role of linguistic formulation.
- Residual variability accounts for 69.6%, encompassing factors such as model stochasticity and scoring subjectivity.

These insights underscore the necessity of statistically grounded evaluation frameworks when comparing LLM configurations, particularly for high-stakes applications.

Summary. Fixed effects (model, retrieval, decoding) are material; prompt phrasing is non-negligible; residual variability remains the dominant component—justifying bootstrap-enhanced inference.

5.4.3 Research Questions and Bootstrap Estimation Robustness

Bootstrap resampling ($B = 1000$ iterations) was applied to all parameter estimates, yielding empirically derived confidence intervals and standard errors. The bootstrap-enhanced results exhibit:

- Increased stability of variance component estimates across resampled datasets.
- Robust detection of significant interaction terms, mitigating the influence of non-Gaussian output distributions.
- Narrower confidence intervals for fixed effects, supporting reliable attribution of observed performance differences.

This validates the utility of integrating bootstrap procedures into the evaluation pipeline, particularly for complex LLM outputs that exhibit stochastic behaviour and heteroscedasticity.

Table 5.8 provides the bootstrap-derived coefficient estimates, standard errors, and 95% confidence intervals for the fixed effects and their interactions within the IMMBA framework. These results directly support the research questions outlined in Section 4.4.4.

- First, **RQ1** is confirmed by the statistical significance of model, retrieval, and decoding parameters, which account for a substantial proportion of systematic variance.
- Second, **RQ2** is supported by robust detection of prompt-level random variability, as indicated by the consistent width of the confidence intervals across resamples.
- Finally, **RQ3** is supported by narrow bootstrap confidence intervals and stable coefficient estimates, demonstrating that resampling improves reliability and mitigates the risk of overfitting to a single dataset realisation.

Together, these findings confirm that IMMBA provides a statistically principled approach for decomposing and interpreting LLM performance variability. Traditional evaluation of Large Language Models is predominantly based on single-run experiments and aggregate point estimates, such as mean accuracy or F1 score, which implicitly assume fixed conditions and ignore sources of variability introduced by prompts, sampling strategies, and data heterogeneity. In contrast, the proposed IMMBA assessment method adopts a statistically principled perspective, combining mixed-effects modelling with bootstrap resampling to account for prompt-level and run-level variability explicitly. This shift enables uncertainty-aware performance estimates, more reliable model comparisons, and statistically defensible conclusions. As a result, IMMBA provides a more robust and reproducible assessment framework, particularly well-suited to complex, stochastic LLM

evaluation settings where traditional metrics may lead to overconfident or unstable conclusions.

5.5 Chapter Summary

This chapter presents the empirical results from implementing the Tri-Layered Intelligent Framework for LLMs, encompassing the Simplified Logical Intelligent Model Retrieval-Augmented Fine-Tuning, Two-Step RAG for Advanced Retrieval and Metadata Filtering, and Integrated Mixed Models with Bootstrap Analysis components. The experiments demonstrated that the proposed fine-tuning and retrieval strategies significantly enhanced performance in lexical normalisation tasks, particularly for NCM and CEP classification. The SLIM-RAFT model achieved superior accuracy compared to larger proprietary models, while the Two-Step RAG method improved contextual retrieval precision and reduced hallucination rates. Finally, the statistical evaluation conducted through IMMBA confirmed the robustness of the results, decomposing performance variability into interpretable components. Collectively, these findings validate the effectiveness and reproducibility of the proposed approach, establishing a solid foundation for the analytical discussion presented in the next chapter.

Chapter 6

Evaluation and Limitations

This chapter provides a comprehensive evaluation of the proposed Tri-Layered Intelligent Framework for LLMs framework, focusing on its methodological robustness, interpretability, and generalisability. Following the integrated presentation of findings and discussions in Chapter 5, this chapter assesses the strengths and constraints of the research design, highlighting both its practical implications and its inherent limitations. Emphasis is placed on three complementary perspectives: (i) the methodological soundness of the fine-tuning and retrieval strategies, (ii) the reliability of statistical inference derived from Integrated Mixed Models with Bootstrap Analysis, and (iii) the contextual applicability and ethical considerations for real-world deployment.

6.1 Results Evaluation

This section critically evaluates the methodological soundness and scientific integrity of the Tri-Layered Intelligent Framework for LLMs framework in light of the theoretical and empirical contributions developed across Chapters 3–5. Drawing from the experiments and analyses presented therein, the following subsections assess (i) methodological robustness, including sample size adequacy, bootstrap reliability, and rater agreement; (ii) the principal sources of experimental error, namely prompt randomness, dataset bias, and residual variance; (iii) the framework's comparative positioning against other state-of-the-art retrieval and evaluation paradigms, such as Multi-Meta-RAG and GraphRAG; and (iv) ethical and computational considerations related to reproducibility, transparency, and open-source implementation.

6.1.1 Methodological Soundness Evaluation

The methodological design of Tri-Layered Intelligent Framework for LLMs demonstrates a strong alignment with statistical best practices for LLM experimentation. The factorial configuration and sample-size planning presented in Chapter 5 ensured sufficient statistical power to detect main and interaction effects across experimental factors. Specifically, the study implemented a fully balanced $5 \times 3 \times 3 \times 2$ factorial design, encompassing 90 configurations with 196 replicates per condition, totalling over 21,000 evaluated responses. This design provides robust sensitivity to parameter-level differences while maintaining cross-model comparability.

The integration of the Integrated Mixed Models with Bootstrap Analysis framework introduced bootstrap-enhanced Linear Mixed Models (LMMs) to overcome the limitations of parametric inference. The use of 1,000 bootstrap replications ensured empirical confidence intervals that accurately reflect the stochastic nature of LLM outputs. By resampling the evaluation dataset, IMMBA derived stable variance component estimates, confirming the reliability of the fixed-effect coefficients (model, retrieval type, temperature, and top-p) and of the random prompt-level terms. This methodological choice directly addressed the non-normal, heteroscedastic distributions typical of LLM-generated data, thereby significantly improving inferential robustness.

Furthermore, inter-rater reliability was quantitatively assessed using Spearman's ρ, confirming high agreement across evaluators and validating the scoring consistency of the multi-dimensional evaluation scheme (Quality, Agreement, Accuracy, Hallucination). The bootstrapped LMM results revealed narrow confidence intervals around the estimated coefficients, demonstrating that the reported improvements—especially in Two-Step RAG's precision and SLIM-RAFT's lexical accuracy—were not artefacts of sample variability. Collectively, these methodological controls reinforce the internal validity and reproducibility of the experimental findings.

6.1.2 Sources of Error and Uncertainty

Despite the methodological rigour, several intrinsic and extrinsic sources of error must be acknowledged. Three primary contributors were identified: prompt randomness, dataset bias, and residual variance.

