Engineering Large Foundational Models for Enterprise Integration

Deliverable D2.2
Initial Release of benchmarking techniques

Project title	Engineering Large Foundational Models for Enterprise Integration
Project acronym	ELFMo
Project number	23004
Work package	WP2
Deliverable	D2.2
Dissemination level	PU (public)
License	CC-BY 4.0
Version	1.0
Date	2025-10-30

Contributors

Editor(s)	Fabio Román/Juan Pablo Gerbi (Dextromedica)
Reviewer(s)	Robin Bornoff (Siemens Industry Software NV)
Contributor(s)	Diogo Martinho (ISEP), Isabel Ribeiro (FTP), André Rodrigues (FTP), Marcos Cobo (CIC), Igor Casado (KONECTA BTO), Rico Pircklen (Siili Solutions Oyj)

Abstract

Deliverable D2.2 presents the *Initial Release of Benchmarking Techniques* developed within Work Package 2 (WP2) of the ELFMo project (*Engineering Large Foundational Models for Enterprise Integration*). It provides the first operational framework for the **selection, efficient training, fine-tuning, and benchmarking** of Large Foundation Models (LFMs) tailored to enterprise contexts. The document defines standardized, reproducible methodologies that combine classical NLP metrics, semantic similarity measures, and novel *LLM-as-a-judge* approaches to enable transparent, evidence-based model evaluation. It also introduces parameter-efficient training strategies (e.g., LoRA, QLoRA) and sustainability-oriented practices to reduce computational and energy costs while maintaining performance and compliance. Together, these contributions lay the foundation for a modular benchmarking toolkit that integrates into MLOps environments, supporting interoperability, traceability, and regulatory alignment. D2.2 thus establishes the methodological baseline upon which future ELFMo deliverables (D2.3, D3.x) will build toward large-scale, domain-specific validation and trustworthy enterprise AI adoption.

Table of contents

1 Introduction

1.1 Context

The ELFMo project (*Engineering Large Foundational Models for Enterprise Integration*) develops a framework for the **reliable integration of Large Foundation Models (LFMs) into business applications**, addressing the high risks, costs, and regulatory challenges associated with Generative AI (GenAI) adoption.

Within this context, **Work Package 2 (WP2): Model training and benchmarking** focuses on developing effective strategies for adapting, training, and evaluating LFMs. Deliverable D2.2, *Initial Release of Benchmarking Techniques*, represents a first step in building systematic evaluation methods that go beyond generic performance indicators and provide **domain-specific benchmarks** aligned with industrial needs.

Benchmarking is a crucial activity because it enables European enterprises to **reduce dependency on dominant model providers** and to **democratize access** to advanced capabilities in *Simulation and Analysis (S&A)*, lowering the complexity of techniques and tools. This ensures that smaller, resource-efficient models can still meet demanding industrial requirements, strengthening competitiveness and sovereignty in AI adoption.

1.2 Objectives

The main objectives of this deliverable are:

1. **To provide an initial set of benchmarking techniques** tailored to enterprise use cases, supporting the fair and reproducible evaluation of adapted LFMs.
2. **To identify and refine critical metrics** beyond generic accuracy, enabling assessment based on industrial KPIs such as efficiency, reliability, and compliance.
3. **To ensure that benchmarking contributes to operational value**, particularly by validating **cost-efficient and low-latency models** suitable for real-world integration.
4. **To contribute to ELFMo's innovation pillars**, particularly:
    - **Innovation 2**: Tools, methods, and infrastructures for trustworthy adaptation and integration of LFMs.
    - **Innovation 3**: Evidence-based procedures for quality and compliance assessment

1.3 Target Audience

This deliverable is published at **PU (public)** level and is therefore intended for a broad audience:

- **Consortium partners**, especially technical partners in WP2, including industrial solution providers, AI service providers, and tool developers.
- **External stakeholders**, including:
  - **Research peers and the scientific community**, interested in advancing benchmarking practices (see Section 3.6 of this deliverable).
  - **Industry and commercial actors**, seeking reliable methods for LFM adaptation and evaluation.
  - **Standardization bodies and policymakers**, interested in technical baselines for compliance with European regulation (EU AI Act, GDPR).

1.4 Related documents

Readers may consult the following documents for complementary context:

- **D2.1 Research baseline for model training and benchmarking** – defines the technical foundations of WP2 and serves as the baseline for this deliverable.
- **D3.1 Research baseline for risk, quality and conformity assessment tools and procedures** – provides additional baselines relevant for WP2 and WP3 activities.
- **FPP (Full Project Proposal)** – details the broader project rationale and innovation objectives

2 Section for technical contributions

2.1 Telemarketing use case

Title	Standardized methodological framework for executing Proof of Concepts (PoC) for LLM solutions in the Contact Center
Description	Through a structured, phased methodology, a strong emphasis on linking with existing business KPIs, and a dedicated chapter to ensure statistical validity, this document seeks to empower the organization to make informed decisions, minimize risks, and maximize the value of investment in LLM technology. This ensures that every step is aligned with the tangible improvement of productivity, quality, and customer experience.
Corresponding contact	Igor Casado
Contributors	Konecta BTO
Life-cycle stage	Development
EFLMo innovations	2 adaptation and integration of LFMs to domain-specific tasks to Contact Center use case
Technological environment	Cloud
Contributions to sustainability? (In relation to UN SDG Goals & Tasks)	☒ Supports responsible data governance and transparent AI use (G16: T.6 & T.10) ☐ Improves energy and resource efficiency of model training and/or inference (G12: T2 & T6) ☐ Enables fair and inclusive access to LFM technologies (G10: T2) ☐ Strengthens safety and robustness of AI systems (G9: T.1 & T.5) ☒ Enhances workforce upskilling and human-AI collaboration (G4: T4 & G8: T.2)
Access	Proprietary

2.1.1 What is it that we have done?

The modern Contact Center has evolved from being a cost center to a strategic value center. In a highly competitive market, every interaction is an opportunity to build customer loyalty or to lose a customer. LLM technology offers the unprecedented ability to analyze and act on these interactions at a scale and depth previously unattainable, enabling a shift from a reactive service to a proactive and even predictive one. Companies that successfully integrate

these tools will gain a significant competitive advantage through greater efficiency, superior personalization, and a deeper understanding of the "voice of the customer". This framework aims to be the roadmap for achieving that successful integration.

1.2. Objectives of the Methodological Framework

- **Standardize:** Provide a unique and repeatable process for evaluating any LLM solution, allowing for objective comparison between different providers or use cases.
- **Focus on Value:** Ensure that each PoC focuses on solving a real and measurable business problem, avoiding "technology testing for the sake of technology".
- **Measure Real Impact:** Define a clear set of quantitative and qualitative metrics to measure the direct impact on Contact Center KPIs (AHT, FCR, CSAT, etc.).
- **Minimize Risks:** Proactively identify and mitigate the risks associated with AI implementation, including data security, privacy, model "biases" and "hallucinations," and resistance to organizational change.
- **Facilitate Decision-Making:** Establish clear success criteria and a final report format that allows leadership to make investment (Go/No-Go) decisions based on empirical evidence.

1.3. Fundamental Principles

Every PoC carried out under this framework will be governed by the following principles:

1. **Focus on Business Value:** Each initiative must begin with a clear hypothesis that links the technology to a desired business outcome (e.g., "Reduce cost per interaction" or "Increase customer retention").
2. **Statistical Rigor:** The experimental design must be robust to ensure that the results are mathematically significant and not a product of chance. This implies the mandatory use of control groups, adequate sample sizes, and sufficient measurement periods.
3. **Agile and Iterative Methodology:** PoCs are not monolithic projects. They should be executed in short cycles that allow for rapid learning, solution adjustment, and adaptation to discoveries made during the process.
4. **Holistic View (People, Processes, Technology):** The evaluation will not be limited to software performance. It will analyze its impact on agents (usability, satisfaction), customers (experience), and existing processes (need for redesign).
5. **Human-in-the-Loop Centrality:** Especially in the initial phases, technology should be seen as a tool to augment human capabilities. Feedback from agents and supervisors is a critical component of the validation process.

2.1.2 Why have we done this?

Generative Artificial Intelligence, and particularly Large Language Models (LLMs), represent a fundamental paradigm shift in how organizations can interact with, understand, and serve their customers. For the

Contact Center, the epicenter of the customer relationship, this technology is not an incremental improvement but a transformative force with the potential to redefine operational efficiency, service quality, and customer experience (CX). However, the path to capturing this value is fraught with technical, operational, and financial complexities. A rushed or poorly planned implementation can result in failed investments, poor customer experiences, and low adoption by agents.

