KLens

Technical Whitepaper

Version 1.0 · April 17, 2026

# The K-Lens Taxonomy Generation Pipeline

Scaling Document Classification with Large Language Models

**Elevate AI Team**

X-Technology Development Factory (XTDF)

research@i-elevate.com

# Contents

**Abstract**

Transforming unstructured text into organized, meaningful categories is a fundamental challenge in enterprise document management. This paper details the architecture and implementation of the K-Lens Taxonomy Generation Pipeline, a production-grade system designed to automate the discovery, generation, and assignment of complex document taxonomies. Building upon the foundational TnT-LLM framework, we propose several advancements tailored for real-world enterprise applications. We introduce a highly structured, use-case-driven summarization step and optimize prompts for general-purpose documental classification. To anchor AI discovery to predefined business logic, we propose a novel Seed-Guided Cartridge workflow. We also implement multi-level hierarchical structuring and robust Human-in-the-Loop (HITL) breakpoints managed via an expanded compiled state graph. Leveraging the base framework's dual-model strategy, the pipeline pairs heavy reasoning models for taxonomy generation with lightweight models for mass classification. This allows the system to efficiently deliver high-throughput document organization, structured classification prompts, and automated labeled datasets for downstream AI tasks.

# 1 Introduction

In the modern enterprise landscape, the ability to derive intelligence from unstructured data is constrained by the difficulty of accurately organizing documents at scale. As organizations ingest diverse and ever-growing document corpora, the creation of robust, adaptable taxonomies becomes a fundamental bottleneck—one where brittle manual annotation fails to keep pace with operational demands.

Addressing the full spectrum of these challenges, the K-Lens platform is an enterprise AI solution designed to transform unstructured information into traceable, verifiable insights for highly regulated environments, such as public administrations and large enter-

prises. The platform's core architecture allows organizations to instantly process diverse document ecosystems—ranging from lengthy legal contracts and unstructured PDFs to scanned invoices—enabling automated fact-checking, anomaly detection, and seamless integration with existing data lakes via MCP-compliant protocols. However, the success of these advanced downstream features often relies on a critical foundational step: the accurate and automated classification of incoming documents.

Traditionally, in the domain of enterprise text mining [1, 2], organizing massive, unstructured document corpora is bottlenecked by two disparate approaches. The first is labor-intensive manual annotation, which requires domain experts to curate taxonomies and classify data [3], incurring prohibitive costs and time [4]. The second involves unsupervised text clustering (e.g., embedding-based K-Means or topic modeling) [5]. While highly scalable, these unsupervised methods often lack interpretability, resulting in vague clusters that fail to align with concrete business objectives—a problem often likened to "reading tea leaves" [6].

Historically within the K-Lens ecosystem, document classification has suffered from the limitations of the first approach, acting as a heavily manual and iterative bottleneck. Clients typically provide only a vague or incomplete list of expected document classes without precisely defined logical boundaries. Consequently, developers and domain experts are required to manually review small subsets of documents to deduce categories and draft initial class definitions. In production, a large language model (LLM) evaluates the first few pages of an ingested document against these manually drafted descriptions to output a class label, which subsequently triggers other workflows (e.g., class-specific key-value extraction). As new document types naturally emerge, or when inappropriate labeling causes downstream extraction failures, these definitions require continuous, manual developer patching.

The proposed new K-Lens Taxonomy Generation Pipeline addresses these limitations by replacing this brittle, developer-heavy process with an automated, data-driven architecture. Designed as a core feature of the K-Lens platform, the system utilizes LLMs to bridge the gap between human interpretability and algorithmic scalability. It serves multiple purposes: outputting an operational classification prompt crafted on the specific use case for the application, automatically organizing ingested documents into semantic folder structures based on a representative subset of data, and ultimately potentially generating high-volume labeled datasets for subsequent fine-tuning procedures.

The automatically generated classification prompt is rigorously structured. It explicitly defines the core essence, key identifiers, and boundary rules of every class directly derived from the corpus data. This kind of description for each class is generated and progressively refined based on provided documents with the goal of creating effective definitions explicitly resolving ambiguities. This prompt is configured to evaluate the text extracted from an incoming document (or its initial pages) and output a definitive label, seamlessly driving downstream analytical tasks without the need for manual developer intervention.

This paper details the engineering and architectural design of the K-Lens pipeline, clearly

separating the foundational TnT-LLM concepts from our own novel contributions. Specifically, we introduce a use-case-driven summary generation step and refactor prompts with specific instructions to generate detailed and effective descriptions, preserving native domain jargon. To align AI outputs with strict business rules, we introduce seed-guided drift analysis. Finally, we enhance the final outputs with hierarchical structuring and manage the entire workflow through advanced state-based execution routing, allowing granular user feedback injections in the generation loop.

# 2 Background: The Base TnT-LLM Framework

The architecture of our system originates from the foundational principles of **TnT-LLM** (Text Mining at Scale with Large Language Models) [7], a methodology developed to automate the end-to-end generation and assignment of label taxonomies. Traditionally, text mining forces a compromise: manual human annotation yields high interpretability but fails to scale, whereas automated unsupervised clustering (e.g., K-Means on text embeddings) scales infinitely but produces opaque, poorly defined categories. TnT-LLM bridges this gap by utilizing the instruction-following and reasoning capabilities of LLMs to dynamically induce, refine, and apply interpretable taxonomies.