Prompt Randomness. As shown in Chapter 5, prompt phrasing accounted for approximately 7.2% of total performance variance (σ_P^2), underscoring the sensitivity of LLMs to linguistic formulation. Even minor lexical or syntactic changes in prompts produced measurable shifts in Quality and Accuracy metrics. Although this effect was statistically

isolated through random intercepts in the LMM formulation, it remains an inherent challenge in generative models. It motivates future research into prompt ensemble techniques and uncertainty quantification.

Dataset Bias. The experiments relied primarily on the ELEVEN Dataset, a structured dataset of Brazilian fiscal descriptions mapped to the Mercosur Common Nomenclature (NCM). While this dataset provides high-quality and linguistically rich data, its domain homogeneity limits external validity. Bias may arise from overrepresentation of certain commodity categories or from artefacts introduced during synthetic data generation (e.g., paraphrasing via ChatGPT 3.5 during SLIM-RAFT fine-tuning). Such biases can subtly influence learned associations, thereby constraining model generalisation to other corpora or tasks.

Residual Variance. The IMMBA variance decomposition revealed that 69.6% of total variability remained unexplained by fixed or random effects. This residual component likely stems from a combination of stochastic decoding effects, inter-rater subjectivity, and unobserved dependencies among model configurations. Although bootstrap estimation reduces estimation error, it cannot eliminate the inherent randomness in autoregressive generation. This residual variability highlights the importance of statistical replication and uncertainty reporting in future LLM evaluations.

6.1.3 Comparison with State-of-the-Art Frameworks

The Tri-Layered Intelligent Framework for LLMs framework distinguishes itself from contemporary retrieval and evaluation methodologies by combining interpretability, modularity, and statistical traceability. In particular, the Two-Step RAG method extends upon recent advances such as Multi-Meta-RAG [44] and GraphRAG [108], offering a more computationally tractable yet domain-flexible design.

Multi-Meta-RAG. Multi-Meta-RAG introduces metadata-aware retrieval through predefined schema-based filtering, effectively improving multi-hop reasoning and query disambiguation. However, it depends on explicit metadata structures and static ontologies. By contrast, Two-Step RAG dynamically extracts metadata via LLM inference, decoupling semantic retrieval from metadata conditioning. This makes it adaptable to domains lacking rigid schemas—such as fiscal data—and allows higher recall before selective refinement, reducing the risk of early retrieval bias.

GraphRAG and Meta Knowledge RAG. GraphRAG and its derivatives employ graph-based clustering and meta-summary generation to guide retrieval. While these methods achieve high precision in well-defined corpora, they incur substantial preprocessing overhead and rely on extensive document summarisation pipelines. The Two-Step RAG model, in contrast, achieves comparable accuracy improvements without such upfront costs, balancing computational feasibility with interpretability. Moreover, its integration within the Tri-Layered Intelligent Framework for LLMs architecture enables subsequent statistical evaluation through IMMBA, providing a full-cycle methodology absent from most RAG-based systems.

Evaluation Frameworks. Compared with statistical evaluation frameworks such as HELM [115], IMMBA advances LLM evaluation by hierarchically decomposing variance and integrating bootstrap inference. This approach yields transparent confidence intervals, enabling fine-grained attribution of uncertainty to prompt effects, model design, and retrieval strategies. Such multi-layered statistical interpretability is rare among current benchmarks, which often rely on aggregate scores or pairwise preference tests without formal variance modelling.

6.1.4 Ethical and Computational Considerations

The Tri-Layered Intelligent Framework for LLMs framework was conceived with ethical responsibility and computational sustainability at its core. All models employed are open-source (e.g., TeenyTinyLLaMA) or publicly accessible via documented APIs, ensuring full reproducibility and adherence to open-science principles. The fine-tuning and evaluation procedures were executed using freely available or low-cost computational resources (Google Colab A100 GPU sessions), illustrating that meaningful domain-specific advances in LLM research can be achieved without exclusive access to large-scale proprietary infrastructure.

From an ethical standpoint, the use of fiscal and address data from the ELEVEN Dataset adhered to anonymisation and public data compliance standards. No personally identifiable information (PII) was used. Moreover, all scripts, datasets, and evaluation protocols are openly published on GitHub (Section 5.1), enabling transparency, peer review, and community replication.

In terms of computational ethics, the compact model design (160M–460M parameters) substantially reduces environmental and financial costs relative to billion-scale architectures, aligning with current best practices in sustainable AI research. Nonetheless, limitations remain regarding potential model bias propagation and the interpretability of

fine-tuned reasoning traces, which should be further investigated to ensure fairness and accountability in real-world applications.

Reproducibility and Open Science. Each stage of the framework—fine-tuning, retrieval, and evaluation—was implemented with transparent codebases and fixed random seeds to facilitate deterministic replication. Data sampling and scoring protocols follow a fixed JSON schema, and evaluation metrics are reported alongside bootstrap-derived confidence intervals. These practices collectively satisfy reproducibility standards advocated in contemporary AI reproducibility checklists and ensure that subsequent researchers can extend or critique this work with methodological clarity.

6.2 Threats to Validity

Although rigorous statistical controls were applied throughout the research, several potential threats to validity should be considered when interpreting the results. Evaluation criteria involving human judgment, despite being operationalised through standardised JSON schemas, penalty rules, and fixed prompt structures, remain susceptible to evaluator bias. The performance metrics adopted—*quality*, *agreement*, *accuracy*, and *hallucination*—reflect human-judged aspects of text generation and may not fully capture end-user perceptions of relevance or usability.

Additionally, the experiments were conducted in a highly structured and linguistically constrained domain (the Mercosur Common Nomenclature). Consequently, results may not generalise to less formal or noisier text environments such as social media, conversational dialogue, or multilingual corpora. Although factorial design and bootstrap-based mixed modelling reduce statistical error, residual confounders—particularly those associated with prompt variation—cannot be excluded entirely. These considerations highlight the importance of replication in diverse linguistic and operational contexts to verify robustness across applications.

6.3 Limitations

Despite the overall success of the proposed framework, several limitations remain. These limitations are categorised into methodological, contextual, and evaluative aspects.

6.3.1 Methodological Constraints

- **Model size and scope:** The SLIM-RAFT implementation was developed using the TeenyTinyLLaMA 160M model, a lightweight architecture that limits general

reasoning capabilities. Larger models (e.g., TTL 460M or 1B+) could further improve comprehension and generalisation, particularly in open-domain scenarios.

- **Simplified reasoning structure:** The fine-tuning dataset employed a reduced form of chain-of-thought prompting, facilitating computational efficiency but restricting the complexity of reasoning patterns learned by the model. Expert supervision was necessary to ensure the quality of intermediate reasoning steps, underscoring the indispensable role of human expertise in the training process.
- **RAG dependency:** The efficacy of the Two-Step RAG method is directly influenced by metadata quality. In domains with inconsistent or poorly structured metadata, retrieval precision may decrease substantially, limiting the model's overall effectiveness.