This document presents a comprehensive, rigorous, and standardized methodological framework for executing

Proof of Concepts (PoC) for LLM solutions in the Contact Center. Its purpose is to provide a clear guide for objectively and quantitatively evaluating the feasibility, impact, and return on investment (ROI) of these technologies before a large-scale deployment.

2.1.3 How it works/how it is used?

Each PoC, regardless of the pillar it belongs to, will follow a four-phase structured process. This standardization is key to ensuring the consistency and quality of all evaluations.

Phase 1: Definition and Planning

This is the most critical phase. An error in defining the problem or the measurement plan will invalidate all subsequent efforts.

- **1.1. Use Case Selection and Prioritization:**
  - **Brainstorming:** Identify a list of possible use cases in collaboration with operations, quality, and business leaders.
  - **Selection Criteria:** Evaluate each use case based on:
    - **Volume:** Is it a high-frequency process?
    - **Impact:** Does it have significant potential for improvement in costs, revenue, or CX?
    - **Feasibility:** Do we have the necessary data and technology to address it?
    - **Complexity:** Is it a well-defined and manageable problem for a PoC?
  - **Result:** Selection of a single, specific use case for the PoC (e.g., "Managing order status inquiries in the voice channel").

- **1.2. Hypothesis and Success Criteria Definition:**
  - **Main Hypothesis:** Formulate a clear and falsifiable statement. (e.g., "The implementation of the LLM Copilot will reduce the Average Handling Time (AHT) by at least 15% for agents in the experimental group, while maintaining or improving CSAT").
  - **Primary and Secondary KPIs:** Select the key metrics (see sections for each pillar).
  - **Establish Baselines:** Measure the current performance of the selected KPIs for a representative period (minimum 2 weeks) for the group that will act as a control. This is the benchmark against which improvement will be measured.
  - **Success Criteria (Go/No-Go):** Define clear numerical thresholds. (e.g., "The PoC will be considered a success if it achieves: AHT < 320s, FCR > 85%, and CSAT > 4.5/5. If at least two of these three are not met, the decision will be No-Go") .
- **1.3. Experiment Design:**
  - **Groups:** Define the **Experimental Group (EG)** that will use the new technology and the **Control Group (CG)** that will continue working in the traditional way. Both groups must be equivalent in size, experience, and initial performance.
  - **Sample Size:** Calculate the minimum number of agents and interactions needed to obtain statistically significant results (see section 6).
  - **Duration:** Establish the duration of the execution phase (typically 4-6 weeks) to mitigate novelty effects and capture variability.

Phase 2: Preparation and Configuration

- **2.1. Data Governance and Preparation:**
  - Collect the necessary data assets (recordings, transcripts, knowledge bases, process manuals).
  - Ensure compliance with data protection regulations (GDPR, etc.) through anonymization or pseudo-anonymization processes.
  - Create a **"Golden Dataset"** or a manually labeled reference dataset to validate the model's accuracy (e.g., 1,000 interactions classified by quality experts).
- **2.2. Technical Configuration and Integration:**
  - Deploy the LLM platform in a secure environment (preferably in the cloud).
  - Configure the specific models, prompts, and workflows for the use case.
  - Perform the necessary integrations via API with existing systems (CRM, telephony, ticketing).
- **2.3. Training and Change Management:**

- Clearly communicate the objectives of the PoC to all participants.
- Conduct practical training sessions with the agents of the Experimental Group, ensuring they feel comfortable and competent with the new tool.
- Establish a direct communication channel so that agents can report incidents and provide feedback.

Phase 3: Execution and Measurement

- 3.1. Controlled Launch:
  Activate the LLM solution for the Experimental Group.
- **3.2. Continuous Monitoring:**
  - **Technical:** Supervise the platform's performance (latency, uptime, API error rate).
  - **Operational:** Conduct daily monitoring of key KPIs for both groups, identifying any deviations or anomalies.
- **3.3. Data Collection:**
  - **Quantitative:** Store all data from the defined KPIs in a structured way.
  - **Qualitative:** Conduct weekly satisfaction surveys for agents in the EG. Organize one or two focus group sessions midway and at the end of the trial to obtain detailed feedback on the tool's usability, utility, and frustrations.

Phase 4: Analysis, Conclusions, and Decision

- **4.1. Results Analysis:**
  - Perform a comparative statistical analysis between the EG and the CG (see section 6).
  - Verify if the observed differences are statistically significant.
- **4.2. Impact and ROI Calculation:**
  - Quantify the improvements in operational terms (e.g., "saving X agent hours").
  - Translate these improvements into economic value and project the annualized ROI of the solution.
- **4.3. Results Report and Recommendation:**
  - Prepare a final report that includes: executive summary, methodology, quantitative and qualitative results, ROI analysis, lessons learned, and a clear recommendation (Go/No-Go/Re-test).
- 4.4. Decision Session:
  Present the results to the steering committee or defined stakeholders to make an informed decision about the next steps.

2.2 Selection of LFMs

Title	Selection of Large Foundation Models for Enterprise Integration
Description	Development of a structured methodology for exploring, filtering, and ranking Large Foundation Models (LFMs) suitable for enterprise use cases. This contribution defines criteria (e.g., model size, license, domain, supported languages, resource efficiency) and provides an initial ranking engine supported by benchmarking metrics. The methodology ensures that models selected for adaptation and integration align with business requirements, regulatory constraints, and sustainability goals.
Corresponding contact	Fabio Román (DEXTROMEDICA S.L)
Contributors	DEXTROMEDICA
Life-cycle stage	Model selection
EFLMo innovations	2 (Adaptation and integration of LFMs to domain-specific tasks) 3 (Evidence-based procedures for quality and conformity assessment)
Technological environment	Model selection interfaces connected to HuggingFace Hub, internal repositories, and corporate catalogs; benchmarking engines based on Python and MLOps platforms.
Contributions to sustainability? (In relation to UN SDG Goals & Tasks)	☒ Supports responsible data governance and transparent AI use (G16: T.6 & T.10) ☒ Improves energy and resource efficiency of model training and/or inference (G12: T2 & T6) ☐ Enables fair and inclusive access to LFM technologies (G10: T2) ☐ Strengthens safety and robustness of AI systems (G9: T.1 & T.5) ☒ Enhances workforce upskilling and human-AI collaboration (G4: T4 & G8: T.2)
Access	Proprietary

2.2.1 What is it that we have done?

We have implemented a **systematic selection framework** for LFMs that acts as a strategic entry point for the ELFMo toolkit. It allows exploration of public repositories (e.g., HuggingFace Hub, Model Zoo) and internal corporate catalogs, applying filters based on model characteristics (architecture, license, domain, languages, compute needs) and aligning results with ELFMo's enterprise integration goals.

2.2.2 Why have we done this?

The rapid growth of available LFMs creates a **selection bottleneck** for enterprises. Without standardized methods, organizations risk choosing models that are misaligned with business goals, non-compliant with EU regulations, or inefficient in resource consumption. The selection framework addresses these challenges by:

- Reducing **vendor lock-in** and dependency on proprietary models.
- Enabling **evidence-based comparison** between models using reproducible benchmarks.
- Supporting **compliance** with European AI regulations (GDPR).
- Contributing to **sustainable AI adoption** by prioritizing energy-efficient and cost-effective models.

2.2.3 How it works/how it is used?

The selection methodology consists of three main steps:

1. **Exploration**: Automated scanning of open-source hubs and internal registries to identify candidate models.
2. **Filtering**: Application of criteria including license compatibility, supported modalities, domain relevance, multilingual coverage, and enterprise-grade documentation.
3. **Ranking**: Use of initial benchmarking results (accuracy, inference cost, energy consumption) to prioritize models most suitable for further adaptation.

All results are logged in MLOps platforms (e.g., MLflow, Weights & Biases) to ensure traceability, reproducibility, and transparent decision-making.