The original framework operates through two interrelated phases, which serve as the baseline logic for our enterprise adaptation:

## 2.1 Phase 1: Taxonomy Generation via Stochastic Optimization

The core innovation of TnT-LLM is treating the discovery of a text taxonomy as an iterative optimization problem, conceptually analogous to learning a Gaussian Mixture Model via Stochastic Gradient Descent (SGD). Because it is computationally infeasible to pass an entire enterprise document corpus into an LLM's context window simultaneously, the framework employs a multi-stage, batch-based reasoning approach:

1. **Stage 1 — Feature Extraction (Summarization):** Analogous to featurization in classic machine learning, an LLM first processes a representative sample of raw documents, generating concise summaries. This normalizes varying document lengths, reduces noise, and extracts salient semantics relevant to a given use-case instruction.
2. **Stage 2 — Iterative Taxonomy Optimization:** The summarized documents are divided into equal-sized minibatches. The framework then executes a sequential chain of zero-shot LLM prompts:
    - *Initial Generation Prompt:* Takes the first minibatch and induces a baseline taxonomy, complete with label names and descriptions.

- *Taxonomy Update Prompt:* Functions as the "gradient update." For each subsequent minibatch, the LLM evaluates how well the current taxonomy fits the new data, identifying coverage gaps or semantic overlaps, and updates, merges, or splits the taxonomy labels accordingly.
- *Review Prompt:* A final pass to verify formatting, resolve ambiguities, and ensure the taxonomy adheres to the requested scale and use-case constraints.

## 2.2 Phase 2: LLM-Augmented Text Classification

Once the taxonomy is finalized, TnT-LLM shifts from discovery to mass application. Rather than deploying expensive, heavy-reasoning LLMs to classify millions of documents in real-time, Phase 2 utilizes the LLM solely as a high-quality data annotator. The LLM applies the Phase 1 taxonomy to a broader corpus sample, generating a rich dataset of "pseudo-labels." These pseudo-labels are then used as ground-truth training data to train lightweight, traditional machine learning models (e.g., Logistic Regression, LightGBM, or Multilayer Perceptrons). This knowledge distillation process yields a highly efficient classifier that can be reliably and cheaply served at scale.

## 2.3 Transition to Enterprise Requirements

While TnT-LLM provides a highly effective theoretical foundation for unsupervised taxonomy generation, its baseline prompt sequences present limitations in strict enterprise environments. Specifically, the base framework assumes a "blank slate" unsupervised discovery process. Furthermore, while the original paper theorizes hierarchical generation and the base codebase introduces a foundational `LangGraph` architecture with basic interrupt mechanisms, it lacks the complex, multi-path routing required for enterprise oversight. To transition this framework into the production-grade **K-Lens Pipeline**, we built upon the repository's foundational `LangGraph` implementation, significantly expanding its baseline directed acyclic graph (DAG) by introducing complex enterprise-specific execution branches: rigid business-logic anchoring (Seed Cartridges), bottom-up multi-level hierarchical structuring, and expanded interactive HITL routing.

# 3 System Architecture

The K-Lens pipeline builds upon the foundational `LangGraph` implementation of the TnT-LLM repository. We expanded its baseline DAG into a highly complex, multi-branching orchestration, significantly extending the robustly typed `State` dataclass to support immutable structural anchors, multi-level hierarchy generation, and advanced state-routing logic. The graph (`graph.py`) orchestrates a series of specialized functional nodes, passing the typed `State` object throughout execution.

The architecture consists of four primary stages:

1. **Ingestion & Distillation:** Raw documents are loaded and sampled (`document_loader.py`). A fast LLM processes each document to generate a high-density summary (`summary_generator.py`), and the corpus is then split into manageable chunks (`minibatches_generator.py`).
2. **Taxonomy Generation & Refinement:** The system checks for the presence of user-defined business categories (the "Seed Cartridge") and routes execution through either a purely Unsupervised Discovery loop (`generate_taxonomy`) or a Seed-Guided Enrichment loop (`ingest_cartridge` followed by strict `update_taxonomy` steps). A feedback loop iteratively refines the taxonomy across minibatches, culminating in a global review pass (`review_taxonomy`).
3. **Hierarchy Generation & HITL Validation:** The pipeline pauses execution, allowing users to interact with the generated base classes. Following human approval, the system constructs a multi-level hierarchical tree (`hierarchy_generator.py`), followed by a second pause for structural validation.
4. **Classification Prompt & Mass Document Labeling:** The finalized taxonomy matrix is compiled into a specialized classification prompt and passed to a high-throughput labeling chain (`doc_labeler.py`).

The high-level execution flow is visualized in Figure 1.

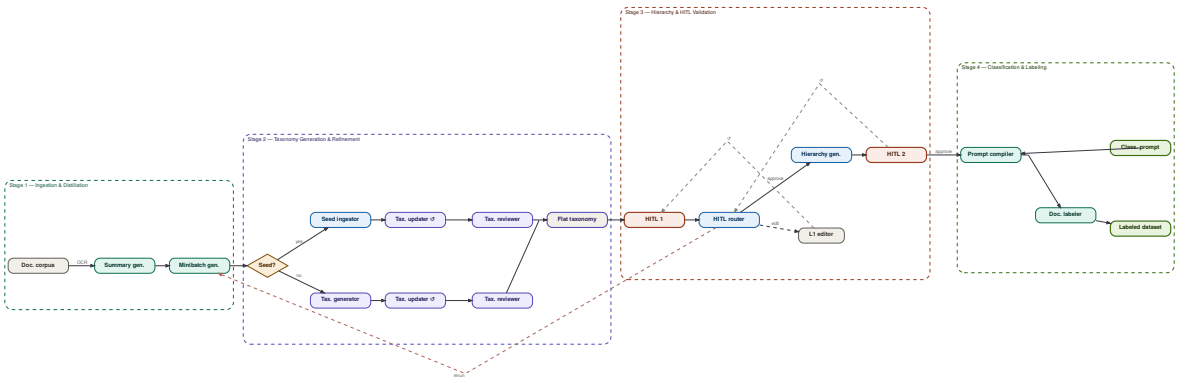


**Figure 1.** *The K-Lens pipeline architecture, illustrating the state-driven orchestration of the document ingestion, taxonomy generation, hierarchy structuring, and final classification labeling nodes.*

# 4 Key Modifications and Contributions

To transition from an academic framework to a production-ready enterprise tool, we introduced several architectural enhancements over the base TnT-LLM framework.

## 4.1 Use-Case-Driven Summarization and Prompt Refactoring

A general refactoring of the system's prompt architecture was carried out to ensure optimal performance in real-world environments. Our refactored summarization prompt dynamically adapts to an upfront use-case definition, instructing the LLM to focus exclusively on extracting precise functional elements (e.g., document typology, key entities, critical metrics) that are genuinely relevant to the specified goal. Furthermore, the taxonomy generation prompts have been customized to enforce clear logical boundaries between document classes, and explicitly configured to retain domain jargon in the original language of the documents (e.g., preserving terms like *DURC* or *modello F24* in Italian texts), ensuring no vital semantic nuance is lost during translation.

## 4.2 The Seed-Guided Cartridge Workflow

Original TnT-LLM assumes a "blank slate" discovery process, but enterprise users rarely require pure unsupervised discovery; they often possess predefined business requirements and rigid expectations. To address this, we introduced the **Semi-Supervised, Guided Enrichment Workflow**.

- If a user provides a raw "Seed Cartridge," the `ingest_cartridge` node normalizes it into a baseline XML taxonomy.
- The system treats these seed categories as **immutable**. The `update_taxonomy` node is strictly forbidden from deleting them or unilaterally merging them. The primary value of this workflow lies in its ability to organically *enrich* these minimal human definitions with real-world complexities—automatically injecting newly discovered domain jargon, metrics, and boundary rules extracted from the provided data into the seed descriptions, achieving an ideal balance between strict human intent and granular AI discovery.

**Key Innovation: Immutable Seed Anchoring**

The Seed-Guided workflow treats all user-defined categories as structurally immutable. The `update_taxonomy` node is explicitly forbidden from deleting or merging them, ensuring that AI discovery enriches—but never overrides—predefined business logic.

## 4.3 Multi-Level Hierarchical Taxonomies

While the original TnT-LLM paper theorizes a top-down approach to hierarchy (re-running generation on subgroups to drill down), this can lead to structural bloat and is generally computationally expensive for enterprise use cases. Instead, K-Lens implements a novel **bottom-up hierarchical grouping** component (`hierarchy_generator.py`). After the Level

1 leaf taxonomy is finalized and approved, a single prompt-based step groups these base classes into a broader, logical multi-level structural graph (Level 3 Broad Categories → Level 2 Sub-Categories → Level 1 Leaves). To prevent structural bloat, the prompt strictly enforces a "no single-child" rule, forcing the LLM to flatten redundant parent nodes.

## 4.4 Human-in-the-Loop (HITL) Integration

While the foundational repository introduced a basic LangGraph interrupt mechanism for binary user feedback (continue vs. modify), enterprise deployments require far more granular control over the graph's state. The K-Lens pipeline advances this by implementing a complex multi-path routing controller (`level_1_router`), parsing natural language feedback to explicitly control state manipulation across three distinct execution paths:

- `DIRECT_EDIT`: Routes to `level_1_editor.py` for localized state mutations (e.g., surgically renaming a class without affecting the rest of the taxonomy).
- `FULL_RERUN`: Purges the taxonomy history state and routes back to the minibatch processing stage for a full structural overhaul.
- `APPROVE`: Advances the graph to the hierarchy generation nodes.

## 4.5 Evolution Logging and Strict Drift Review

To provide operational transparency during iterative batch updates, we introduced an `evolution_log` that generates chronological logs detailing why classes were created or suggested for merging. The `review_taxonomy` node then executes a strict **Drift Analysis**: it evaluates the updated taxonomy against the original Seed Cartridge, detects any dropped user-defined categories, and autonomously reinstates missing classes before allowing human review—ensuring strict compliance with business requirements.

# 5 Implementation Details

## 5.1 State Management and Orchestration

Building upon the base repository's strongly typed `State` dataclass (`state.py`), we significantly expanded the payload to act as the single source of truth passed between LangGraph nodes. Key properties include:

- `documents` and `minibatches`: Managing the active corpus payload.
- `clusters` and `evolution_log`: Utilizing LangGraph's `Annotated` capabilities with custom reduction functions (e.g., `clearable_add`). The `clusters` state tracks the active taxonomy across iterations, while the `evolution_log` acts as a persistent chronological ledger, explaining why specific classes were created or merged and serving as

contextual memory for the LLM across batches to prevent circular reasoning.

- `level_1_route`: Storing the routing decision post-HITL feedback.

## 5.2 LLM Orchestration and Dual-Model Paradigm

Configured in `configuration.py`, the system separates tasks by cognitive load:

- `model` **(Heavy Reasoner):** Defaulting to advanced reasoning models (e.g., `deepseek-reasoner`). Reserved for tasks requiring deep structural logic: taxonomy generation, hierarchical structuring, and complex conflict resolution.
- `fast_llm` **(High-Throughput):** Defaulting to lighter, faster models (e.g., `deepseek-chat`). Utilized for highly parallelized tasks: summarizing hundreds of documents, parsing natural language feedback in the router, and the final mass labeling phase.

## 5.3 Structural Guardrails and Classification Prompt Generation

The prompt templates (`prompts.py`) enforce a rigid three-part logic for every generated taxonomy class—not merely as a formatting requirement, but as an architectural guardrail:

1. **Core Definition:** The functional essence of the category.
2. **Key Identifiers:** Mandatory entities and specific domain jargon.
3. **Boundary Rules:** Explicit instructions resolving edge-case overlaps.

This logic forces the LLM to systematically isolate distinguishing elements and promote clear separations between classes. Its primary value is realized when the taxonomy is finalized: the 3-part structure feeds directly into a ready-to-use operational Classification Prompt, equipping the downstream automated labeler with the precise criteria needed to classify documents without ambiguity.

## 5.4 Pipeline Configuration and Hyperparameters

The K-Lens pipeline exposes several key user-configurable hyperparameters to provide granular control over taxonomy generation:

- **Maximum Taxonomy Size ($N_{max}$):** A hard limit on the maximum number of leaf classes the system may generate. A lower limit forces the heavy reasoning model to generalize and find higher-level abstractions, preventing the structural bloat and over-fragmentation common in unbounded clustering.
- **Minibatch Size ($B$):** The number of document summaries processed during each taxonomy update step. Smaller minibatches result in more frequent, granular updates but

risk localized semantic drift. Larger minibatches provide broader contextual visibility but consume more context window tokens and increase the cognitive load per prompt.

- **Output Language:** A novel addition over TnT-LLM, this parameter decouples the source document language from the output taxonomy language. By default set to English, it instructs the LLM to generate all class descriptions in a unified target language while strictly preserving domain-specific jargon and key identifiers in the native document language—standardizing the resulting classification prompt for global IT systems without losing local lexical precision.

# 6 Evaluation and Results: An Empirical Case Study

To evaluate the efficacy of the K-Lens Taxonomy Generation Pipeline in a real-world enterprise environment, we conducted an empirical case study using a proprietary dataset from a corporate client (*Vega*). Because the primary objective of this system is to replace manual human ontology drafting with automated, semantically coherent classifications, our evaluation focuses on qualitative system utility, structural integrity, and the semantic precision of the generated artifacts.

## 6.1 Experimental Setup

The evaluation corpus consisted of text extracted (using EasyOCR) from 174 unstructured documents representing a cross-section of the client's operations. The system was initialized with a single, high-level use-case configuration: *"Identify and classify Italian legal and administrative documents."*

To evaluate the system under controlled hyperparameter constraints, we configured the pipeline with a **minibatch size of 15 documents** per iterative update step and a hard constraint of maximum **30 generated leaf classes**. The taxonomy **output language was set to English**, ensuring that while the semantic evaluations and extracted entity jargon (e.g., *DURC*, *Modello F24*) remained faithful to the Italian source texts, the resulting structural descriptions and boundary rules were generated in a standardized language for seamless downstream system integration.

The pipeline was executed using `deepseek-reasoner` as the heavy reasoning model for taxonomy generation and hierarchy structuring, paired with `deepseek-chat` for high-throughput distillation. The full generated taxonomies are available in Appendix B.

## 6.2 Unsupervised Discovery Workflow Analysis

In this execution, the system was run without a Seed Cartridge (blank slate) to test its ability to autonomously extract and organize a specialized ontology purely from the latent

semantics of the provided corpus. The pipeline generated a highly detailed flat taxonomy of 30 distinct leaf classes, subsequently mapped into 9 Level-2 categories and 6 Level-3 root domains.

### 6.2.1 Process-Aware Granularity and Lifecycle Mapping

Traditional unsupervised clustering methodologies often group documents by simplistic keyword frequency, resulting in bloated, generic clusters like *"Contracts."* In contrast, the K-Lens pipeline demonstrated "process-aware" granularity, successfully mapping the temporal and legal lifecycles of complex business operations.

Within real estate transactions, the system autonomously disaggregated the lifecycle of a property sale into its chronological components:

1. **Class 10 (Conditional Preliminary Real Estate Sale Agreements):** Defined specifically as private preliminary contracts (*compromessi*) that precede a final sale, heavily reliant on suspensive conditions and deposits (*caparra confirmatoria*).
2. **Class 11 (Definitive Real Estate Sale Deeds):** Defined as unconditional transfers of full ownership (*rogito notarile*) executed before a notary.

Similarly, the pipeline mapped the lease lifecycle, generating distinct, mutually exclusive classes for Preliminary Commitments (Class 5), Core Lease Contracts (Classes 1 and 2), Asset Handover Reports (Class 6), and subsequent Contractual Modifications (Class 4). This temporal granularity is vital for enterprise data lakes, as it allows downstream systems to trigger vastly different extraction workflows (e.g., extracting deposit amounts from a preliminary agreement versus extracting final transfer taxes from a definitive deed).

### 6.2.2 Lexical Precision and Entity-Driven Disambiguation

The generated *Key Identifiers* demonstrate the pipeline's capacity to isolate the precise domain jargon and entities required to distinguish highly similar document types. In Italian administrative datasets, the noun *"Visura"* (Registry Extract) is ubiquitous, yet its semantic target dictates entirely different business logic.

The heavy reasoning model successfully split this concept based on underlying entity features extracted during the distillation phase. For **Class 13 (Cadastral Registry Extracts — *Visura Catastale*)**, the AI identified real-estate-specific markers: *foglio, particella, subalterno, rendita*. Conversely, for **Class 14 (Business Registry Extracts — *Visura Camerale*)**, it identified corporate identity markers: *Registro Imprese, REA, Partita IVA, amministratore*. By structurally forcing the AI to output these exact features as *Key Identifiers*, the prompt compilation node equips the downstream mass-labeling classifier with the exact semantic details required to prevent misclassification.

### 6.2.3 Resolving ``Intent vs. Execution'' Ambiguities

A frequent cause of downstream automation failure in enterprise document processing is an AI confusing an instruction or obligation document with its corresponding proof of execution. The K-Lens pipeline demonstrated systemic robustness in resolving these ambiguities.

This is most clearly evidenced in the system's handling of tax and insurance documents. The pipeline successfully decoupled tax obligations into the instruction form (**Class 21: Consolidated Tax Payment Forms / Modello F24**) and the proof of payment (**Class 22: Official Tax Payment Receipts / Quietanza di Versamento**). It replicated this logical decoupling for insurance, splitting the **Business Insurance Policies (Class 17)** from the **Insurance Premium Payment Receipts (Class 18)**. For Class 22, the autonomously generated boundary rule forcefully states: *"Do NOT use for the payment instruction form (Modello F24)...This category is exclusively for the final, official receipt proving a tax payment has been settled."*

### 6.2.4 Multi-Level Hierarchical Coherence and Adaptive Structuring

While a flat taxonomy of 30 specialized leaf nodes provides the necessary granularity for algorithmic classification, navigating such a list presents cognitive friction for human operators. To bridge this gap, the `hierarchy_generator.py` component maps the generated leaf classes into a multi-level directed graph with two primary structural advantages:

**1. Functional Abstraction over Lexical Similarity:** The reasoning model grouped classes based on operational intent rather than keyword frequency. For instance, the system structurally decoupled real estate documents into **Level 3_1 (*Real Estate Transactions*)**—capturing the legal movement of assets—and **Level 3_6 (*Property and Legal Verification*)**—grouping compliance reports and cadastral extracts. This indicates that the hierarchy is constructed around underlying business workflows rather than superficial lexical overlaps.

**2. Adaptive Depth and Redundancy Mitigation:** Strict prompt-level guardrails enforce adaptive structural depth. For dense, multi-faceted domains like **Level 3_3 (*Tax Compliance*)**, the model generated distinct Level-2 sub-categories separating *Tax Registration* from *Tax Payment*. Conversely, for narrower domains like **Level 3_4 (*Banking Operations*)**, the model dynamically flattened the tree, attaching leaf nodes directly to the Level-3 root.

By combining semantic grouping with strict structural constraints, the pipeline successfully translates a dense classification matrix into a pragmatic, human-navigable knowledge graph, eliminating the need for manual hierarchical curation.

## 6.3 Seed-Guided Cartridge Workflow Analysis

Enterprise users rarely approach document classification as a purely academic, blank-slate clustering problem; they typically possess predefined business requirements, legacy

system schemas, or specific analytical targets. To evaluate the pipeline's capability to anchor unsupervised AI discovery to rigid human intent, we executed the Seed-Guided Workflow using an input cartridge of 26 concise, human-defined classes (e.g., *Commercial Lease Contract*, *F24 Tax Payment Receipt*, *Termination Notice Document*).

The pipeline ingested these initial minimal definitions (detailed in Appendix A) as a structured baseline, treating them as immutable anchors, then processed the evaluation corpus through the iterative minibatch loop.

### 6.3.1 Autonomous Structural Enrichment and Lexical Injection

The most immediate value added by the pipeline is the transformation of simplistic human definitions into detailed, robust classification prompts. During the ingestion phase, the pipeline autonomously mined the document summaries and injected native terminology directly into the seed structures. For example, the user's minimal definition for **Seed 2 (Energy Performance Certificate (APE))** was enriched with specific *Key Identifiers* such as *"validity period, energy class (e.g., A4, E), performance indices, climatic zone."*

Furthermore, the system autonomously generated explicit *Boundary Rules* to protect seeds from false positives. For **Seed 3 (Insurance Documentation)**, the human seed read only: *"Documentation of insurance policies covering risks and commercial activities."* The pipeline expanded this to isolate the policy contract from the payment receipt, enforcing the rule: *"The policy document, coverage declaration, or schedule. For proof of premium payment, see Insurance Premium Receipt."* This autonomous structural enrichment eliminates the need for developers to manually draft and patch prompts.

### 6.3.2 Data-Driven Scope Adaptation and Assumption Correction

While the system is strictly forbidden from deleting user-provided seed classes, it is engineered to autonomously correct human assumptions based on the empirical reality of the ingested data. For example, the user provided Seed 8 as *"Commercial Lease Contract."* During batch processing, the heavy reasoning model detected residential lease documents (*contratto di locazione ad uso abitativo*) that violated the user's "commercial" constraint. The `evolution_log` captured the autonomous scope adaptation: *"Renamed seed class 8 from 'Commercial Lease Contract' to 'Lease Contract' to encompass both commercial and residential property rentals."* Similarly, it expanded Seed 16 (*Technical Drawings and Plans*) to encompass judicial appraisal reports, renaming it *"Technical Documentation and Appraisal Reports."*

### 6.3.3 Autonomous Discovery of Operational Blind Spots

In complex administrative environments, human experts frequently overlook niche or highly specialized document variants. Despite the user providing 26 distinct seeds, the pipeline generated 4 entirely new, mutually exclusive classes to capture critical legal events present in the data. Most notably, the AI recognized that the user's generic *Operating License Documentation* seed was insufficient for the highly regulated energy sector, autonomously generating **Class 30: Customs and Excise Authorizations for Fuel Distribution**—explicitly

noting in the logs that this is a *"specific public license from the Agenzia delle Dogane, distinct from municipal operating licenses."* It also generated a new **Class 29 (Preliminary Contract)** to capture preliminary lease commitments entirely absent from the human seed cartridge.

### 6.3.4 Dynamic Conflict Resolution and Cognitive Traceability

A significant architectural advantage of the minibatch loop is its capacity for iterative refinement. Analyzing the `evolution_log`, we can observe the AI dynamically resolving complex semantic conflicts, providing a level of "cognitive traceability" that black-box embedding clusters cannot offer.

This is illustrated by the system's handling of contract transfers. During an early batch, the system suggested merging a new class for *Notification of Contract Transfer* with Seed 13 (*Letter of Assignment*):

> *"MERGE_SUGGESTION: Documents for notifying contract transfers (new class) and assignments of business branch leases (seed class 13) are functionally similar. Consider merging into a broader 'Contract Transfer Notification' category."*

Upon processing a subsequent minibatch with more nuanced examples, the heavy reasoning model recognized a critical legal distinction and autonomously reversed its decision:

> *"LOG_CORRECTION: New data confirms a functional distinction: seed class 13 (Assignment Letter) is a payment instruction following a transfer, while new class 29 (Transfer Notification) is the formal notice of the legal event. Merge suggestion retracted."*

This granular, log-driven behavior validates the system's core design proposition: it pairs the deterministic safety of state-graph orchestration with the advanced, iterative reasoning capabilities of modern LLMs.

### 6.3.5 Adaptive Integration into the Hierarchical Matrix

Despite the output taxonomy being a hybrid of human-defined seeds and newly discovered AI classes, the `hierarchy_generator.py` component successfully integrated the 30 final leaves into a cohesive, multi-level tree. The AI logically mapped these disparate nodes into four overarching Level-3 domains: *Contractual Documents*, *Financial and Payment Documents*, *Official and Regulatory Documents*, and *Property and Technical Documentation*. This unified structuring proves that the Seed-Guided workflow actively weaves disparate findings into a holistic, logically sound enterprise ontology ready for immediate downstream deployment.

# 7 Discussion

## 7.1 Design Trade-Offs

- **Heavy Reasoning vs. Inference Cost:** Iteratively updating the taxonomy across minibatches using a heavy reasoning model can consume significant compute. However, compared to the operational cost of employing human domain experts to curate taxonomies over weeks, the compute cost is negligible. Furthermore, the taxonomy generation pipeline is generally run only once per use case, keeping the overall compute expense remarkably low.
- **Strict Seed Compliance vs. Semantic Truth:** In the Seed-Guided workflow, the strict rule against deleting user seeds can result in "empty" classes if the actual document corpus contains no matching data. We accepted this trade-off to prioritize business-logic alignment over pure algorithmic accuracy.

## 7.2 Limitations and VLM Mitigation

A primary limitation of text-based mining pipelines is their dependence on the quality of upstream OCR extraction. Poor OCR can obscure vital metrics and entities, degrading the summaries. Processing large document minibatches also presses against the context window limits of current models.

**Current & Future Mitigation**

This limitation can be mitigated by utilizing Vision Language Models (VLMs) during the initial `Summary Generator` stage. The pipeline architecture can be seamlessly adjusted to pass raw PDFs or full document images directly to a VLM, reliably extracting visual cues (e.g., stamps, signatures, tabular financial data) into highly condensed text summaries. The subsequent taxonomy generation and batching loops can then be processed by standard, text-only LLMs operating safely within their context windows.

## 7.3 Future Work

Currently, Phase 2 relies on the `fast_llm` to perform the final mass classification. Moving forward, the K-Lens pipeline acts as a data engine for **active learning** and **model distillation**. The high-quality labels generated by the system can be directly exported to train and fine-tune smaller, custom embedding models or lightweight classifiers (e.g., via QLoRA on smaller open-source models), further decoupling the operational labeling phase from third-party API dependencies and maximizing privacy and throughput.

# 8 Conclusion

The K-Lens Taxonomy Generation Pipeline successfully adapts the academic TnT-LLM framework into a robust, deterministic enterprise system. By introducing specialized summary prompt generation, seed-guided drift analysis, multi-level hierarchies, and advanced state-graph-based human oversight, the system effectively automates one of the most resource-intensive aspects of text mining. It provides the K-Lens platform with the capability to immediately impose semantic order onto chaotic document lakes. By replacing vague human intuition with data-driven, structured classification prompts, the pipeline guarantees reliable downstream extraction workflows, while simultaneously serving as an automated data factory for future AI model training.

# Acknowledgements

The authors thank the Elevate team and Klens platform contributors for their support, and the *Vega* client for providing the proprietary evaluation corpus used in this study.

# References

[1] A. Hotho, A. Nürnberger, and G. Paaß, "A brief survey of text mining," *Journal for Language Technology and Computational Linguistics*, vol. 20, no. 1, pp. 19–62, 2005.

[2] A.-H. Tan et al., "Text mining: The state of the art and the challenges," in *Proceedings of the PAKDD 1999 workshop on knowledge discovery from advanced databases*, vol. 8, 1999, pp. 65–70.

[3] B. B. Cambazoglu, L. Tavakoli, F. Scholer, M. Sanderson, and B. Croft, "An intent taxonomy for questions asked in web search," in *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, 2021, pp. 85–94.

[4] P. Thomas, S. Spielman, N. Craswell, and B. Mitra, "Large language models can accurately predict searcher preferences," *arXiv preprint arXiv:2309.10621*, 2023.

[5] C. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," *Mining text data*, pp. 77–128, 2012.

[6] J. Chang, S. Gerrish, C. Wang, J. Boyd-Graber, and D. Blei, "Reading tea leaves: How humans interpret topic models," *Advances in neural information processing systems*, vol. 22, 2009.

[7] M. Wan et al., "Tnt-llm: Text mining at scale with large language models," *arXiv preprint arXiv:2403.12173*, 2024.

# A Appendix: Initial Normalized Seed Cartridge

The following list details the initial, human-defined Seed Cartridge provided as input for the Seed-Guided Cartridge Workflow evaluation (Section 6.3). These minimal definitions served as the immutable structural anchors that the pipeline subsequently expanded, enriched, and supplemented with data-driven discoveries.

1. **Rent Adjustment Communication**: Communications regarding ISTAT index variations and new rental amounts for lease agreements.
2. **Energy Performance Certificate (APE)**: Energy Performance Certificate containing technical data, consumption details, and system information for buildings.
3. **Insurance Documentation**: Documentation of insurance policies covering risks and commercial activities.
4. **Notarial Certificate**: Notarial certification related to the lease of a business branch.
5. **Tax Agency Declaration for Business Branch Transfers (Cessione ADE)**: Declarations to the Revenue Agency (Agenzia delle Entrate) for the transfer of business branches.
6. **Business Branch Lease Contract**: Contract for the lease of business branches.
7. **Service Supply Agreement**: Agreements for the supply of services, including costs and clauses.
8. **Commercial Lease Contract**: Lease contract for commercial properties or land.
9. **Contract for Management of Fuel Distribution Plants**: Contract for the management of fuel distribution plants.
10. **Termination Notice Document**: Formal letters for the termination of contractual relationships.
11. **Contribution Regularity Certificate (DURC)**: Certificate of contribution regularity issued by INPS and INAIL for social security and insurance compliance.
12. **Financial Guarantee Documentation**: Financial documentation such as guarantees, deposits, and bank guarantees.
13. **Letter of Assignment for Business Branch Lease**: Letter for the assignment of a business branch lease.
14. **Production Bonus Letter for Plant Managers**: Communications regarding production bonuses for plant managers.
15. **Contract Amendment Document**: Amendment acts, resolutions, or proxies for lease contracts.
16. **Technical Drawings and Plans**: Technical and graphical drawings of buildings or areas.

17. **Contract Extension Letter**: Letters extending the duration of a contract.
18. **Insurance Premium Receipt**: Receipts for insurance premium payments.
19. **Other Source Receipts (e.g., RLI12, Contract Registration Acts)**: Receipts like RLI12 and contract registration acts from the Revenue Agency (Agenzia delle Entrate).
20. **F24 Tax Payment Receipt**: Receipts for tax and duty payments using the F24 form.
21. **Private Deed for Tobacco Shop Management**: Agreements for the operational management of tobacco shops.
22. **Handover Report with Equipment List and Meter Readings**: Handover reports documenting the delivery of premises, including equipment lists and meter readings.
23. **Chamber of Commerce Company Extract (Visura Camerale)**: Anagraphic document of a company extracted from the Business Register (Registro Imprese).
24. **Cadastral Data Extract for Commercial Properties (Visura Catasto)**: Cadastral data and income details for commercial properties.
25. **Derogation or Concession for Facilitated Rent**: Document specifying new conditions for facilitated or concessional rent.
26. **Operating License Documentation**: Authoritative documentation permitting the exercise of commercial activities.

# B Appendix: Full Generated Taxonomies

## B.1 Unsupervised Discovery Workflow Taxonomy

1. **Commercial and Business Lease Contracts**

   **Core Definition**: Legally binding agreements for the rental of commercial or business-use real estate, including fixed properties, land for advertising, or business branches, and including subleases (subaffitto). Such contracts may be privately signed or authenticated by a notary.

   **Key Identifiers**: contratto di locazione (immobiliare, commerciale, di ramo d'azienda, subaffitto), canone (annuo, mensile), durata, fideiussione bancaria, adeguamento ISTAT, clausola risolutiva, foro competente, trascrizione.

   **Boundary Rules**: Do NOT include residential leases, lease modifications, or mere notifications about a lease. This category is for the primary, signed contract establishing the lease relationship.

2. **Residential Lease Contracts**

   **Core Definition**: Legally binding agreements for the rental of residential properties for

dwelling purposes, governed by Italian tenancy law.

**Key Identifiers**: contratto di locazione ad uso abitativo, canone, durata (3+2, 4+4), adeguamento ISTAT, cauzione, fideiussione, manutenzione ordinaria, divieto di sublocazione.

**Boundary Rules**: Do NOT include commercial leases, lease modifications, or tax registration forms. This category is for the main residential lease contract, not ancillary documents.

3. **Gratuitous Loan for Use Contracts (Comodato d'uso)**

   **Core Definition**: Contracts granting the gratuitous use of property (typically real estate) for a defined period, without rent payment, often linked to a service agreement.

   **Key Identifiers**: contratto di comodato d'uso, comodante, comodatario, durata, rinnovo tacito, preavviso di recesso, obblighi di manutenzione, pagamento utenze, divieto di subcomodato.

   **Boundary Rules**: Do NOT include lease contracts (which involve rent) or service contracts alone. This category is for non-remunerated use agreements, usually ancillary to another commercial relationship.

4. **Lease Contract Modifications and Amendments**

   **Core Definition**: Formal documents that modify, extend, terminate, or otherwise alter the terms of an existing lease contract (including subleases).

   **Key Identifiers**: modifica di contratto, proroga, risoluzione consensuale, cessione anticipata, canone agevolato, compenso forfettario, verbale di consegna, nuova validità.

   **Boundary Rules**: Do NOT use for the original lease contract or for simple informational notifications. This category is for documents that directly alter the financial, temporal, or legal status of a standing lease.

5. **Preliminary Lease Agreements and Commitments**

   **Core Definition**: Preliminary contracts or written commitments to enter into a future lease agreement, such as a *scrittura di impegno alla stipula del contratto di affitto*, which may involve the payment of a security deposit.

   **Key Identifiers**: scrittura di impegno, preliminare di affitto, deposito cauzionale, caparra, condizione sospensiva, termine per la stipula.

   **Boundary Rules**: Do NOT include the definitive lease contract (category 1 or 2) or lease modifications (category 4). This category is for preliminary agreements that precede the final lease execution.

6. **Asset Handover and Inventory Reports (Verbale di Consegna)**

   **Core Definition**: Formal reports recording the physical transfer and inventory of assets (e.g., furnishings, equipment) upon execution of a lease or business transfer.

**Key Identifiers**: verbale di consegna, inventario, stato dei locali, contatori (acqua, gas, luce), letture finali, chiavi, elenco beni mobili, firme delle parti, PDR (gas).

**Boundary Rules**: Do NOT use for the main lease contract or contractual amendments. This category is for the ancillary document detailing the transferred assets and their condition at handover.

7. **Technical Annexes and Plans**

   **Core Definition**: Technical documents such as floor plans (planimetrie), site plans, or other graphical annexes that are integral parts of a contract and describe the physical characteristics of the property.

   **Key Identifiers**: planimetria, allegato tecnico, mappa, pianta, dati catastali (foglio, particella, subalterno), superficie, scala.

   **Boundary Rules**: Do NOT use for the main contract (categories 1, 2, 10, 11) or for inventory reports (category 6). This category is specifically for technical drawings and plans attached to agreements.

8. **Formal Business Notifications and Communications**

   **Core Definition**: Official letters, PEC communications, or formal notifications regarding changes in business relationships, corporate status (e.g., mergers, incorporations), or contractual matters that inform but do not themselves alter a contract.

   **Key Identifiers**: comunicazione, notifica, fusione per incorporazione, subentro, cessione, variazione dati sociali, nuovo contraente, istruzioni di pagamento.

   **Boundary Rules**: Do NOT use for signed contractual amendments (category 4) or for the original contracts. This category is for one-way informational notices that may require acknowledgment but not signature to become effective.

9. **Bank Guarantees (Fideiussioni Bancarie)**

   **Core Definition**: Standalone guarantees and their formal amendments issued by banks to secure the contractual obligations (e.g., lease payments) of a debtor, making the bank liable up to a specified amount.

   **Key Identifiers**: fideiussione bancaria, polizza fideiussoria, beneficiario, debitore principale, garante, importo garantito, durata, rinnovo tacito, clausola di rinuncia al beneficio della preventiva escussione, addendum.

   **Boundary Rules**: Do NOT include insurance policies or personal guarantees. This category is for formal bank guarantees and their amendments, not clauses within a main contract.

10. **Conditional Preliminary Real Estate Sale Agreements**

    **Core Definition**: Private preliminary contracts (compromessi) for the future sale of real estate, conditional upon the fulfillment of specified suspensive conditions within a defined term.

**Key Identifiers**: contratto preliminare di compravendita, condizione sospensiva, caparra confirmatoria, termine finale di avveramento, autorizzazioni, visure catastali, certificato di destinazione urbanistica, promittente venditore/promissario acquirente.

**Boundary Rules**: Do NOT use for definitive sale deeds, lease agreements, or other contract types. This category is for conditional promises to sell/purchase that precede the final notarial deed.

## B.2 Seed-Guided Cartridge Workflow Taxonomy

The Seed-Guided execution generated a 30-class hybrid taxonomy comprising 26 enriched human seeds and 4 autonomously discovered classes. Each class follows the same three-part structure (Core Definition / Key Identifiers / Boundary Rules), with the `evolution_log` annotating each enrichment decision.

Selected enriched examples are documented in Section 6.2.

**Note on Taxonomy Completeness**

Classes 11–30 for both workflows may be appended above following the same three-part structure. The classes presented illustrate the breadth and precision of the system's output. Contact `research@i-elevate.com` for the full machine-readable taxonomy export.