6.3.2 Contextual Limitations

- **Domain specificity:** The research was conducted exclusively in the context of NCM classification and CEP mapping. Although the findings confirm strong performance within these tasks, cross-domain applicability has not been empirically established. Further studies are recommended to evaluate the adaptability of the framework to other knowledge-intensive or multilingual contexts.
- **Generalisation to unstructured data:** The experimental corpus comprised formalised product descriptions and fiscal records, which differ from unstructured or ambiguous text data. The performance of the proposed models on noisy, short, or user-generated text remains an open question.

6.3.3 Evaluation Constraints

- **Human scoring bias:** Despite using fixed scoring criteria and cross-rater adjudication, subjective judgment inevitably introduces variability. The incorporation of automated or hybrid evaluation systems could strengthen the objectivity and reproducibility of future analyses.
- **Residual variance:** The IMMBA framework revealed that approximately 70% of total variance remains unexplained, reflecting underlying model stochasticity and environmental noise. Future work should explore ensemble prompting or uncertainty quantification methods to further reduce residual variability.

6.3.4 Implications for Model Selection and Deployment

Empirical evidence supports the conclusion that smaller, well-tuned models can deliver reliable performance when paired with robust retrieval and evaluation strategies. Among the models tested, *gpt-4o-mini-2024-07-18* and *deepseek-chat* demonstrated superior quality and stability across configurations, making them appropriate candidates for real-world deployment in NCM-related systems. Conversely, models with higher variability and hallucination rates—such as *Mistral-7B-Instruct-v0.3*—should be avoided in mission-critical contexts.

From a practical standpoint, conservative decoding configurations (low temperature and moderate top-p) are recommended to ensure greater reproducibility in classification tasks. The results suggest that the Two-Step RAG methodology provides a scalable mechanism for metadata-driven retrieval that can enhance operational efficiency in contexts such as e-commerce cataloguing, fiscal verification, and legal document classification.

The methodological advances introduced by the Tri-Layered Intelligent Framework for LLMs framework extend beyond metric improvement. By reducing hallucination rates and enhancing interpretability, this approach strengthens the trustworthiness of AI systems in regulatory or high-stakes environments, where precision and transparency are essential. In legal, fiscal, or academic domains, the capacity to decompose model variability into measurable components (fixed, random, residual) offers a statistically defensible foundation for decision-making supported by language models.

6.3.5 Paths for Extension and Methodological Refinement

Future research can mitigate the identified limitations by progressively extending the proposed framework along complementary methodological and infrastructural dimensions. From a modelling perspective, scaling SLIM-RAFT to moderately larger architectures would allow more expressive reasoning patterns to be learned while preserving the efficiency principles that motivate the approach. Likewise, richer supervision strategies, such as selectively expanded chain-of-thought representations or hybrid human–machine curation, could enhance reasoning depth without incurring prohibitive annotation or computational costs. In retrieval-centric components, systematic investment in metadata standardisation and enrichment pipelines would directly strengthen Two-Step RAG performance, particularly in domains where structured attributes are currently sparse or noisy. Together, these directions suggest that many of the observed constraints are not intrinsic to the framework itself, but rather reflect deliberate design trade-offs aligned with feasibility and reproducibility goals.

Looking ahead, future studies should prioritise broader empirical validation across domains, languages, and data regimes to assess the generalisability of the TRINITY-LLM framework beyond fiscal and logistical contexts. Evaluations on noisier, user-generated, or semi-structured text would help clarify robustness boundaries, while cross-lingual or multilingual adaptations could test the portability of the proposed techniques. On the evaluation side, hybrid assessment strategies that combine structured human judgment with automated consistency checks and uncertainty-aware metrics may further reduce bias and unexplained variance. Finally, deeper integration of statistical evaluation methods, such as ensemble-based uncertainty estimation or hierarchical variance decomposition, offers a promising path toward a more stable and interpretable assessment of LLM. Collectively, these directions position the present work as a foundation for continued methodological refinement rather than a closed solution, reinforcing its relevance for both research and deployment-oriented settings.

6.4 Chapter Summary

This chapter critically evaluated the methodological integrity, limitations, and implications of the Tri-Layered Intelligent Framework for LLMs framework. It identified both the robustness and the boundaries of the proposed approach, highlighting the dependency on metadata quality, domain specificity, and subjective evaluation. Despite these constraints, the combination of fine-tuning simplification, retrieval optimisation, and bootstrap-based mixed modelling constitutes a reproducible, transparent, and scalable foundation for evaluating large language models. The insights drawn here set the stage for the synthesis of findings and future perspectives presented in Chapter 7.

Chapter 7

Conclusion

This thesis addressed the challenge of enhancing Large Language Models (LLMs) for Portuguese, focusing on the case study of lexical normalisation. The research emerged from the observation that most state-of-the-art LLMs are primarily trained on English corpora, resulting in reduced performance when applied to morphologically rich, syntactically flexible languages such as Portuguese. By analysing this gap, the study sought to contribute to the development of more accurate, efficient, and fair LLMs for non-English linguistic contexts.

The main goal was to improve LLMs' ability to handle informal, noisy, and heterogeneous Portuguese text, such as that found in fiscal, social media, and public administration data. To that end, a hybrid architecture was devised that integrated Retrieval-Augmented Generation (RAG) with fine-tuning, combining external knowledge retrieval with model adaptation. Tri-Layered Intelligent Framework for LLMs (TRINITY-LLM), the proposed model—tested under several configurations and evaluated through robust metrics—demonstrated significant performance gains in lexical normalisation and downstream classification tasks.

The research began with a comprehensive theoretical background that covered the evolution of LLMs, the principles of prompt engineering, and the mechanisms of fine-tuning. It also introduced the concept of retrieval-augmented learning as a bridge between static model training and dynamic information access. Based on this conceptual foundation, the work justified the need for hybrid architectures that can both generalise from large corpora and adapt to domain-specific realities.

From a methodological perspective, the study implemented a carefully designed experimental protocol. Multiple LLM variants were compared under controlled conditions, using metrics such as accuracy, agreement rate, and quality (semantic similarity). A bootstrap-based linear mixed model was applied to analyse performance variance and to assess the significance of prompt and model-level effects. This statistical framework enabled esti-

mation of reliability and robustness across experimental runs, reducing the influence of the stochastic variability typical of generative models. The results demonstrated that the devised hybrid model outperformed baseline LLMs by a statistically significant margin, achieving improvements in lexical normalisation accuracy and consistency, particularly in contexts with unstandardised orthography.