2.2.4 Further Reading

- HuggingFace Hub: https://huggingface.co/models
- Model Zoo: https://modelzoo.co/
- ELFMo Deliverable D2.1: *Research baseline for model training and benchmarking*
- ELFMo Deliverable D2.2, Section 5: *Toolkit components for model selection and integration*

2.3 Adaptation and Integration of LFMs

Title	Adaptation and Integration techniques for LFMs
Description	Development of tools and infrastructures that enable the adaptation of Large Foundational Models (LFMs) to enterprise-specific contexts. Contributions include: (i) a Python library for synthetic dataset generation to accelerate model fine-tuning, (ii) orchestration and deployment of inference engines (vLLM, llama.cpp) in local environments, and (iii) routing mechanisms for expert-based load balancing. Together, these elements provide a reproducible and scalable pathway for adapting LFMs and integrating them into production workflows.
Corresponding contact	Marcos Cobo Carrillo (CIC)
Contributors	Marcos Cobo Carrillo (CIC)
Life-cycle stage	Prototype (TRL 4–5)
EFLMo innovations	2, 4
Technological environment	Python library (OSS, GitHub), local GPU servers with vLLM and llama.cpp, Kubernetes/SLURM orchestration, REST routers for inference routing.
Contributions to sustainability? (In relation to UN SDG Goals & Tasks)	☐ Supports responsible data governance and transparent AI use (G16: T.6 & T.10) ☒ Improves energy and resource efficiency of model training and/or inference (G12: T2 & T6) ☒ Enables fair and inclusive access to LFM technologies (G10: T2) ☒ Strengthens safety and robustness of AI systems (G9: T.1 & T.5) ☐ Enhances workforce upskilling and human-AI collaboration (G4: T4 & G8: T.2)
Access	Proprietary

2.3.1 What is it that we have done?

We have implemented an adaptation and integration framework that bridges the gap between research-grade LFMs and enterprise-ready solutions. Our contributions include:

- A Python library (python-data-augmenter) for synthetic dataset generation, enabling domain adaptation even with scarce or sensitive data.
- Infrastructure for orchestrating and deploying multiple inference backends (vLLM and llama.cpp), allowing flexible use of both GPU-intensive and CPU-friendly environments.

- An inference router that balances requests across different models (experts) and backends, enabling ensemble-like responses and resilience against single-model failure.

2.3.2 Why have we done this?

Adapting LFMs to enterprise-specific contexts requires more than fine-tuning pretrained models. Challenges include:

- Data scarcity and sensitivity: many use cases lack sufficient annotated data or cannot expose confidential corpora.
- Heterogeneous compute environments: enterprises need to run LFMs on diverse hardware (local servers, cloud, edge devices).
- Reliability and efficiency: inference needs to be cost-efficient, scalable, and robust, avoiding dependence on a single model/provider.

Our work addresses these challenges by providing data augmentation, flexible inference runtimes, and routing mechanisms that together ensure adaptability, efficiency and trustworthiness in enterprise deployments.

2.3.3 How it works/how it is used?

The adaptation and integration framework is composed of three main building blocks that interact in a unified pipeline:

1. Synthetic Dataset Generation (Python Data Augmenter)
   - Implemented as a Python package, installable via pip.
   - Provides transformation functions such as:
     - Lexical (synonym substitution, lemmatization).
     - Structural (active-passive transformations, sentence reordering).
     - Semantic (paraphrasing using pretrained LLMs, backtranslation).
   - Includes privacy-preserving augmentations (named entity masking, synthetic replacements) to generate datasets without exposing sensitive information.
   - Outputs structured datasets (in JSONL format) and logs for traceability and reproducibility of synthetic data.
2. Orchestration and Deployment of Inference Engines
   - vLLM integration:
     - Optimized for GPU clusters, supports tensor parallelism and paged attention for efficient batched inference.
     - Integrated with Kubernetes operators for elastic scaling (automatic pod replication when load increases).
   - llama.cpp integration:

    - Lightweight C++ backend for inference on CPU-only environments, enabling edge deployments and testing scenarios where GPUs are not available.
    - Bindings exposed through gRPC/REST APIs, containerized for reproducible deployment.
  - Scheduling layer:
    - SLURM or Kubernetes is used to allocate jobs depending on hardware availability (GPU vs CPU).
    - Each deployment logs latency, throughput and resource usage into Prometheus/Grafana dashboards.
3. Inference Router and Expert Balancing
  - Implemented as a gateway service in Python (FastAPI) with pluggable routing strategies:
    - Round-robin: distributes requests equally across models.
    - Performance-aware routing: uses historical latency and quality scores to allocate requests to the most efficient model.
    - Mixture-of-experts (MoE): selects models by domain affinity (e.g., cybersecurity queries routed to a fine-tuned vLLM model; multilingual queries routed to llama.cpp).
  - All inference logs are stored in structured JSON with request ID, model used, latency, and output quality metrics, supporting compliance and auditability.

Usage example:

An enterprise user deploying the framework for a cybersecurity use case can:

1. Use the python-data-augmenter to generate synthetic attack logs and balance benign/malicious classes.
2. Fine-tune a vLLM-based model on augmented data while running a llama.cpp lightweight model for rule-based fallback.
3. Route real-time traffic classification requests through the inference router, which dynamically selects the most reliable expert model depending on load and confidence.

2.3.4 Further Reading

- CIC GitHub repository: https://github.com/CIC-SL/python-data-augmenter
- vLLM documentation: https://github.com/vllm-project/vllm
- llama.cpp project: https://github.com/ggml-org/llama.cpp
- ELFMo Deliverable D2.2, Section 5 (Toolkit components).

2.4 Efficient Training and Fine-tuning Approaches

Title	Efficient Training and Fine-tuning of Large Foundation Models
Description	Development of methodologies and tools for **parameter-efficient training and fine-tuning** of Large Foundation Models (LFMs). The approach prioritizes techniques such as LoRA, QLoRA, adapters, and prompt tuning, combined with synthetic data generation, to adapt models to enterprise-specific contexts while minimizing computational cost, energy consumption, and data requirements.
Corresponding contact	Fabio Román (DEXTROMEDICA)
Contributors	CIC, ISEP, FTP, DEXTROMEDICA
Life-cycle stage	Prototype (TRL 3–4)
EFLMo innovations	2 (Adaptation and integration of LFMs to domain-specific tasks) 4 (Energy-efficient and sustainable AI practices)
Technological environment	Python-based fine-tuning pipelines (PyTorch, HuggingFace Transformers, PEFT libraries), local GPU/TPU clusters, integration with MLOps (MLflow, Kubeflow), synthetic data generation tools.
Contributions to sustainability? (In relation to UN SDG Goals & Tasks)	☒ Supports responsible data governance and transparent AI use (G16: T.6 & T.10) ☒ Improves energy and resource efficiency of model training and/or inference (G12: T2 & T6) ☒ Enables fair and inclusive access to LFM technologies (G10: T2) ☐ Strengthens safety and robustness of AI systems (G9: T.1 & T.5) ☒ Enhances workforce upskilling and human-AI collaboration (G4: T4 & G8: T.2)
Access	Proprietary

2.4.1 What is it that we have done?

We have designed **efficient training and fine-tuning strategies** that allow enterprises to adapt LFMs to domain-specific tasks without the prohibitive costs of full retraining. By combining parameter-efficient techniques (LoRA, QLoRA, adapters) with synthetic dataset generation and transfer learning, we enable organizations to achieve **domain adaptation with lower energy and compute requirements**.

2.4.2 Why have we done this?

Fine-tuning large models poses significant **technical and economic challenges** due to high GPU/TPU costs, energy consumption, and large annotated data needs. Enterprises often lack the resources to retrain full-scale LFMs. This work addresses the following motivations:

- **Lowering the barrier of adoption**: enabling SMEs and regulated industries to customize LFMs efficiently.
- **Sustainability**: reducing energy usage and CO_2 footprint by applying lightweight fine-tuning methods.
- **Data sensitivity**: enabling adaptation through synthetic or privacy-preserving data augmentation, avoiding exposure of sensitive datasets.
- **Rapid deployment**: accelerating the adaptation cycle for enterprise-specific applications.

2.4.3 How it works/how it is used?

The proposed approach integrates into the ELFMo toolkit and follows these steps:

1. **Dataset preparation** – anonymization, augmentation, and generation of synthetic data using LLMs.
2. **Parameter-efficient fine-tuning** – leveraging LoRA/QLoRA, prefix tuning, and adapters to adjust only a fraction of the model's parameters.
3. **Training orchestration** – execution on hybrid infrastructures (local GPU clusters, cloud, or edge) using orchestrators such as Kubernetes or SLURM.
4. **Traceability and governance** – logging all experiments, hyperparameters, and weights into MLOps systems for reproducibility and compliance.
5. **Evaluation and benchmarking** – integrated with D2.2 benchmarking modules (Section 3.5), ensuring adapted models meet performance, efficiency, and compliance thresholds before deployment.