The main contributions of this research are threefold. First, it proposed a systematic approach to adapt LLMs for Portuguese through a hybrid Retrieval-Augmented Fine-Tuning (SLIM-RAFT and Two-Step RAG) methodology. Second, it introduced a reproducible evaluation framework based on mixed-effects modelling and bootstrap analysis, enabling more rigorous comparisons among stochastic generative models (IMMBA). Third, it provided empirical evidence that language-specific adaptations are essential to enhance the fairness, interpretability, and generalisation of LLMs across multilingual and domain-sensitive environments.

Beyond quantitative outcomes, the findings highlight the linguistic and societal relevance of extending AI research beyond the English-centric paradigm. In public administration, where lexical variability and informal expressions are frequent, improved normalisation supports more reliable document analysis and data integration. From a broader perspective, the thesis reinforces that linguistic inclusivity in LLM development is not merely a technical goal but also a matter of equity and accessibility in AI.

Nonetheless, the study has limitations that open directions for future research. The dataset, although representative of Brazilian Portuguese, could be further diversified with dialectal and temporal variations to strengthen model generalisation. Moreover, the experiments focused primarily on lexical normalisation; future work should extend the methodology to syntactic and semantic normalisation, exploring its application to speech, multimodal data, and code-mixed inputs. The retrieval mechanism may also benefit from adaptive relevance feedback and improved semantic indexing techniques. Finally, interpretability tools—such as attention visualisation and causal probing—could help elucidate how LLMs internalise linguistic regularities after fine-tuning.

In conclusion, this thesis contributes to advancing multilingual natural language processing by demonstrating that Portuguese can serve as a practical case study for enhancing LLM architectures. The hybrid approach proposed here, Tri-Layered Intelligent Framework for LLMs (TRINITY-LLM), combines retrieval and adaptation to address lexical variability, producing both theoretical insights and practical outcomes. By consolidating methodological rigour with linguistic sensitivity, this work provides a foundation for the next generation of LLMs capable of operating across languages with accuracy, transparency, and inclusivity.

Chapter 8

Related Academic Production

This section presents published academic works that are either directly or indirectly related to this research. We'll list key papers arising from this study (1, 4, 6, and 7) and others with a more indirect connection (2, 3, and 5) that still offer valuable insights; the research context and understanding shaped all of these works.

1. ELEVEN Data-Set: A Labeled Set of Descriptions of Goods Captured from Brazilian Electronic Invoices [119].
    - Authors: **Di Oliveira, Vinícius** and Weigang, Li and Rocha Filho, Geraldo Pereira.
    - Proceedings: International Conference on Web Information Systems and Technologies - WEBIST (QUALIS A4)
    - 2022
    - DOI: 10.5220/0011524800003318
2. SCAN-NF: A CNN-based System for the Classification of Electronic Invoices through Short-text Product Description [120].
    - Authors: Kieckbusch, Diego S and Geraldo Filho, PR and **Di Oliveira, Vinícius** and Weigang, Li.
    - Proceedings: International Conference on Web Information Systems and Technologies - WEBIST (QUALIS A4)
    - 2021
    - DOI: 10.5220/0010715200003058
3. Visual analysis of electronic invoices to identify suspicious cases of tax frauds [121].

- Authors: Marinho, Mayara C and **Di Oliveira, Vinicius** and Neto, Sérgio APB and Weigang, Li and Borges, Vinícius RP.
- Proceedings: International Conference on Information Technology & Systems CIT (QUALIS B1)
- Organization: Springer
- 2022
- DOI: 10.1007/978303096293718
- Award: First place in the CTCCSI competition at the Brazilian Symposium on Information Systems - SBSI 2024.

4. SLIM-RAFT: A Novel Fine-Tuning Approach to Improve Cross-Linguistic Performance for Mercosur Common Nomenclature [9].

   - Authors: **Di Oliveira, Vinícius** and Bezerra, Yuri Façanha and Weigang, Li and Brom, Pedro Carvalho and Celestino, Victor Rafael R.
   - Proceedings: accepted to International Conference on Web Information Systems and Technologies - WEBIST (QUALIS A4)
   - 2024
   - DOI: 10.5220/0012943400003825
   - Award: Nominee for Best Student Paper on WEBIST 2024

5. Implementing AI for Enhanced Public Services Gov.br: A Methodology for the Brazilian Federal Government [122].

   - Authors: Melo, Maisa and dos Reis, Silvia Araújo and **Di Oliveira, Vinícius** and Faria, Allan and de Lima, Ricardo and Li, Weigang and Salm Junior, Jose Francisco and de Moraes Souza, Joao Gabriel and Freitas, Vérica and Brom, Pedro and Kimura, Herbert and Cajueiro, Daniel and da Silva, Gladston Luiz and Celestino, Victor.
   - Proceedings: accepted to International Conference on Web Information Systems and Technologies - WEBIST (QUALIS A4)
   - 2024
   - DOI: 10.5220/0012997000003825
   - Award: Nominee for Best Industrial Paper on WEBIST 2024

6. IMMBA: Integrated Mixed Models with Bootstrap Analysis - A Statistical Framework for Robust LLM Evaluation [11].

- Authors: **Di Oliveira, Vinícius** and Brom, Pedro Carvalho, and Weigang, Li.
- Proceedings: accepted to International Conference on Web Information Systems and Technologies - WEBIST (QUALIS A4)
- 2025
- DOI: 10.5220/0013819400003985

7. Two-Step RAG for Metadata Filtering and Statistical LLM Evaluation: Integrating Prompt-Aware Retrieval and Bootstrap Mixed Models [10].

    - Authors: **Di Oliveira, Vinícius** and Brom, Pedro Carvalho, and Weigang, Li.
    - Journal: IEEE Latin America Transactions (QUALIS A4)
    - 2025
    - DOI: 10.1109/TLA.2025.11231222

References

[1] Schulhoff, Sander, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, *et al.*: *The Prompt Report: A Systematic Survey of Prompting Techniques*. arXiv preprint arXiv:2406.06608, 2024. 1, 60, 62, 72

[2] Radosavovic, Ilija, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik: *Humanoid locomotion as next token prediction*. Advances in neural information processing systems, 37:79307–79324, 2024. 1, 72

[3] Weigang, Li, Liriam Michi Enamoto, Denise Leyi Li, and Geraldo Pereira Rocha Filho: *New directions for artificial intelligence: human, machine, biological, and quantum intelligence*. Frontiers of Information Technology & Electronic Engineering, 23(6):984–990, 2022. 1, 13, 72

[4] Souza, Fábio, Rodrigo Nogueira, and Roberto Lotufo: *BERTimbau: pretrained BERT models for Brazilian Portuguese*. In *Intelligent Systems: 9th Brazilian Conference, BRACIS 2020, Rio Grande, Brazil, October 20–23, 2020, Proceedings, Part I 9*, pages 403–417. Springer, 2020. 1, 12, 47, 56, 72