2.4.4 Further Reading

- Hu et al. (2021) *LoRA: Low-Rank Adaptation of Large Language Models*
- Dettmers et al. (2023) *QLoRA: Efficient Finetuning of Quantized LLMs*

- HuggingFace PEFT library: https://huggingface.co/docs/peft
- ELFMo Deliverable D2.1: *Research baseline for model training and benchmarking*
- ELFMo Deliverable D2.2, Section 5: *Toolkit for adaptation and training*

2.5 Benchmarking Techniques

Title	Benchmarking Techniques for Large Foundation Models
Description	Definition and implementation of initial benchmarking methodologies to evaluate Large Foundation Models (LFMs) across industrial use cases. This work establishes a reproducible framework that integrates classical NLP metrics, semantic similarity measures, LLM-as-a-judge approaches, and standardized test suites. The goal is to enable **evidence-based comparison** of models, guiding their selection, adaptation, and deployment in enterprise environments.
Corresponding contact	Fabio Román (DEXTROMEDICA)
Contributors	DEXTROMEDICA, Siili, CIC, FTP, ISEP
Life-cycle stage	Prototype (TRL 3–4)
EFLMo innovations	2 (Adaptation and integration of LFMs to domain-specific tasks) 3 (Evidence-based procedures for quality and conformity assessment)
Technological environment	Python benchmarking libraries (e.g., HuggingFace Evaluate, BERTScore, Sentence-BERT), MLOps integration (MLflow, Weights & Biases), standardized datasets (MMLU, TruthfulQA, HELM).
Contributions to sustainability? (In relation to UN SDG Goals & Tasks)	☒ Supports responsible data governance and transparent AI use (G16: T.6 & T.10) ☒ Improves energy and resource efficiency of model training and/or inference (G12: T2 & T6) ☐ Enables fair and inclusive access to LFM technologies (G10: T2) ☒ Strengthens safety and robustness of AI systems (G9: T.1 & T.5) ☐ Enhances workforce upskilling and human-AI collaboration (G4: T4 & G8: T.2)
Access	Proprietary

2.5.1 What is it that we have done?

We have defined and piloted **benchmarking methodologies** tailored to the evaluation of LFMs in enterprise contexts. These methods go beyond academic metrics by aligning evaluation with business-critical dimensions such as robustness, factual accuracy, efficiency, and compliance.

2.5.2 Why have we done this?

Benchmarking is essential for:

- **Supporting model selection and adaptation**: ensuring enterprises can compare LFMs on a transparent and reproducible basis.
- **Reducing dependency on proprietary providers**: offering European actors independent tools to evaluate open-source and commercial models.
- **Addressing compliance and trustworthiness**: providing evaluation frameworks consistent with EU AI Act requirements.
- **Driving sustainability**: highlighting energy-efficient and cost-effective models suitable for production environments.

2.5.3 How it works/how it is used?

The benchmarking framework combines four complementary approaches:

1. **Traditional NLP metrics** – BLEU, ROUGE, METEOR for text generation quality.
2. **Embedding-based semantic measures** – BERTScore, Sentence-BERT, cosine similarity to capture meaning beyond surface wording.
3. **LLM-as-a-judge** – using advanced LLMs to assess fluency, coherence, and factuality at scale.
4. **Benchmark datasets and test suites** – e.g., MMLU, TruthfulQA, HELM, to provide standardized, multi-dimensional evaluation.

All results are logged in MLOps platforms, generating comparative reports that feed into the ELFMo toolkit (Section 5) for model selection and deployment decisions.

2.5.4 Further Reading

- Hendrycks et al. (2020) *MMLU: Massive Multitask Language Understanding*
- Min et al. (2023) *FActScore: Fine-grained Evaluation of Factual Precision*
- Liang et al. (2022) *HELM: Holistic Evaluation of Language Models*
- ELFMo Deliverable D2.1: *Research baseline for model training and benchmarking*
- ELFMo Deliverable D2.2, Section 4: *Evaluation Frameworks and Comparative Analysis*

2.6 Efficient LLM inference via routing

Title	Efficient LLM inference via routing
Description	Aiming to utilize available OSS language models we are building a custom solution catering to the needs of an e-commerce company for better customer support.
Corresponding contact	Anna Kuokkanen (UH)
Contributors	Anna Kuokkanen (UH), Tomi Sarni (Nosto), Anatoly Soldatov (Nosto).
Life-cycle stage	Processing training data, implementation in progress
EFLMo innovations	2, 4trust
Technological environment	Router module is custom Python implementation; candidate models are open source LLMs from commonly used model families (Llama, Qwen, Mistral)
Contributions to sustainability? (In relation to UN SDG Goals & Tasks)	☐ Supports responsible data governance and transparent AI use (G16: T.6 & T.10) ☒ Improves energy and resource efficiency of model training and/or inference (G12: T2 & T6) ☒ Enables fair and inclusive access to LFM technologies (G10: T2) ☐ Strengthens safety and robustness of AI systems (G9: T.1 & T.5) ☐ Enhances workforce upskilling and human-AI collaboration (G4: T4 & G8: T.2)
Access	MIT

2.6.1 What is it that we have done?

We have gathered anonymous samples of customer prompts given to a chatbot currently in use by Nosto, cleaned them up and manually divided them into three categories: one for text generation without reasoning requirements, second for questions requiring a higher level of reasoning and third for questions requiring coding capabilities. The resulting embeddings will be analyzed to see if the coding category is semantically far enough to use the k-nearest neighbors' algorithm to distinguish it from the other two classes. Then a classifier will be trained to split the first and second categories by complexity, providing a custom mix of these routing strategies. The results will then be compared to using plain kNN, a trained classifier that is distinguished between all three categories and using a single LLM that is more powerful than the LLMs in the ensemble.

The results will be evaluated on domain-specific tasks instead of the usual all-purpose benchmarks, such as MMLU, as commonly used in scientific literature concerning LLM routing.

The reason for this is that the routing solution is tailored to use case in a particular industry setting. Evaluation will target latency, resource consumption and relevance of responses.

2.6.2 Why have we done this?

Routing between LLMs for efficiency (either based on query domain or query complexity) is a well-established system-level improvement to using a general-purpose large LLM. We have identified that even though some queries made to a customer support chatbot do require a more powerful model, most queries are quite simple. A smaller LLM should be able to handle the simple queries if given access to correct information, and possibly some fine-tuning.

By testing ways to implement custom routing solutions and considering their challenges from an industry perspective we hope to provide insights into how relatively simple routing decisions can be utilized to combine smaller models into a more powerful solution. These insights are crucial to reducing wasted computation, leading to lower costs and easier access to LFM technologies.

2.6.3 How it works/how it is used?

The solution will be made available as a Slack extension, similarly to how the customer support bot is currently used. For purposes of conducting experiments, we will also build an application that allows choosing between different alternatives for routing, specifying own sample questions and pretrained routers, prompting the system, viewing the evaluation metrics and results for each prompt and recording them for further analysis and comparison. Code to this application will be publicly available.

3 Evaluation Frameworks and Comparative Analysis

There are multiple ways to approach evaluating generative AI systems. Generative AI models are the starting point when whole generative AI systems are developed. This paper briefly describes four different frameworks for evaluating generative AI models.

3.1 Evaluating Trustworthiness: Factual Accuracy, Bias and such

Is it possible to trust in the correctness of a generative AI model's output to a user's prompt? A slow and costly way to evaluate factual accuracy of a generative AI model is to do it by human annotation, i.e., humans rate the model's answers. Researchers have developed different benchmarks for checking and scoring – partially or completely automatically – how close a model's outputs are to ground truth. Multiple factual accuracy evaluation metrics have been developed. One example is FActScore, which breaks a long-form text generated by a large language model into a series of atomic facts (Min et al., 2023). Another example is TruthfulQA, which is a set of 817 questions that span 38 categories, including health, law, finance and politics. The questions have been designed so that some humans would answer falsely due to a false belief or misconception. To perform well, a generative AI model has to avoid generating false answers learned from imitating human texts. (Lin et al., 2021)

Another approach is to check if an output generated by a large language model is entailed by a reference text. This is called Natural Language Inference (NLI). It is a task and data format, which consists of two input texts and three output classes. The texts are a "context" and a "hypothesis". The task is to determine if the hypothesis is True, False or Neutral given the context (Laurer et al., 2022).

Bias and fairness of a generative AI model can be assessed with benchmarks that test for gender, racial, or cultural bias in responses. Winogender is one example of this technique, which can be used to evaluate and confirm systematic gender bias in coreference resolution systems (Rudinger et al., 2018). Other possible methods to assess bias and fairness in a generative AI model are stereotype content tests (Fraser et al., 2021) and quantitative disparity metrics (Clavell et al., 2025).

Other trust aspects of a generative AI model are consistency and robustness. Model's consistency is checked by asking the same question multiple times and measuring agreement. Model's robustness can be tested by introducing slight input variations (typos, rewordings) and checking stability of outputs.

3.2 Embedding-based and semantic similarity measures

Generative AI models encode data into high-dimensional vectors called embeddings. Embeddings capture the semantic meaning of for example texts, images or audio.

Embeddings enable comparison of meaning of data instead of exact data. Embedding-based and semantic similarity measures are used to judge how well a generated answer matches a reference answer (or retrieved source) by comparing their embeddings. They go beyond surface-level matching – like bilingual language understudy (BLEU) – capturing paraphrases and semantic overlap.

Mathematical analysis can be used to check how close two embeddings are i.e., their semantic similarity. Cosine similarity is one such evaluation method, though recent findings suggest it has its limitations (Steck et al., 2024). Other methods are for example BERTScore (Zhang et al., 2019) and Sentence-BERT (Reimers & Gurevych, 2019), both based on bidirectional encoder representations from transformers (BERT).

3.3 LLM-as-a-judge approaches

LLM-as-a-judge has emerged as a transformative evaluation approach for generative AI systems, utilizing the assessment capabilities of large language models to judge the quality and performance of AI-generated outputs. This methodology marks a fundamental shift from conventional human-annotation evaluation practices, delivering scalability and reliability that surpass manual assessment limitations. The approach centers on employing advanced language models (such as GPT-4, Claude, or comparable high-performance models) as evaluators to assess AI system outputs across various criteria. These judge models receive structured prompts containing evaluation guidelines, assessment frameworks, and examples of quality responses which guide them in making systematic and objective assessments of the generated content. (Guo, 2025)

The methodology demonstrates effectiveness in assessing conversational AI, creative writing tasks, and complex reasoning challenges where conventional metrics like BLEU or ROUGE prove inadequate (Zheng et al., 2023). However, the LLM-as-a-judge approach is not without limitations. Position bias, where the judge model favours responses presented first or last, remains a significant concern (Shi et al., 2024). Additionally, the approach can perpetuate biases present in the judge model itself in addition to there being an inherent circularity in using AI to evaluate AI (Clarke & Dietz, 2025). Moreover, different judge models can produce varying assessments of the same outputs, highlighting the importance of choosing appropriate judge models and validation methods (Zheng et al., 2023). Therefore, the practical implementation of LLM-as-a-judge requires careful prompt engineering, consideration of evaluation dimensions, and often involves multiple judge models for cross-validation.

3.4 Benchmark datasets and test suites

Standardized benchmarking datasets and test suites serve as the foundation for systematic GenAI evaluation, delivering consistent, measurable assessments of model performance across varied tasks and domains. These frameworks allow researchers and practitioners to objectively compare models, monitor advancement over time, and pinpoint capabilities and limitations within AI systems.

Modern benchmarks have shifted from simple correctness measures to nuanced assessments that evaluate how models think, reason, and navigate complex real-world situations rather than just whether they produce the right answer. These modern evaluations span diverse dimensions, from traditional NLP tasks to advanced reasoning challenges and safety assessments. For example, Massive Multitask Language Understanding (MMLU) doesn't just test whether models can answer factual questions correctly but evaluates their ability to apply knowledge across 57 diverse academic subjects, thereby measuring genuine understanding and knowledge transfer rather than memorized responses (Hendrycks et al., 2020).

However, benchmark limitations have become increasingly apparent as models achieve near-perfect scores on established datasets. Benchmarks can suffer from data contamination issues, where models may have encountered test examples during training, inflating performance scores but also static benchmarks may not capture the dynamic nature of real-world AI applications (Reuel et al., 2024). In addition, there's growing recognition of the need for culturally diverse benchmarks that assess model performance across different languages, cultural contexts, and demographic groups (Franklin et al., 2024).

The field is moving toward more holistic evaluation frameworks that combine multiple benchmarks, incorporate human judgment, and assess real-world performance. Initiatives like the Holistic Evaluation of Language Models (HELM) project aim to provide comprehensive, multi-dimensional assessment of language models that goes beyond traditional accuracy metrics to include fairness, robustness, bias detection, and safety considerations across diverse real-world scenarios (Liang et al., 2022).

3.5 Comparative assessment of frameworks

These four evaluation frameworks each bring distinct advantages and limitations. Trustworthiness evaluations directly address factual accuracy, bias, and robustness, but they are often costly and resource intensive. Embedding-based and semantic similarity measures offer efficient, automated ways to capture meaning beyond surface wording, though they may conflate similarity with truth. The LLM-as-a-judge approach provides scalable, nuanced assessment across complex tasks, yet it risks bias and circularity by relying on AI to evaluate

AI. Benchmarking datasets ensure standardization and comparability, but they can become saturated, face data contamination issues, and struggle to reflect real-world diversity.

Each framework contributes unique value to evaluating generative AI models, but none is sufficient alone. The most robust path forward is a hybrid evaluation strategy: combining the objectivity of benchmarks, the semantic depth of embedding-based measures, the scalability of LLM-as-a-judge, and the reliability of trustworthiness checks. Such integrated frameworks would provide a multi-dimensional, balanced picture of generative AI performance, ensuring both technical rigor and real-world applicability.

4 Initial Toolkit for Selection, Adaptation, Training and Benchmarking

4.1 Software components and architecture

After describing in the previous section the benchmarking techniques applied to models, it is essential to take a further step and define architecture that enables working with LFMs in a structured way. The objective is not limited to evaluating models in isolation: we need a toolkit that facilitates their selection, adaptation, training, and deployment within a controlled and reproducible environment.

This toolkit is conceived as the bridge between the research domain and operational environments. It must ensure experiment traceability, efficient resource usage, and explicit alignment with ELFMO's strategic use cases. Only under this vision will it be possible to establish an end-to-end flow that transforms academic results into robust solutions for customers and end users.

4.1.1 Design principles

Before detailing the technical modules, it is appropriate to set the principles that guide the proposed architecture. The goal is to have a stable system, capable of evolving at the pace of the rapid progress of foundation models.

- Modularity. Each stage of the lifecycle (selection, fine-tuning, benchmarking, deployment, and monitoring) should be implemented as an independent module. This allows components to be replaced, optimized, or reused without affecting the rest of the system.
- Scalability. The toolkit must operate both in lightweight development environments (on a laptop) and in distributed clusters under Kubernetes, SLURM, or public clouds.
- Interoperability. Interaction between modules is standardized through REST/gRPC APIs, ensuring seamless integration into existing corporate pipelines.
- Reproducibility and governance. All metrics, weights, and training configurations must be recorded in MLOps platforms, enabling audits and version control.
- Security and compliance. Access to data and models is governed by strict RBAC policies and log traceability, ensuring conformity with regulatory frameworks such as GDPR or ISO 27001.

4.1.2 Core toolkit components

The toolkit is organized into six functional blocks, integrated within an orchestrated pipeline. Each block serves a specific purpose and, together, they constitute the ecosystem required to manage foundation models.