[5] Sato, Júlia, Helena Caseli, and Lucia Specia: *Choosing What to Mask: More Informed Masking for Multimodal Machine Translation*. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 244–253, 2023. 1

[6] WCO: *List of Contracting Parties to the HS Convention and countries using the HS - World Customs Organization*, 2024. Published in www.wcoomd.org Accessed on Jun 6th, 2024. 1

[7] MERCOSUR: *MERCOSUR - Consultas à Nomenclatura Comum e à Tarifa Externa*, 2024. `https://www.mercosur.int/pt-br/politica-comercial/ncm/`, `https://www.mercosur.int/pt-br/politica-comercial/ncm/` Accessed on Jun 4th, 2024. 1, 2, 3

[8] Valença, Paulo Ricardo Mendes *et al.*: *Essays on foreign trade, labor, innovation and environment*. Universidade Católica de Brasília, 2023. 1

[9] Di Oliveira, Vinícius, Yuri Façanha Bezerra, Li Weigang, Pedro Carvalho Brom, and Victor Rafael R Celestino: *SLIM-RAFT: A Novel Fine-Tuning Approach to Improve Cross-Linguistic Performance for Mercosur Common Nomenclature*. In *Proceedings*

of the 20th International Conference on Web Information Systems and Technologies - WEBIST, pages 234–241. INSTICC, SciTePress, 2024, ISBN 978-989-758-718-4. 2, 5, 18, 91, 92, 93, 116

[10] Di Oliveira, Vinícius, Pedro Carvalho Brom, and Li Weigang: *Two-step rag for metadata filtering and statistical llm evaluation: Integrating prompt-aware retrieval and bootstrap mixed models.* IEEE Latin America Transactions, 23(12):1201–1210, 2025. 2, 6, 21, 76, 117

[11] Di Oliveira, Vinícius, Pedro Brom, and Li Weigang: *Immba: Integrated mixed models with bootstrap analysis - a statistical framework for robust llm evaluation.* In *Proceedings of the 21st International Conference on Web Information Systems and Technologies - WEBIST*, pages 92–102. INSTICC, SciTePress, 2025, ISBN 978-989-758-772-6. 2, 6, 116

[12] WCO: *THE HARMONIZED SYSTEM A universal language for international trade.* World Customs Organization, 2018. `https://www.wcoomd.org/-/media/wco/public/global/pdf/topics/nomenclature/activities-and-programmes/30-years-hs/hs-compendium.pdf` (Visited 2024-06-04). 2, 3

[13] MERCOSUR: *MERCOSUR Countries*, 2024. `https://www.mercosur.int/en/about-mercosur/mercosur-countries/`, `https://www.mercosur.int/en/about-mercosur/mercosur-countries/` Accessed on Jun 6th, 2024. 2

[14] CORREIOS: *Correios do Brasil - Tudo sobre CEP.* `https://www.correios.com.br/enviar/precisa-de-ajuda/tudo-sobre-cep`, 2025. Accessed on MAR 4th, 2025. 2, 3

[15] CONFAZ Brazil: *Ajuste SINIEF 17*, 2016. `https://www.confaz.fazenda.gov.br/legislacao/ajustes/2016/AJ_017_16`, `https://www.confaz.fazenda.gov.br/legislacao/ajustes/2016/AJ_017_16` Accessed on Jun 6th, 2024. 4

[16] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin: *Attention is all you need.* Advances in neural information processing systems, 30, 2017. 10, 11, 14

[17] Brown, Tom B, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, *et al.*: *Language models are few-shot learners.* Advances in neural information processing systems, 33:1877–1901, 2020. 10, 11, 15, 16, 30

[18] Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, *et al.*: *Training compute-optimal large language models.* In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030, 2022. 10

[19] Voleti, Rohit, Julie M Liss, and Visar Berisha: *A review of automated speech and language features for assessment of cognitive and thought disorders.* IEEE journal of selected topics in signal processing, 14(2):282–298, 2019. 10

[20] Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei: *Scaling laws for neural language models*. arXiv preprint arXiv:2001.08361, 2020. 10

[21] Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, *et al.*: *Llama: Open and efficient foundation language models*. arXiv preprint arXiv:2302.13971, 2023. 12, 58

[22] OpenAI: *Gpt-4o: Openai's latest flagship model*. `https://openai.com/index/gpt-4o`, 2024. Accessed: 2025-10-17. 12, 14, 17, 30

[23] Jiang, Zheng Xin *et al.*: *Mistral 7b*. Mistral AI, 2024. Accessed: 2025-10-17. 12

[24] Corrêa, Nicholas Kluge, Sophia Falk, Shiza Fatimah, Aniket Sen, and Nythamar De Oliveira: *Teenytinyllama: open-source tiny language models trained in brazilian portuguese*. Machine Learning with Applications. Elsevier, 16:100558, 2024. 13, 29, 57, 59, 72

[25] Cao, Yihan, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip Yu, and Lichao Sun: *A survey of ai-generated content (aigc)*. ACM Computing Surveys. ACM New York, NY, 57(5):1–38, 2025. 13

[26] Rajendran, Rajashree Manjulalayam: *Importance Of Using Generative AI In Education: Dawn of a New Era*. Journal of Science & Technology, 4(6):35–44, 2023. 13

[27] Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, *et al.*: *Language models are few-shot learners*. Advances in neural information processing systems, 33:1877–1901, 2020. 14

[28] Karmaker, Shubhra Kanti and Dongji Feng: *TELeR: A General Taxonomy of LLM Prompts for Benchmarking Complex Tasks*. The 2023 Conference on Empirical Methods in Natural Language Processing, 2023. 14, 15, 16

[29] Yang, Zhe Rui, Jindong Han, Chang Dong Wang, and Hao Liu: *Graphlora: Structure-aware contrastive low-rank adaptation for cross-graph transfer learning*. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1785–1796, 2025. 15

[30] Chen, Banghao, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu: *Unleashing the potential of prompt engineering for large language models*. Patterns. Elsevier, 2025. 15

[31] Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou: *Chain of thought prompting elicits reasoning in large language models*. Advances in Neural Information Processing Systems, 35:24824–24837, 2022. 17, 19, 60

[32] Kojima, Takeshi, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa: *Large language models are zero-shot reasoners.* Advances in neural information processing systems, 35:22199–22213, 2022. 17, 20

[33] Wang, Xuezhi, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou: *Self-consistency improves chain of thought reasoning in language models.* In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. 17, 20, 61

[34] Zhou, Denny, Nathanael Schärli, Yujia Hou, Jason Wei, Xuezhi Wang, Swaroop Mishra, Ed H Chi, Quoc Le, *et al.*: *Least-to-most prompting enables complex reasoning in large language models.* The Eleventh International Conference on Learning Representations, 2022. 17

[35] Zhou, Wangchunshu, Yuchen Eleanor Jiang, Ryan Cotterell, and Mrinmaya Sachan: *Efficient prompting via dynamic in-context learning.* arXiv preprint arXiv:2305.11170, 2023. 18

[36] Greshake, Kai, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz: *Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection.* In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pages 79–90, 2023. 19

[37] Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, *et al.*: *Retrieval-augmented generation for knowledge-intensive nlp tasks.* Advances in Neural Information Processing Systems, 33:9459–9474, 2020. 21, 23, 53, 54, 72

[38] Izacard, Gautier and Edouard Grave: *Leveraging passage retrieval with generative models for open domain question answering.* In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pages 874–880, 2021. 23

[39] Li, Hang, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon: *To interpolate or not to interpolate: Prf, dense and sparse retrievers.* In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 2495–2500, 2022. 24

[40] Karpukhin, Vladimir, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih: *Dense passage retrieval for open-domain question answering.* In *EMNLP (1)*, pages 6769–6781, 2020. 24, 55

[41] Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang: *Retrieval-augmented generation for large language models: A survey.* arXiv preprint arXiv:2312.10997, 2023. 24, 26, 72

[42] Lee, Hyunji, Sohee Yang, Hanseok Oh, and Minjoon Seo: *Generative multi-hop retrieval.* In *2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, 2022. 24, 25

[43] Zhang, Tianjun, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez: *Raft: Adapting language model to domain specific rag*. First Conference on Language Modeling, 2024. 25, 27, 52, 53, 54, 72

[44] Poliakov, Mykhailo and Nadiya Shvai: *Multi-meta-rag: Improving rag for multi-hop queries using database filtering with llm-extracted metadata*. In *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications*, pages 334–342. Springer, 2024. 25, 62, 63, 65, 76, 79, 80, 107

[45] Lin, Xinyu, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat Seng Chua: *Data-efficient Fine-tuning for LLM-based Recommendation*. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 365–374, 2024. 27, 28, 29

[46] VM, Kushala, Harikrishna Warrier, Yogesh Gupta, *et al.*: *Fine Tuning LLM for Enterprise: Practical Guidelines and Recommendations*. arXiv preprint arXiv:2404.10779, 2024. 28

[47] Zhang, Biao, Zhongtao Liu, Colin Cherry, and Orhan Firat: *When scaling meets llm finetuning: The effect of data, model and finetuning method*. The Twelfth International Conference on Learning Representations, ICRL, 2024. 28

[48] Devlin, Jacob, Ming Wei Chang, Kenton Lee, and Kristina Toutanova: *Bert: Pre-training of deep bidirectional transformers for language understanding*. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019. 29

[49] Larcher, Celio, Marcos Piau, Paulo Finardi, Pedro Gengo, Piero Esposito, and Vinicius Caridá: *Cabrita: closing the gap for foreign languages*. arXiv preprint arXiv:2308.11878, 2023. 29, 58

[50] Garcia, Gabriel Lino, Pedro Henrique Paiola, Luis Henrique Morelli, Giovani Candido, Arnaldo Cândido Júnior, Danilo Samuel Jodas, Luis Afonso, Ivan Rizzo Guilherme, Bruno Elias Penteado, and João Paulo Papa: *Introducing Bode: A Fine-Tuned Large Language Model for Portuguese Prompt-Based Task*. CoRR, 2024. 29, 57, 59

[51] Khan, Junaed Younus and Gias Uddin: *Automatic code documentation generation using gpt-3*. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–6, 2022. 30

[52] Hu, Edward J, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen: *Lora: Low-rank adaptation of large language models*. International Conference on Learning Representations, ICLR, 2021. 30, 31, 54, 75

[53] Zeng, Yuchen and Kangwook Lee: *The expressive power of low-rank adaptation*. The Twelfth International Conference on Learning Representations, ICRL, 2023. 31

[54] Xu, Yuhui, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhensu Chen, Xiaopeng Zhang, and Qi Tian: *Qa-lora: Quantization-aware low-rank adaptation of large language models*. The Twelfth International Conference on Learning Representations, ICLR, 2023. 31

[55] Zhang, Longteng, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li: *Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning*. arXiv preprint arXiv:2308.03303, 2023. 31

[56] Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer: *Qlora: Efficient finetuning of quantized llms*. Advances in neural information processing systems, 36:10088–10115, 2023. 31, 54

[57] Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly: *Parameter-efficient transfer learning for nlp*. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 32

[58] Pfeiffer, Jonas, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych: *Adapterhub: A framework for adapting transformers*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, 2020. 32

[59] Zhao, Wayne Xin, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, *et al.*: *A survey of large language models*. arXiv preprint arXiv:2303.18223, 2023. 33

[60] Chang, Yupeng, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, *et al.*: *A survey on evaluation of large language models*. ACM transactions on intelligent systems and technology, 15(3):1–45, 2024. 33, 34

[61] Hu, Taojun and Xiao Hua Zhou: *Unveiling llm evaluation focused on metrics: Challenges and solutions*. arXiv preprint arXiv:2404.09135, 2024. 35

[62] Yang, Z., S. Hao, L. Jiang, Q. Gao, and Y. Ma: *Understanding the Sources of Uncertainty for Large Language and Multimodal Models*. In *International Conference on Learning Representations (ICLR)*, 2024. `https://openreview.net/pdf?id=5By0rus8z7`. 35

[63] Papineni, Kishore, Salim Roukos, Todd Ward, and Wei Jing Zhu: *Bleu: a method for automatic evaluation of machine translation*. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002. 35

[64] Liu, X., D.F. Wong, D. Li, and Z. Wang: *Selectit: Selective Instruction Tuning for LLMs via Uncertainty-Aware Self-Reflection*. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. `https://proceedings.neurips.cc/paper_files/paper/2024/file/b130a5691815f550977e331f8bec08ae-Paper-Conference.pdf`. 36

[65] Kapoor, S., N. Gruver, and M. Roberts: *Large Language Models Must Be Taught to Know What They Don't Know*. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. `https://proceedings.neurips.cc/paper_files/paper/2024/file/9c20f16b05f5e5e70fa07e2a4364b80e-Paper-Conference.pdf`. 36

[66] Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi: *Bertscore: Evaluating text generation with bert*. International Conference on Learning Representations, ICLR, 2019. 37

[67] Sellam, Thibault, Dipanjan Das, and Ankur Parikh: *Bleurt: Learning robust metrics for text generation*. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7881–7892, 2020. 37

[68] Chaudhary, Akhil, Enayat Rajabi, Somayeh Kafaie, and Evangelos Milios: *Fact retrieval from knowledge graphs through semantic and contextual attention*. Expert Systems with Applications, page 127612, 2025. 38

[69] Kalai, Adam Tauman, Ofir Nachum, Santosh S Vempala, and Edwin Zhang: *Why language models hallucinate*. arXiv preprint arXiv:2509.04664, 2025. 38

[70] Fu, Jiayi, Xuandong Zhao, Chengyuan Yao, Heng Wang, Qi Han, and Yanghua Xiao: *Reward shaping to mitigate reward hacking in rlhf*. ICML 2025 Workshop on Reliable and Responsible Foundation Models, 2025. 38

[71] Li, Dawei, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu: *From generation to judgment: Opportunities and challenges of LLM-as-a-judge*. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791. Association for Computational Linguistics, 2025. 39, 66

[72] Boubdir, Meriem, Edward Kim, Beyza Ermis, Sara Hooker, and Marzieh Fadaee: *Elo uncovered: Robustness and best practices in language model evaluation*. Advances in Neural Information Processing Systems, 37:106135–106161, 2024. 39

[73] Liu, Zirui, Jiatong Li, Yan Zhuang, Qi Liu, Shuanghong Shen, Jie Ouyang, Mingyue Cheng, and Shijin Wang: *am-elo: A stable framework for arena-based llm evaluation*. Forty-second International Conference on Machine Learning, 2025. 39

[74] Efron, Bradley and Robert J. Tibshirani: *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993. 40, 66

[75] Laird, Nan M. and James H. Ware: *Random-effects models for longitudinal data*. Biometrics, 38(4):963–974, 1982. 40, 41

[76] Benjamini, Yoav and Yosef Hochberg: *Controlling the false discovery rate: a practical and powerful approach to multiple testing*. Journal of the Royal statistical society: series B (Methodological), 57(1):289–300, 1995. 40, 41, 67, 97

[77] Koehn, Philipp: *Statistical significance tests for machine translation evaluation.* In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395, 2004. 41

[78] Aliero, Abubakar Ahmad, Sulaimon Adebayo Bashir, Hamzat Olanrewaju Aliyu, Amina Gogo Tafida, Umar Kangiwa Bashar, and Muhammad Dankolo Nasiru: *Systematic review on text normalization techniques and its approach to non-standard words.* 2023. 41

[79] Han, Bo and Timothy Baldwin: *Lexical normalization for social media text.* In *ACM Transactions on Intelligent Systems and Technology (TIST)*, volume 4, pages 1–27, 2013. 41, 42, 43, 44

[80] Paetzold, Gustavo H and Lucia Specia: *A survey on lexical simplification.* Journal of Artificial Intelligence Research, 60:549–593, 2017. 44

[81] Cook, Clayton R, Elizabeth Holland, and Tal Slemrod: *Evidence-based reading decoding instruction.* In *Academic Assessment and Intervention*, pages 199–218. Routledge, 2014. 44

[82] Michel, Paul and Graham Neubig: *Mtnt: A testbed for machine translation of noisy text.* Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 543–553, 2018. 44

[83] Gahbiche-Braham, Souhir, Hélène Maynard, and François Yvon: *Two ways to use a noisy parallel news corpus for improving statistical machine translation.* In *Workshop on Building and Using Comparable Corpora*, 2011. 44

[84] Baldwin, Timothy, Marie Catherine De Marneffe, Bo Han, Young Bum Kim, Alan Ritter, and Wei Xu: *Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition.* In *Proceedings of the workshop on noisy user-generated text*, pages 126–135, 2015. 45, 46

[85] Xu, Wei, Alan Ritter, Timothy Baldwin, and Afshin Rahimi: *Proceedings of the seventh workshop on noisy user-generated text (w-nut 2021).* In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, 2021. 45, 46

[86] Du, Shaohua, Zhihao Wu, Huaiyu Wan, and YouFang Lin: *HScodeNet: Combining hierarchical sequential and global spatial information of text for commodity HS code classification.* In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 676–689. Springer, 2021. 45, 56

[87] Cheng, Minhao, Jinfeng Yi, Pin Yu Chen, Huan Zhang, and Cho Jui Hsieh: *Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples.* In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3601–3608, 2020. 46

[88] Ueno, Sei, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara: *Multi-speaker sequence-to-sequence speech synthesis for data augmentation in acoustic-to-word speech recognition.* In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6161–6165. IEEE, 2019. 46

[89] Xue, Linting, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel: *Byt5: Towards a token-free future with pre-trained byte-to-byte models*. Transactions of the Association for Computational Linguistics, 10:291–306, 2022. 46

[90] Van Der Goot, Rob, Alan Ramponi, Arkaitz Zubiaga, Barbara Plank, Benjamin Muller, Iñaki San Vicente Roncal, Nikola Ljubešic, Özlem Çetinoğlu, Rahmad Mahendra, Talha Çolakoglu, *et al.*: *Multilexnorm: A shared task on multilingual lexical normalization*. In *Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 493–509. Association for Computational Linguistics, 2021. 46

[91] Chen, Shijie, Yu Zhang, and Qiang Yang: *Multi-task learning in natural language processing: An overview*. ACM Comput. Surv., 56(12), July 2024, ISSN 0360-0300. `https://doi.org/10.1145/3663363`. 46

[92] Bhattacharjee, Kasturi, Miguel Ballesteros, Rishita Anubhai, Smaranda Muresan, Jie Ma, Faisal Ladhak, and Yaser Al-Onaizan: *To bert or not to bert: Comparing task-specific and task-agnostic semi-supervised approaches for sequence tagging*. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP, pages 7927–7934, 2020. 47

[93] Khattab, Omar and Matei Zaharia: *ColBERT: Efficient and effective passage search via contextualized late interaction over BERT*. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2020. 55

[94] Shi, Weijia, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih: *REPLUG: Retrieval-augmented black-box language models*. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 8371–8384. Association for Computational Linguistics, 2024. 55, 64

[95] Ogaga, Destiny and Abiodun Olalere: *Evaluation and comparison of svm, deep learning, and naïve bayes performances for natural language processing text classification task*. Preprints.org, 2023. 56

[96] Lima, R. R. de, A. M. R. Fernandes, J. R. Bombasar, B. A. da Silva, P. Crocker, and V. R. Q. Leithardt: *An empirical comparison of portuguese and multilingual bert models for auto-classification of ncm codes in international trade*. Big Data and Cognitive Computing, 6(1):8, 2022. `https://doi.org/10.3390/bdcc6010008`. 56, 57

[97] Vertsel, Aliaksei and Maksim Rumiantsau: *Hybrid llm/rule-based approaches to business insights generation from structured data*. arXiv preprint arXiv:2404.15604, 2024. 57

[98] Corradini, Flavio, Moreno Leonesi, and Manuela Piangerelli: *State of the art and future directions of small language models: A systematic review*. Big Data and

Cognitive Computing, 9(7):189, 2025. https://doi.org/10.3390/bdcc9070189. 57

[99] Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, *et al.*: *Llama 2: Open foundation and fine-tuned chat models.* arXiv preprint arXiv:2307.09288, 2023. 58

[100] Pires, Ramon, Hugo Abonizio, Thales Sales Almeida, and Rodrigo Nogueira: *Sabiá: Portuguese large language models.* In *Brazilian Conference on Intelligent Systems*, pages 226–240. Springer, 2023. 59

[101] Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yifeng Li, Xuezhi Wang, Mostafa Dehghani, Sudarshan Brahma, Alex Webson, Sharanay S. Gu, Zhuyun Dai, Mirac Suzgun, Xuan Chen, Aakanksha Chowdhery, Anja Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Shachi Narang, Gaurav Mishra, Allen Yu, Vincent Zhao, Yaochung Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeffrey Dean, Jacob Devlin, Cody Adams, Adam Roberts, Denny Zhou, and Jason Wei: *Scaling instruction-finetuned language models.* Journal of Machine Learning Research, 25:1–53, 2024. http://jmlr.org/papers/v25/23-0870.html. 60

[102] Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou: *Chain-of-thought prompting elicits reasoning in large language models.* In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022. 60

[103] Wang, Yunzhe: *Something-of-thought in llm prompting: An overview of structured llm reasoning.* Blog post on Medium plataform, published in Towards Data Science, sep 2023. medium.com, visited on 2025-10-20, https://medium.com/data-science/something-of-thought-in-llm-prompting-an-overview-of-structured-llm-reasoning-70302752b390. 60, 61

[104] Zhou, Denny, Nathalie Schärli, Le Hou, Jason Wei, Natasa Scales, Xuezhi Wang, Timo Schick, Mingfei Lu, Ilya Sutskever, and Quoc V. Le: *Least-to-most prompting enables complex reasoning in large language models.* The Eleventh International Conference on Learning Representations, 2022. 60

[105] Shin, Jinhan, Chen Tang, Tuhin Mohati, Mohammad Nayebi, Shuai Wang, and Hadi Hemmati: *Prompt engineering or fine tuning: An empirical assessment of large language models in automated software engineering tasks.* arXiv preprint arXiv:2310.10508, 2023. 61

[106] Wang, Haoyu, Tuo Zhao, and Jing Gao: *BlendFilter: Advancing Retrieval-Augmented Large Language Models via Query Generation Blending and Knowledge Filtering.* Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 1009–1025, 2024. 62, 65, 77, 79, 80

[107] Jeong, Soyeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park: *Database-augmented query representation for information retrieval.* In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024. 63, 65, 76, 79, 80

[108] Mombaerts, Laurent, Terry Ding, Adi Banerjee, Florian Felice, Jonathan Taws, and Tarik Borogovac: *Meta Knowledge for Retrieval Augmented Large Language Models.* ArXiv arXiv.2408.09017, 2024. 63, 64, 65, 77, 107

[109] Liu, Yuntao, Dalton Iter, Yichong Xu, Shuai Wang, Ruochen Xu, and Chenguang Zhu: *G-eval: Nlg evaluation using gpt-4 with better human alignment.* Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, ACL, 2023. 66

[110] Li, Haonan, Qingsong Dong, Jiahui Chen, Hongfei Su, Yi Zhou, Qingcai Ai, Guodong Zhang, and Yi Liu: *Llms-as-judges: A comprehensive survey on llm-based evaluation methods.* arXiv preprint arXiv:2412.05579, 2024. 66

[111] Li, Yubin, Yu Fu, Yang Dong, and Cong Liu: *Mace: A hybrid llm serving system with colocated slo-aware continuous retraining alignment.* arXiv preprint arXiv:2510.03283, 2025. 66

[112] Lyu, Xuan, Chang Cao, Maosong Sun, Biao Liang, Ling Yao, and Ru Xu: *Bootstrapping llm-based fact-checking via iterative rationalization finetuning.* In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, April 2025. 67

[113] Borse, Nishant S., R. C. Subramaniam, and Noel S. Rebello: *Investigation of the inter-rater reliability between large language models and human raters in qualitative analysis.* arXiv preprint arXiv:2508.14764, 2025. 67

[114] Pradhan, Apurba, Alex Ortan, Amit Verma, and Murty Seshadri: *LLM-as-a-Judge: Rapid evaluation of legal document recommendation for retrieval-augmented generation.* arXiv preprint arXiv:2509.12382, 2025. 67

[115] Bommasani, Rishi, Percy Liang, and Tony Lee: *Holistic evaluation of language models.* Annals of the New York Academy of Sciences, 1525(1):140–146, 2023. 67, 108

[116] Yu, Fei: *When AIs judge AIs: The rise of Agent-as-a-Judge evaluation for LLMs.* arXiv preprint arXiv:2508.02994, 2025. 67

[117] Shen, Chenhui, Liying Cheng, Xuan Phi Nguyen, Yang You, and Lidong Bing: *Large language models are not yet human-level evaluators for abstractive summarization.* In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. `https://openreview.net/forum?id=ldbYAF0ad0`. 67, 68

[118] Warnakulasuriya, Shakthi and Kaviru Hapuarachchi: *From Knowledge to Action: Leveraging Retrieval Augmented Fine Tuning (RAFT) to Empower Quick and Confident First-Aid Decisions in Emergencies.* Researchgate (Preprint) `http://dx.doi.org/10.13140/RG.2.2.35911.30888`, 2024. 72

[119] Di Oliveira, Vinícius, Li Weigang, and Geraldo Pereira Rocha Filho: *ELEVEN DataSet: A Labeled Set of Descriptions of Goods Captured from Brazilian Electronic Invoices*. In *Proceedings of the 18th International Conference on Web Information Systems and Technologies - WEBIST*, pages 257–264. INSTICC, SciTePress, 2022, ISBN 978-989-758-613-2. 86, 90, 115

[120] Kieckbusch, Diego S, PR Geraldo Filho, Vinícius Di Oliveira, and Li Weigang: *SCAN-NF: A CNN-based System for the Classification of Electronic Invoices through Short-text Product Description.* In *WEBIST*, pages 501–508, 2021. 115

[121] Marinho, Mayara C, Vinicius Di Oliveira, Sérgio APB Neto, Li Weigang, and Vinícius RP Borges: *Visual analysis of electronic invoices to identify suspicious cases of tax frauds*. In *International Conference on Information Technology & Systems*, pages 185–195. Springer, 2022. 115

[122] De Melo, Maısa Kely, Silvia Araújo dos Reis, Vinıcius Di Oliveira, Allan Victor Almeida Faria, Ricardo de Lima, Li Weigang, JF Salm Junior, Joao Gabriel de Moraes Souza, Vérica Freitas, Pedro Carvalho Brom, *et al.*: *Implementing ai for enhanced public services gov. br: A methodology for the brazilian federal government*. In *Proceedings of the 20th International Conference on Web Information Systems and Technologies*, pages 90–101, 2024. 116