The following diagram illustrates an example end-to-end workflow covering from model selection to its deployment and monitoring in production:

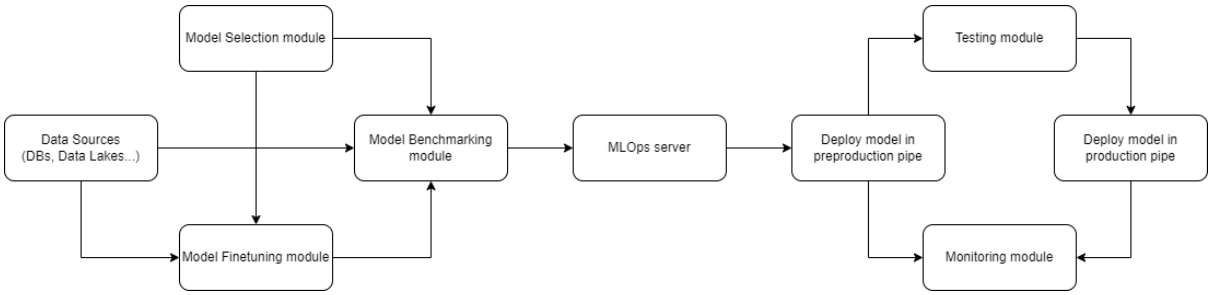

Figure 1. Example end-to-end workflow diagram

- **Model selection module**

Acts as a strategic entry point. It enables exploration of public repositories (HuggingFace Hub, Model Zoo) and internal corporate catalogs. Filters are based on criteria such as model size, license, domain, or supported languages. It also includes an initial ranking engine supported by benchmarking metrics (accuracy, inference cost, energy consumption). Its output directly supports decision-making, avoiding allocation of resources to low-potential models.

- **Fine-tuning module**

When required, it enables adapting a model to specific domains. It includes dataset preparation (cleaning, anonymization, synthetic generation with LLMs), the use of efficient adaptation techniques (LoRA, adapters, prefix-tuning), and detailed logging of weights, hyperparameters, and logs in MLOps systems. In this way, a general-purpose model can be tailored to concrete contexts such as cybersecurity (malicious traffic detection) or business intelligence (multilingual queries in LUCA BDS).

- **Benchmarking module**

Key to rigorous and reproducible evaluation. It runs tests with classical metrics (BLEU, ROUGE, METEOR) and more advanced semantic metrics (BERTScore, cosine similarity, CLIPScore). It also incorporates qualitative assessment via LLM-as-a-judge to evaluate fluency, coherence, and factuality. The results are comparative reports that determine whether a model is ready to advance to deployment.

- **MLOps server**

Operates as the central repository of models, experiments, and artifacts. Solutions such as MLflow, Kubeflow, or Weights & Biases allow model versioning and CI/CD pipeline automation, integrating validations and records. This guarantees traceability and a single source of truth throughout the lifecycle.

- **Testing environment**

Before releasing a model to production, the full pipeline is validated. It includes a Testing module (functional robustness, load, security) and a Monitoring module for real-time observability (latency, energy consumption, data drift, concept drift). This phase acts as a critical barrier against risks in sensitive environments.

- **Production environment**

The validated model is exposed through secure APIs. Deployment incorporates continuous monitoring and rollback capability, ensuring resilience against performance degradation.

4.1.3 Data flow and orchestration

The defined pipeline follows a logical order that ensures consistency and control at each stage:

1. Model selection from public and corporate catalogs.
2. (Optional) Fine-tuning with internal or synthetic data.
3. Initial benchmarking, comparing base and adapted models.
4. Registration in MLOps of weights, hyperparameters, and metrics.
5. Deployment to pre-production, with exhaustive validation via Testing and Monitoring.
6. Promotion to production, where the model is available via a secure API, with active monitoring and recovery capability.

This flow guarantees that each transition is documented and that no model reaches production without passing all validation phases.

4.1.4 Supporting infrastructure

The correct functioning of the toolkit relies on a solid infrastructure, organized into three main layers:

- Data layer. Provides unified access to structured sources (databases, data lakes) and unstructured sources (documents, logs).

- Security layer. Implements authentication, authorization, and encryption both in transit and at rest.
- Monitoring layer. Delivers key metrics (performance, inference latency, energy consumption, and drift) that allow anticipating incidents and triggering early responses.

Taken together, these layers reinforce system reliability and ensure that LFM integration within ELFMO is carried out under enterprise-grade technical and security standards.

4.2 APIs and integration with adaptation pipelines

With the core components of the toolkit defined, the next step is to specify how they integrate operationally within real corporate environments. The backbone of this integration lies in standardized, resilient APIs capable of linking internal modules while seamlessly connecting to adaptation pipelines already deployed across organizations.

APIs should not be understood solely as software interfaces, but as interoperability contracts that ensure compatibility, reproducibility, and scalability. Under this premise, the toolkit incorporates an abstraction layer that unifies interaction with models, datasets, and artifacts. This layer is implemented following open standards (OpenAPI, Protobuf) and integrates natively with orchestrators such as Kubeflow Pipelines, Apache Airflow, or Jenkins. In the ELFMO context, this approach enables reuse of existing infrastructure (data lakes on S3/GCS, MLOps systems like MLflow or Weights & Biases), avoiding redundancy and accelerating LFM adoption in strategic processes.

4.2.1 API design principles

API design must guarantee reliability and flexibility in production. To this end, four foundational principles are established:

- Standardization. All operations are exposed via REST/gRPC interfaces, described in OpenAPI or Protobuf, enabling automatic parameter validation and simplifying integration into heterogeneous pipelines.
- Versioning. Each API implements explicit version control (semantic versioning) and backward compatibility, ensuring continuity of existing integrations as the platform evolves.
- Security. Access is controlled through OAuth2/JWT, RBAC policies, TLS encryption, and centralized access auditing (ELK Stack, OpenTelemetry).
- Observability. Each invocation emits traces and metrics exposed through Prometheus and visualized in Grafana, enabling usage monitoring, anomaly detection, and early identification of bottlenecks.

4.2.2 Integration with adaptation pipelines

The second design layer focuses on orchestration within adaptation pipelines. The objective is for any training, validation, or deployment process to be managed uniformly, irrespective of the execution environment.

- Training and fine-tuning. APIs allow pipelines to load datasets from corporate data lakes, configure hyperparameters (batch size, learning rate, number of epochs), and launch processes on Kubernetes, SLURM, or public clouds with GPU/TPU. The entire cycle is documented in an MLOps server, guaranteeing traceability and reproducibility.
- Evaluation and validation. APIs enable automated benchmarks (BLEU, ROUGE, BERTScore, CLIPScore, LLM-as-a-judge), consolidating results into formats consumable by dashboards (Grafana, Tableau, PowerBI, LUCA-BDS). Objective and subjective metrics are thus integrated into reporting systems, supporting promotion decisions.
- Hybrid pipelines. The architecture allows critical preprocessing phases to run on local infrastructure, while intensive processes (hyperparameter search, distributed training) are delegated to the cloud. This flexibility balances security, cost, and scalability.

4.2.3 Integration benefits

Adopting APIs designed under these principles provides clear benefits:

- Reduced integration and deployment times thanks to standardized interfaces.
- Efficient reuse of existing corporate data and MLOps infrastructure, avoiding artifact duplication.
- Enhanced governance through automatic auditing of interactions with models and datasets.
- Native scalability via hybrid pipelines and distributed orchestration on Kubernetes or public clouds.

In summary, APIs become the toolkit's core interoperability mechanism, transforming it into an extensible platform, integrated with the corporate ecosystem and capable of evolving in parallel with foundation model advances.

4.3 Example workflows and use cases

The practical value of the architecture is manifested in specific workflows that combine the toolkit's modules with the integration of APIs. These flows not only demonstrate the system's technical capability, but also its applicability across domains, from academic research to enterprise production deployments.

The following cases illustrate how organizations can incorporate Large Foundation Models into hybrid environments in a controlled, secure, and efficient manner.

4.3.1 Workflow 1: Adoption of an expert model for cybersecurity log analysis

Example pipeline aimed at detecting anomalies in large volumes of network and system logs:

1. Selection of a base model trained on technical data and registration in MLflow for initial traceability.
2. Preparation of logs extracted from a data lake with anonymization and cleaning using Spark.
3. Efficient fine-tuning with LoRA on Kubernetes with GPUs, logging hyperparameters and weights in Weights & Biases.
4. Model validation with anomaly-detection metrics and semantic evaluation via LLM-as-a-judge.
5. Deployment of the model as a secure API in pre-production, monitored with Prometheus and promoted to production with rollback enabled.

Expected outcome: a system that strengthens SOCs, detecting malicious traffic in real time with full traceability and regulatory compliance.

4.3.2 Workflow 2: Adaptation of a conversational model to answer user questions

Example pipeline focused on creating a multilingual assistant for support in digital platforms:

1. Selection of an instruction-tuned question-answering model and registration in the corporate Model Registry.
2. Preparation of datasets from FAQs, support tickets, and internal documentation, with anonymization and enrichment via synthetic examples.
3. Adaptation with prefix-tuning in scalable cloud environments, orchestrated with Kubeflow Pipelines.
4. Validation using text-generation metrics (BERTScore) and human review to assess factuality.
5. Deployment as an API with OAuth2 authentication, integrated into the platform and monitored with dashboards for A/B testing and user satisfaction metrics.

Expected outcome: a corporate conversational assistant that is accurate, multilingual, and aligned with the organization's communication style.

5 Technical Analysis Benchmark – LFM for ERP

5.1 Research Baseline

The Portuguese use case (as described in the wp1 use case document) addresses the modernization of Enterprise Resource Planning (ERP) platforms, traditionally implemented as monolithic systems with limited flexibility and scalability. The objective is to transition ERP architecture toward modular monoliths and ultimately microservices-based ecosystems, enabling integration of Large Foundation Models (LFMs) for intelligent automation, predictive analytics, and natural language interaction. This transformation is particularly relevant for industrial and e-commerce domains, where interoperability, compliance with European regulations, and advanced data-driven decision-making models are key business enablers.

5.1.1 Objectives

- Automate ERP business processes through AI and LFMs.
- Improve forecasting, planning, and decision support using predictive analytics.
- Enable intuitive, natural language interaction with ERP modules.
- Transition ERP solutions from monolithic to microservices architectures.
- Ensure compliance with GDPR, the AI Act, and information security standards.
- Provide modular, scalable APIs for secure integration with legacy and third-party systems.

5.1.2 Key Activities and Components

- **Architecture Migration Module**: Tools and methods to evolve legacy ERP systems into modular and microservices-based architecture while maintaining compatibility with existing workflows.
- **Conversational AI Interface**: Integration of LFMs to provide natural language interfaces for finance, HR, logistics, and supply chain management.
- **Predictive Analytics Engine**: AI-driven forecasting tools to support financial planning, HR optimization, and supply chain resilience.
- **API Gateway & Data Pipelines**: Secure, scalable APIs and data integration pipelines enabling interoperability across enterprise applications and external platforms.
- **Compliance and Transparency Layer**: Mechanisms for GDPR/AI Act adherence, bias mitigation, and explainability in AI-driven decisions.
- **Monitoring & Validation Framework**: Continuous monitoring of model performance, bias detection, and anomaly tracking aligned with LFMOps practices.

5.1.3 Technological Stack

Some examples of possible technologies to be considered for the Portuguese Use Case development:

- **Modeling & AI**: Hugging Face Transformers, PyTorch, PEFT (LoRA/QLoRA), Retrieval-Augmented Generation (RAG).
- **Architecture & Integration**: Kubernetes, Docker, REST/GraphQL APIs, Event-driven microservices.
- **Data Layer**: PostgreSQL, MongoDB, data lakes with ETL/ELT pipelines, vector databases (e.g., ChromaDB).
- **Monitoring & Security**: Grafana, Prometheus, MLflow, OWASP-aligned validation protocols.
- **Compliance & Fairness**: AI Fairness 360, Fairlearn, audit logging frameworks.

5.1.4 Strategic Contribution to WP2

This use case contributes to WP2 by providing a **scalable, modular solution** for ERP modernization using LFMs. It shows how AI-enhanced ERP systems can ensure secure, explainable, and regulation-compliant automation, supporting enterprise-wide adoption of trustworthy LFMs. The Portuguese ERP use case will provide advanced tools and methods for domain-specific adaptation and integration, while also ensuring validation and dissemination of reproducible frameworks.

5.1.5 Expected Impact

- Improved ERP flexibility and reduced vendor lock-in through modular, microservices-based design.
- Enhanced enterprise decision-making with AI-driven forecasting and analytics.
- Increased user satisfaction via natural language interfaces and intelligent automation.
- Strengthened compliance with GDPR, AI Act, and any existing relevant standards.
- Lower operational and integration costs through scalable APIs and parameter-efficient fine-tuning methods.
- Contribution to European digital sovereignty in enterprise software by promoting secure, transparent, and LFM-enhanced ERP solutions.

5.2 Software Components and Architecture Design

5.2.1 Backend and Frontend Frameworks

The web platform is structured into frontend and backend components, ensuring modularity, scalability, and secure communication with ERP and AI systems. The backend is implemented using the Laravel framework in PHP, which follows the Model-View-Controller paradigm and

provides mechanisms for API development, authentication, and business logic management. A dedicated relational database, (MySQL) is used to store platform-specific data such as user accounts, subscription information, and system logs. The backend layer also exposes RESTful APIs for communication with external clients and manages the interaction with ERP and AI services.

The frontend is developed using React.js, which provides reusable components, responsive design, and state management suitable for enterprise-grade applications. It delivers dashboards, navigation elements, and visualization tools that allow users to interact with ERP records and AI-driven analytics. Communication between the frontend and backend is established through RESTful APIs queries, ensuring a clean separation of concerns. This layered architecture ensures that the frontend remains a lightweight presentation layer, while the backend manages workflows, data integrity, and service integration.

5.2.2 Enterprise Resource Planning (ERP) Syncronization

To support client operations over ERP functionalities, the system enables secure integration with existing ERP software. ERP systems such as Cegid or PHC rely on Microsoft SQL Server databases, which serve as the primary source of enterprise data, including finance, human resources, logistics, and supply chain information. Access to these ERP systems is achieved through dedicated webservice APIs, which may follow REST or SOAP standards depending on the vendor. These webservices enable structured request and response exchanges, either in JSON or XML format, and support both synchronous, real-time communication as well as asynchronous synchronization for batch data transfers.

Security is ensured through the use of authorization tokens embedded in API requests. The management of these tokens, including their issuance and lifecycle, is performed by a separate subscription management platform. This guarantees that multi-tenant access is controlled and compliant with enterprise security requirements. The synchronization process allows clients not only to consult ERP data for reporting or monitoring but also to perform write operations such as updating financial records, processing logistics orders, or managing HR information. This approach maintains interoperability between legacy ERP systems and modern web-based platforms without compromising data integrity or compliance requirements.

5.2.3 LLM Integration

The integration of large language models (LLMs) is designed to enhance ERP functionalities through natural language interaction and AI-driven insights. The LLM services are deployed

as microservices behind an API gateway, which ensures modularity, secure access, rate limiting, and detailed logging of all transactions. User queries submitted through the frontend interface are transmitted in JSON format to the backend, which forwards them to the LLM service. The backend is also responsible for processing responses and adapting them for frontend presentation.

To support context-aware interaction, LLM requests may be augmented with data from enterprise datasets stored in vector databases such as ChromaDB or Pinecone. This enables retrieval-augmented generation (RAG), which enriches responses with domain-specific knowledge derived from ERP records or other business datasets. The LLM integration supports a wide range of use cases, including conversational access to ERP data, predictive analytics for demand forecasting or HR planning, and automatic report generation.

All interactions with LLMs are subject to validation and compliance checks to ensure adherence to GDPR, the AI Act, and information security standards. Requests and responses are logged for auditing purposes, and bias mitigation as well as explainability frameworks can be applied to guarantee transparency in AI-driven decision support. This integration extends the ERP platform from traditional form-driven operations to more intuitive and intelligent interactions, allowing enterprises to leverage advanced predictive and prescriptive capabilities.

5.2.4 System Workflow

The following diagram illustrates the general workflow to support the envisioned system regarding the Portuguese Use Case:

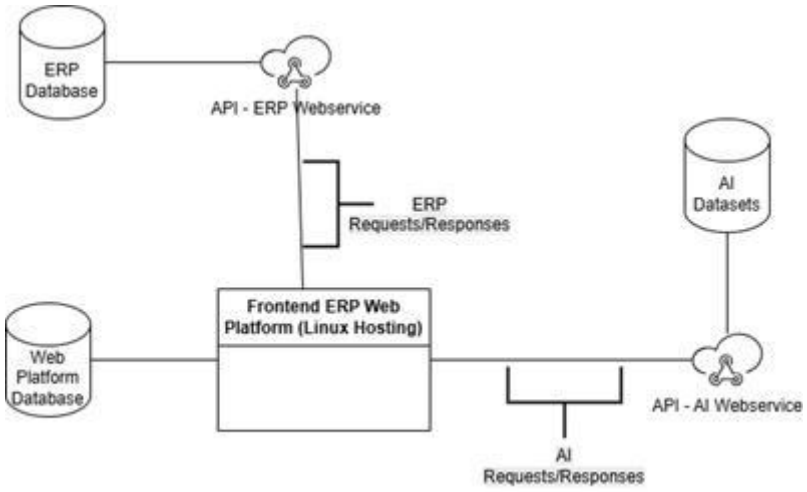

Figure 2. Diagram Portuguese Use Case workflow

The system architecture supports the integration of existing ERP solutions with AI-based services through a modular web platform. The diagram illustrates the main components and data flows.

Frontend ERP Web Platform

- **Role:** Central application layer for user access and system interaction.
- **Implementation:** Developed using React (frontend) and Laravel/PHP (backend), hosted on Linux infrastructure.
- **Database:** Dedicated web platform database (MySQL/MariaDB) to manage application data, subscriptions, and logs.
- **Functionality:** Provides RESTful API endpoints for client applications, manages business logic, and coordinates communication with ERP and AI services.

ERP System Integration

- **ERP Database:** Legacy ERP solutions (e.g., Cegid, PHC) typically based on Microsoft SQL Server.
- **Integration Method:** Connection through an ERP Webservice API, supporting structured requests and responses (JSON or SOAP).
- **Authentication:** Access secured with authorization tokens, managed via a client subscription management tool.
- **Purpose:** Enables secure read and write operations in ERP systems, covering domains such as finance, human resources, logistics, and supply chain management.

AI/LLM Integration

- **AI Webservice API:** Provides access to large language models and related AI modules for conversational interaction, predictive analytics, and automation tasks.
- **AI Datasets:** Storage layer for training data, embeddings, and retrieval-augmented generation pipelines.
- **Data Exchange:** Requests are transmitted in JSON format; responses are processed through the backend before presentation in the frontend interface.
- **Scalability:** Exposed as microservices via an API gateway, allowing modular deployment and future expansion.

General Methods of Application

- **API-Based Interoperability:** REST APIs provides a standardized mechanism for connecting ERP data with AI services and third-party systems.

- **Progressive Migration:** Legacy ERP modules are gradually encapsulated into services, enabling transition from monolithic to microservices-based architectures.
- **AI-Supported Processes:** Predictive models and natural language interfaces enhance planning, decision support, and user interaction across business domains.
- **Security and Compliance:** Authentication, access control, and compliance measures ensure alignment with GDPR, the AI Act, and information security standards.

This architecture enables ERP modernization while ensuring secure, scalable, and regulation-compliant integration of AI technologies.

6 Preliminary Results

At this stage, the focus of WP2 has been on designing and validating **general methods** for the selection, adaptation, training, and benchmarking of Large Foundation Models (LFMs). The preliminary results are not final performance outcomes, but rather **demonstrations of applicability** of the methodological frameworks developed so far.

The methods validated include:

1. **Model Selection Frameworks**
    - Demonstrated that systematic exploration and filtering of LFMs from public hubs (e.g., HuggingFace) and corporate catalogs can be automated.
    - Initial rankings of candidate models based on domain relevance, license constraints, and energy efficiency show that this process is directly applicable across domains such as finance, healthcare, and e-commerce.
2. **Efficient Training and Fine-tuning**
    - Early experiments confirm that parameter-efficient techniques (LoRA, QLoRA, adapters) significantly reduce compute costs while enabling effective domain adaptation.
    - Synthetic data generation has been successfully applied to augment limited datasets, showing promise for regulated domains with scarce or sensitive data.
3. **Benchmarking Techniques**
    - The integration of traditional NLP metrics (BLEU, ROUGE), semantic similarity measures (BERTScore, Sentence-BERT), and LLM-as-a-judge approaches has been validated in initial testbeds.
    - These techniques allow flexible evaluation beyond generic accuracy, capturing dimensions of factuality, efficiency, and compliance.
4. **Toolkit Integration**
    - Preliminary tests of the ELFMo toolkit architecture confirm that the modules for selection, fine-tuning, benchmarking, and deployment can be orchestrated within MLOps environments.
    - The methods are already applicable in hybrid infrastructures (local clusters, cloud, edge), showing scalability and interoperability with existing enterprise pipelines.

5. General Applicability

These results demonstrate that the proposed methods are:

- **Domain-agnostic**: applicable across multiple industrial sectors (telecommunications, ERP systems, cybersecurity, healthcare).
- **Resource-efficient**: adaptable to both high-performance clusters and lightweight environments.
- **Regulation-ready**: designed to align with European requirements on transparency, traceability, and responsible AI.

The next phase of work will involve applying these methods to **specific use cases** (see Section 5) and refining them through empirical benchmarking, ensuring that the approaches move from general applicability to measurable industrial impact.

7 Conclusions

This deliverable, **D2.2 Initial Release of Benchmarking Techniques**, provides the first operational outcomes of WP2, establishing a methodological foundation for the **selection, training, fine-tuning, and benchmarking** of Large Foundation Models (LFMs) in enterprise contexts.

-Executive summary of contributions

- **Model Selection**: A structured framework for systematically exploring and ranking LFMs according to domain relevance, license, efficiency, and compliance.
- **Efficient Training and Fine-tuning**: Parameter-efficient approaches (LoRA, QLoRA, adapters) validated as cost-effective and sustainable methods for adapting models to industrial needs.
- **Benchmarking Techniques**: Initial integration of classical NLP metrics, semantic similarity measures, LLM-as-a-judge approaches, and standardized datasets, enabling transparent and reproducible evaluation.
- **Toolkit Integration**: Early results confirm the feasibility of orchestrating these modules into a unified pipeline with MLOps support.

-Interrelation of techniques

The techniques developed are **complementary and interoperable**:

- Model selection narrows down candidate models.
- Efficient training/fine-tuning enables targeted adaptation.
- Benchmarking provides evidence-based validation.
- Toolkit integration ensures smooth transition from research-grade experimentation to enterprise deployment.

-Status and maturity

The current maturity is at **prototype level (TRL 3–4)**. The methods have been validated conceptually and in controlled environments, demonstrating domain-agnostic applicability, but large-scale industrial validation is ongoing.

-Future work and directions

- Extend benchmarking frameworks with domain-specific testbeds and multilingual evaluation.

- Apply fine-tuning and benchmarking pipelines in **pilot use cases** (ERP modernization, Contact Center, e-commerce).
- Advance toolkit maturity to **TRL 5–6**, focusing on scalability, security, and compliance with the EU AI Act.
- Strengthen sustainability impact by further optimizing energy efficiency and enabling broader access for SMEs.

In summary, **D2.2 delivers the initial building blocks of the ELFMo framework**, showing that systematic methods for model selection, adaptation, and benchmarking are feasible and already applicable across multiple industrial contexts. The next deliverables (D2.3 and beyond) will consolidate these approaches into refined, domain-tested solutions, bridging the gap between research and enterprise integration.

8 References

Clarke, Charles & Dietz, Laura. (2025). LLM-based relevance assessment still can't replace human relevance assessment. arXiv.

Clavell, Gemma G. et al. (2025). Demographic Benchmarking: Bridging Socio-Technical Gaps in Bias Detection. arXiv.

Franklin, Gillian et al. (2024). The Sociodemographic Biases in Machine Learning Algorithms: A Biomedical Informatics Perspective. Life (Basel)

Fraser, Kathleen C. et al. (2021). Understanding and Countering Stereotypes: A Computational Approach to the Stereotype Content Model. arXiv.

Guo Shuai. (2025). LLM-as-a-Judge: A Practical Guide. Towards Data Science

Hendrycks, Dan et al. (2020). Measuring Massive Multitask Language Understanding. arXiv.

Laurer, Moritz et al. (2022). Less Annotating, More Classifying – Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and BERT-NLI. OSF.

Liang, Percy et al. (2023). Holistic Evaluation of Language Models. arXiv.

Lin, Stephanie et al. (2021). TruthfulQA: Measuring How Models Mimic Human Falsehoods. arXiv.

Min, Sewon et al. (2023). FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. arXiv.

Reimers, Nils & Gurevych, Iryna. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv.

Reuel, Anka et al. (2024). BetterBench: Assessing AI Benchmarks, Uncovering Issues, and Establishing Best Practices. arXiv.

Rudinger, Rachel et al. (2018). Gender Bias in Coreference Resolution. arXiv.

Shi, Lin et al. (2024). Judging the Judges: A Systematic Study of Position Bias in LLM-as-a-Judge. arXiv.

Steck, Harald et al. (2024). Is Cosine-Similarity of Embeddings Really About Similarity? arXiv.

Zhang, Tianyi et al. (2019). BERTScore: Evaluating Text Generation with BERT. arXiv.

Zheng, Lianming et al. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv.