

Q-CAT: A Non-Parametric Query Filtering System for Prohibited Content in Dynamic Enterprise Environments

Juheon Kang¹ Eunkyung Choi^{1*} Hongseok Oh^{1*} Hyunseo Shin^{1*}
JuYoun Son² Yunjin Bae² Wonseok Hwang^{1†}

¹University of Seoul ²Samsung SDS

{2715wngjs, rmarud202, cxv0519, hseo98, wonseok.hwang}@uos.ac.kr

{juyoun77.son, yunjin.bae}@samsung.com

Abstract

Large Language Models (LLMs) are increasingly being adopted in enterprise environments, where employees may intentionally or accidentally submit prompts containing information that could cause legal or economic harm if leaked. Organizations may define categories of information (policies) that are prohibited from being submitted to commercial LLMs, but these definitions can vary between companies and evolve over time. This variability makes parametric classifiers less practical, as they require retraining whenever policies change. Here, we present Q-CAT, a non-parametric policy enforcement system that classifies user queries based on given categories of information that can change dynamically. The system operates by matching embedded user queries to synthetically generated queries in the database. This design enables low-cost policy updates through simple database modification, without retraining, while maintaining low latency. A comparison with strong prompt-based LLM baselines shows that Q-CAT delivers competitive performance in static environments, while offering practical advantages in dynamic policy settings, where policies change and the Query Pool can be updated from logs. Moreover, Q-CAT is up to two orders of magnitude faster than comparable-performance baselines in our experimental setup.

1 Introduction

Large Language Models (LLMs) have rapidly become indispensable in modern enterprise environments. Employees rely on LLMs for tasks such as drafting documents, summarizing internal materials, and making decisions (Brachman et al., 2024). However, this widespread use also introduces the risk of sensitive business information being leaked. For example, sensitive information may be exposed

through traffic interception or by commercial LLM service providers (Ramrakhiyani et al., 2025; Wu et al., 2025). Such risks should be mitigated at the system-design level while balancing safety and usability (Lu et al., 2025). A practical approach is to selectively filter only queries that pose a risk of information leakage (Ramrakhiyani et al., 2025). Designing such systems for real-world enterprise environments requires several key considerations. First, the system must provide sufficiently low latency to minimize the delay in response due to the query filtering. Second, the system should be able to operate efficiently under limited computational budgets. Finally, the system should support efficient adaptation to evolving policies.

In business settings, information considered risky to disclose can change over time or may not be clearly defined. For example, in an IT company, product specifications may be treated as prohibited information during the early stages of new product development, but may later become permissible as the product approaches release and the specifications are made public. To address these three considerations, we develop Q-CAT¹, a non-parametric query filtering system.

Q-CAT first processes governance policies that define which types of information are prohibited and which are explicitly allowed (Fig. 1). Using the given policies, the system generates a set of synthetic user queries using a small local language model while systematically iterating over combinations of query formats and internal usage contexts to cover diverse departments and interaction patterns within the organization. Next, to ensure the quality of the generated queries, another language model is used to produce evaluation rubrics and to assess each query accordingly. Once validated, the queries are stored in a database (Query Pool).

*Equal contribution. Author order is alphabetical.

†Corresponding Author

¹Query Classification and Assessment with Training-free method

Finally, for a given user query, Q-CAT retrieves the top-5 nearest synthetic queries from the Query Pool based on cosine similarity, implemented as inner-product search over normalized embeddings. It then performs majority voting over their labels to classify the input query as Prohibited or Allowed. Because each retrieved query is mapped to a specific policy, Q-CAT can also identify which policy the input query violates.

To address user queries from unseen domains or to handle failure cases that frequently arise in business environments, Q-CAT is designed to allow administrators to directly add, remove, or modify queries in the Query Pool without requiring any additional model training. This design enables rapid and simple correction of misclassifications.

The key advantages of our non-parametric approach are summarized as follows:

- Compared with strong baselines based on large reasoning models ($\gtrsim 8B$), Q-CAT achieves competitive performance while being two orders of magnitude faster.
- When governance policies change, Q-CAT requires no large-scale (re)training as the administrator can simply update the Query Pool.
- We enable post-deployment correction of misclassifications through Query Pool updates, reducing recurring errors and improving accuracy over time.

2 Related Work

2.1 General Safety Guardrails

The rapid deployment of LLMs has driven the development of safety guardrails for detecting harmful or policy-violating content. A major line of work trains classifier-style guard models that evaluate risks in user prompts and model responses according to broad, general-purpose safety taxonomies (e.g., hate speech, violence, sexual content). Representative guardrail systems include the LlamaGuard series (Inan et al., 2023; Meta Llama Team, 2024; Grattafiori et al., 2024), QwenGuard (Zhao et al., 2025), and WildGuard (Han et al., 2024). These models are fine-tuned to identify violations across standardized harm categories and are typically deployed as input/output filters around LLMs.

While effective for general safety moderation, such approaches are inherently constrained by fixed taxonomies.

2.2 User-Defined and Context-Aware Policy Compliance

Recent research has moved beyond static safety taxonomies toward enforcing user-defined and context-aware policies. In the domain of autonomous agents, PolicyGuardBench and the PolicyGuard-4B model study whether web agents comply with given policies over long action trajectories, emphasizing the detection of violations across extended sequences of actions (Wen et al., 2025).

Training-free methods have also been proposed for real-time LLM policy compliance detection by leveraging internal representations conditioned on textual policy descriptions (Rachmil et al., 2026).

Complementary efforts focus on LLM behavior in sensitive, real-world domains. CoPriva (Chang et al., 2025) introduces a benchmark for evaluating whether LLMs comply with user-specified security policies—particularly confidentiality constraints—in enterprise and governmental settings. DynaGuard and DynaBench (Hoover et al., 2025) further explore dynamic guardian models and datasets that supervise LLM outputs according to user-defined policies, supporting adaptive and interactive policy enforcement.

Unlike these prior studies, which focus on whether LLM responses leak sensitive information, our work focuses on detecting whether user queries themselves contain information prohibited by defined policies via a non-parametric approach.

3 Task

In enterprise environments, a query is considered *sensitive* if it contains information that could cause financial loss, legal liability, or regulatory violations when disclosed to external parties. Unlike public safety settings, such sensitivity is not defined by universal norms but by organization-specific governance policies that evolve over time.

Our objective is to build a system that can (1) detect whether a user query contains policy-defined information, (2) identify its type, and (3) operate efficiently under limited computational budgets as policies evolve.

The policies describe categories of prohibited information that employees are not allowed to submit into commercial LLMs. Individual policies can include, but are not limited to, definitions, reasons, background context, types of risk, and common usage examples.²

²We initially classified queries based on user actions (inten-

Similarly, some policies define allowed information that is explicitly permitted. Allowed information can help simplify the classification task for commonly used internal content or separate borderline cases that might otherwise be misclassified as prohibited. For example, while employees’ personal information may be categorized as sensitive, internal corporate email addresses may be explicitly defined as not sensitive.

4 Benchmark

We construct two benchmarks for the prohibited query detection task: (i) a high-quality **Core Set** of 750 queries and (ii) an expanded **Large Set** of 7,482 queries. The policies are determined based on the analysis of real-world queries.

Core Set The Core Set is designed to evaluate the system’s ability to distinguish prohibited and allowed queries under varying semantic conditions, and is structured along two axes.

First, queries are divided along a primary axis of *prohibition*: **Prohibit** (PR) consists of queries that contain policy-sensitive information and are therefore prohibited.

Second, the remaining queries are *allowed* and are further structured along two additional axes: (i) their difficulty level, and (ii) whether they contain policy-related information. This results in four subtypes within the allowed set. **Hard Allowed** (HA) contains allowed queries that include policy-related information and are semantically close to PR, making them difficult to distinguish from prohibited queries. **Easy Allowed** (EA) contains allowed queries that include policy-related information but are comparatively easier to separate from PR. **Hard Empty** (HE) contains queries that are topically related to PR categories but explicitly exclude sensitive content, making them challenging despite not containing policy-related information. **Easy Empty** (EE) is a non-policy control set consisting of everyday queries unrelated to organizational information categories.

For PR, HA, and EA, we define 10 information categories per type and write 15 queries per categories), but this approach resulted in ambiguity. For example, consider the query: "We have an M&A discussion scheduled with company A on March 12th. Can you suggest about 5 welcoming greetings for the meeting?" The action itself is simply requesting greeting ideas, which is harmless, but the query contains sensitive information, the scheduled M&A discussion. Thus, we design the filtering system under the assumption that (governance) policies specify which types of *information* should be filtered.

Core Set Statistics (750)			
Type: Prohibit (PR)	# Policies: 10	Queries / Policy: 15	Total Queries: 150
Policy Information (Korean: English)			
공급 계약 조건: Supply contract terms			
관리자 계정 정보: Admin account credentials			
소송 관련 문서: Litigation documents			
신제품 출시 일정: New product launch schedule			
업무 결과 요약: Business outcome summary			
연구 진행 노트: R&D progress notes			
자산 관리 대장: Asset inventory ledger			
직원 급여 정보: Employee compensation information			
직원 징계 이력: Employee disciplinary records			
팀 주간 회의 안건: Weekly team meeting agenda			
Type: Hard Allowed (HA)	# Policies: 10	Queries / Policy: 15	Total Queries: 150
Policy Information (Korean: English)			
고객클레임: Customer complaint records			
논문 초록: Academic paper abstract			
보도자료 원문: Original press release text			
복리후생 안내: Employee benefits information			
서버 IP 주소: Server IP address			
신입직원 온보딩 자료: New employee onboarding materials			
제품 가격표: Product price list			
제품 사양서: Product specification sheet			
직원 성과평가 등급: Employee performance rating			
프로젝트 완료 보고서: Project completion report			
Type: Easy Allowed (EA)	# Policies: 10	Queries / Policy: 15	Total Queries: 150
Policy Information (Korean: English)			
FAQ 문서: FAQ document			
기술 스택 소개: Technology stack overview			
사내 시스템 접속 가이드: Internal system access guide			
사용자 매뉴얼: User manual			
이용약관: Terms of service			
채용 공고 내용: Job posting details			
출장 신청 절차: Business trip request procedure			
특허 출원 내용: Patent application details			
회의실 예약 방법: Meeting room reservation procedure			
휴가 신청 방법: Leave application procedure			
Type: Hard Empty (HE)	–	–	Total Queries: 150
Type: Easy Empty (EE)	–	–	Total Queries: 150

Table 1: Core Set statistics for sensitive query detection (750 queries). The Core Set comprises five query types (PR, HA, EA, HE, EE), each containing 150 queries. Example queries are shown in Appendix L.

egory ($10 \times 15 = 150$). For HE, each query is mapped to a related PR category while avoiding direct sensitive disclosure. EE is constructed as a general-purpose conversational control set. Detailed statistics are provided in Table 1, and example queries with the corresponding policy categories are shown in Appendix L.

Only PR is labeled prohibited (150/750), while HA, HE, EA, and EE are allowed (600/750), forming a 1:4 class ratio. We preserve this skew to reflect realistic enterprise usage—where benign queries dominate—and to evaluate both detection of rare sensitive requests and false-positive control on the majority safe class.

Construction Protocol The PR, HA, and EA subsets of the Core Set originate from real user queries provided by an enterprise, which we use to guide category coverage and linguistic diversity. Real queries are first manually de-identified, then augmented with GPT-5/5.2 under explicit policy conditioning. All resulting queries undergo full human verification and annotation, and validated examples are reused as few-shot prompts for further augmentation. This iterative loop is repeated until the dataset reaches stable quality and diversity. The

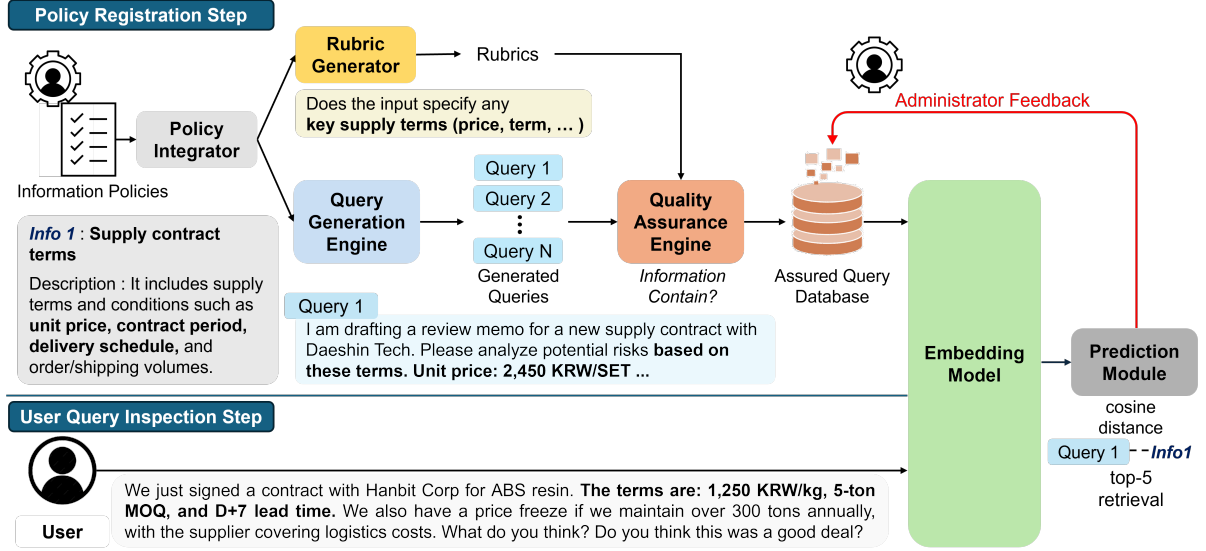


Figure 1: Overview of Q-CAT.

benchmark and the Query Pool are constructed under deliberately separated conditions: benchmark queries are generated with GPT-5/5.2 by a team that did not share generation guidelines with the team building the Query Generation Engine, which uses Qwen3-32B. This separation, together with the MMD analysis (Gretton et al., 2012) in Appendix D, helps prevent the evaluation from being self-referential. For EE, we sample 180 Korean conversational queries from WildChat (Zhao et al., 2024) and manually filter out non-Korean/mixed-language queries and any items that appear policy-relevant, resulting in the final 150 EE queries.

Large Set To better capture the diversity of real-world query patterns, we additionally construct the Large Set, an approximately $10\times$ expansion of the Core Set (7,482 queries) generated through iterative GPT-5.2-based augmentation and sampling from Core Set examples, followed by full human verification, while strictly preserving the policy boundaries established in the Core Set. The Large Set also serves as a source of fresh, never-reused evaluation queries in Appendix H.

5 Methods

Fig. 1 illustrates our system. The pipeline has two stages: a *Policy Registration Step*, where an administrator registers policies and populates the Query Pool with validated synthetic queries, and a *User Query Inspection Step*, where incoming queries are classified against the registered policies via retrieval. A component-level summary of all modules

is given in Table 2, and a formal specification of the components and the inference and update procedures is provided in Appendix A.

Policy Integrator Before registering a new policy, the Policy Integrator embeds its name and description and retrieves the top- k most similar existing policies. The candidates are presented to the administrator, who decides whether to register, merge, or revise the new policy.

Rubric Generator For each registered policy, Qwen3-32B generates six rubrics, each phrased as a binary question probing whether a query contains the policy-defined information. A predefined default rubric set is available as a fallback. Examples of generated rubrics are provided in Appendix K.

Query Generation Engine (QGE) The QGE generates two types of synthetic queries per policy: positive queries containing the policy-defined information, and negative queries on semantically similar topics without it. Positive queries are produced in two formats—information embedded in the prompt text, and information provided as an attached text—to capture different real-world usage patterns. Diversity is further increased by varying intrinsic parameters such as originating department, job level, and target response length. Among three policy-side input configurations (policy only, policy + description, description only), policy-only yields the highest QAE pass rate (64.4%; Appendix C). Full prompts are in Appendix J.

Component	Backbone	Input	Output	Role
Policy Integrator	Qwen3-Embedding-0.6B	Policy name and description	Similar policy candidates	Detects semantic overlap with existing policies before registration
Rubric Generator	Qwen3-32B	Policy information	Six rubrics per policy	Generates criteria used by QAE
Query Generation Engine	Qwen3-32B	Policy + intrinsic inputs (department, job level, response length)	Synthetic positive and negative queries	Generates policy-grounded synthetic queries
Quality Assurance Engine	Qwen3-32B	QGE-generated queries + six rubrics from Rubric Generator	Per-rubric 0/1 scores; filtered queries	Validates queries against rubrics and applies threshold filtering
Embedding Model	Qwen3-Embedding-0.6B	QAE-passed queries (concatenated with policy description)	Dense vectors	Maps queries into a shared retrieval space
Query Pool	FAISS IndexFlatIP + SQLite	Embeddings, natural-language queries, policy labels	Searchable query database	Stores validated queries; supports label-level updates without retraining
Prediction Module	Qwen3-Embedding-0.6B + FAISS	User query; Query Pool	Predicted policy category (top-5 majority vote)	Performs cosine-similarity retrieval and majority voting

Table 2: Summary of Q-CAT components.

Quality Assurance Engine (QAE) The QAE filters QGE outputs before they enter the Query Pool. For each query, the QAE is given the target policy, the query, and the six policy-specific rubrics, and returns a 0/1 score per rubric. The total score ranges from 0 to 6, and queries scoring below a fixed threshold are discarded. Our ablation (Section 8.1) shows that a threshold of 1 with generated rubrics yields the best filtering quality. The prompts used by the Rubric Generator and QAE are listed in Appendix J.

Embedding Model Passed queries are embedded with the model selected in Section 8.2, after concatenation with their originating policy descriptions as validated in the same ablation. The resulting vectors are stored in SQLite and indexed with FAISS (Douze et al., 2024) using IndexFlatIP for exact inner-product search.

Query Pool The Query Pool stores each entry as a natural-language query, its embedding, the corresponding policy label, and optional per-rubric QAE scores. Natural-language entries allow administrators to inspect and edit the pool directly, while embeddings support similarity search. Policy updates are applied by modifying labels in the Query Pool, without retraining. Real user queries appended during deployment are de-identified before storage by an administrator or automated module.

Prediction Module Given a user query, the Prediction Module computes its embedding under the same model and retrieves the top-5 nearest entries from the Query Pool via the FAISS index. The predicted policy category is determined by majority vote over the retrieved labels, from which the binary prohibited/allowed decision is derived. Top-5 was selected over alternative retrieval strategies based on Section 8.3.

6 Experiments

6.1 Baselines

We compare Q-CAT against two families of baselines that represent common approaches for policy-aware query filtering.

Prompt Engineering We provide governance policies (both prohibited and allowed categories with their descriptions) as in-context instructions to general-purpose LLMs, which then classify each user query. We evaluate gpt-5-mini, gpt-5.2, Qwen3-8B, and Qwen3-32B under this setting. All models are evaluated under both thinking and non-thinking modes, except for gpt-5-mini, which is evaluated only in thinking mode. For gpt-5.2, we configure the thinking mode with reasoning effort set to high. The full classifier prompt is provided in Appendix J.

Supervised Fine-Tuning We fine-tune Qwen3-8B on QAE-filtered queries from the Query Pool, using the same data that Q-CAT retrieves over at inference time. Training uses LoRA adapters (Hu et al., 2022) with AdamW (learning rate 2×10^{-4} , cosine scheduling, warmup ratio 0.05, weight decay 0.01) for 3 epochs. The model is trained on 24 policy categories (PR 8 + HA 8 + EA 8), corresponding to the policies an organization would realistically define at initial deployment based on the currently observed query distribution. To avoid disadvantaging this baseline with out-of-distribution categories, the static evaluation in Section 7.1 is restricted to the matching 660-query subset of the Core Set ($24 \times 15 + 300$ HE/EE). The remaining 6 categories are introduced later in the dynamic evaluation (Section 7.2) to simulate previously unseen policies emerging in real-world deployment.

6.2 Hardware

All experiments are conducted on RTX 6000 Ada GPUs. LLM decoding latency for the Qwen3 family is measured using SGLang on $2 \times$ RTX 6000 Ada GPUs, while Q-CAT runs on an AMD EPYC 7763 CPU with $1 \times$ RTX 6000 Ada GPU.

6.3 Settings

We generate 400 queries per policy, approximately 12k across 30 policies, to balance diversity and noise (Appendix B). The policy-only QGE input setting yields the highest QAE pass rate among the tested configurations (Appendix C) and is used throughout.

7 Results

7.1 Performance in a Static Environment

Table 3 shows that Q-CAT achieves broadly comparable performance (row 1) to strong prompt-based baselines such as GPT-5.2 (rows 3–5), Qwen3-8B (rows 6–7), and Qwen3-32B (rows 8–9), while requiring substantially fewer computational resources (Section 7.3).

We also report **CA** (classification accuracy) to assess how reliably a system routes *ground-truth prohibited* queries to the correct policy category. Accurate policy classification can improve interpretability by linking a flagged query to a specific policy category. Q-CAT achieves comparable performance to others (final column). Further analysis

Model	F1 (% , $\uparrow$)	R (% , $\uparrow$)	P (% , $\uparrow$)	CA (% , $\uparrow$)
Q-CAT (ours)	77.7	90.0	68.4	86.7
Q-CAT (ours) (+feedback)	98.3	96.7	100	93.3
Prompt Engineering Baselines				
gpt-5-mini	71.0	99.2	55.4	99.2
gpt-5.2 _{thinking}	76.5	99.2	62.3	99.2
gpt-5.2 _{non-thinking}	81.7	96.7	70.7	99.2
Qwen3-8B _{thinking}	81.7	78.3	85.5	64.2
Qwen3-8B _{non-thinking}	53.0	44.2	66.3	71.7
Qwen3-32B _{thinking}	80.5	87.5	74.5	76.7
Qwen3-32B _{non-thinking}	64.7	84.2	52.6	91.7
Fine-Tuning Baseline				
Qwen3-8B _{non-thinking}	93.4	88.3	99.1	85.0

Table 3: Performance comparison under static policies. The scores for the binary classification of “prohibited” vs. “allowed” queries are shown whereas CA denotes policy classification accuracy.

shows that GPTs trade lower miss rates for higher FP rates (Table 11 in Appendix F).

In this static environment, where all queries are in-domain for the fine-tuning baseline, fine-tuned Qwen3-8B outperforms Q-CAT by 15.7 percentage points in F1. However, real-world queries and governance policies evolve over time, and fine-tuning-based methods may suffer from distribution shift. We therefore evaluate the methods under a dynamic policy environment.

7.2 Performance in a Dynamic Environment

A key advantage of our approach is that administrators can correct misclassifications by directly updating the Query Pool with labeled log data, without model retraining. With misclassified examples incorporated as simulated human feedback, Q-CAT achieves a 98.3% F1 score (Table 3, row 2).

To further validate this property, we design a 24-round experiment that simulates a dynamic enterprise environment with four policy-evolution scenarios (Fig. 2). Policy updates are applied at scenario boundaries (marked by text labels on the top), and each subsequent 6-round block measures performance under the updated policy setting.

At each round, 200 queries are randomly sampled from the active benchmark pool of the current scenario. The Query Pool is initialized with 10,025 synthesized queries. At every round, misclassified queries are appended to the Query Pool.

Rounds 1–6: Initial Deployment Phase In these rounds, the inputs consist of in-domain queries that simulate the early deployment state. The baseline policy has 24 categories (8 prohibited, 16 allowed), and the test set is sampled from 660 queries from

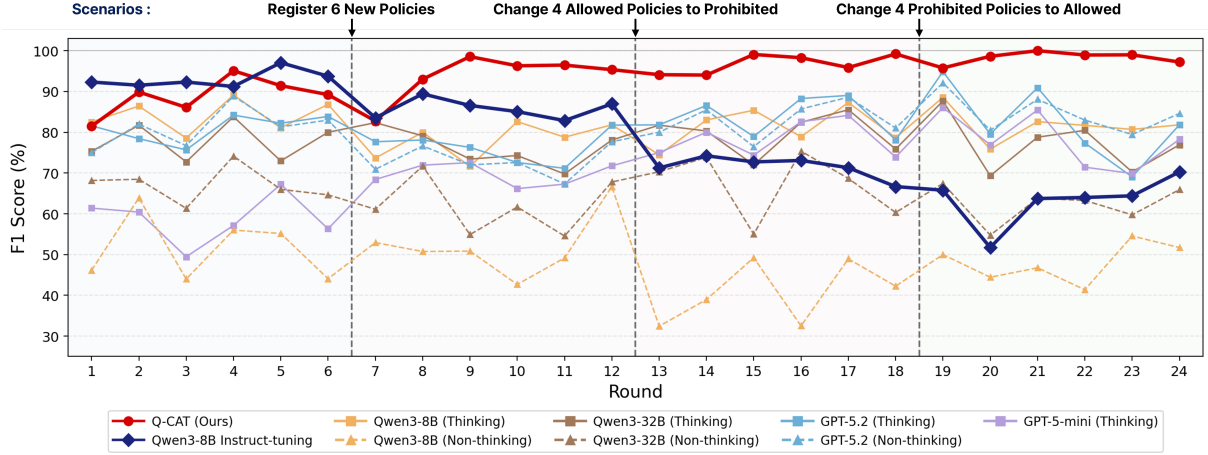


Figure 2: Comparison of F1 score trajectories under a dynamic environment.

the Core Set (360 information-bearing across 24 categories + 300 non-information/general queries). As expected, the fine-tuned approach achieves the best initial score, whereas Q-CAT gradually improves through error correction via Query Pool updates.

Rounds 7–12: Unseen Policy Addition Six previously unseen (holdout) categories are introduced: 2 prohibited³ and 4 allowed⁴ expanding the policy scope from 24 to 30 categories and the sampling pool from 660 to 750 queries. This explicitly challenges instruct-tuned models that never encountered these six categories during fine-tuning.

Rounds 13–18: Allowed → Prohibited Four previously allowed categories are relabeled as prohibited⁵. This simulates regulatory tightening and tests whether methods promptly suppress responses to previously permitted queries.

Rounds 19–24: Prohibited → Allowed Similar to previous rounds, here four categories are relabeled from prohibited to allowed.⁶

Analysis Fig. 2 shows that Q-CAT (red), although it initially shows lower performance due to weaker handling of general non-information queries, is the only evaluated method that shows a clear upward trend in our dynamic setting under time-varying queries and policy drift through

feedback-driven Query Pool updates. On the other hand, the fine-tuning approach (blue) is initially competitive but progressively degrades as policies drift beyond its static training distribution. LLM-based classifiers (GPT-5.2, Qwen3-32B, Qwen3-8B, GPT-5-mini) remain policy-agnostic and show high variance with no deployment-time adaptive capability, because they rely on in-context instructions without feedback accumulation. This interpretation is further supported by the error trajectories in Appendix G: Q-CAT maintains low false negative counts and false positive counts across most rounds, while GPT-family baselines incur higher false positive counts and the fine-tuning baseline exhibits sharp false negative spikes when policy labels flip between allowed and prohibited. We further verify that this adaptation pattern is not an artifact of shared-pool memorization by repeating the experiment on the Large Set with no query reuse across all 24 rounds (Appendix H). The same adaptation pattern also holds beyond the Korean enterprise setting: reproducing the dynamic-environment protocol on CLINC150 (English intent classification) and on a multilingual privacy-information benchmark built from WildChat yields qualitatively consistent results (Appendix I).

7.3 Inference Cost

We analyze the per-query inference latency of Q-CAT on the Core Set (750 queries) and compare it with LLM baselines under comparable hardware configurations.

Retrieval Pipeline Latency. Table 4 presents the component-wise latency breakdown of Q-CAT measured on an AMD EPYC 7763 CPU with $1 \times$

³employee disciplinary records and weekly team meeting agenda

⁴employee performance rating, original press release text, FAQ document, and business trip request procedure

⁵academic paper abstract, product price list, product specification sheet, and patent application details

⁶supply contract terms, new product launch schedule, product price list, and product specification sheet

RTX 6000 Ada GPU. Under a dynamic setting with a 12k Query Pool, the total end-to-end latency is 933 ms per query. The dominant cost arises from DB fetching (577 ms) and index construction (320 ms), while embedding (34 ms) and FAISS search (3 ms) contribute only marginal overhead.

Dynamic latency applies only at update points, when Query Pool changes are incorporated and the index is rebuilt. Outside update points, Q-CAT runs in the static setting, where the expected latency is 36 ms. Scheduled or batched updates can further decouple DB fetching and index construction from online serving, reducing user-facing latency.

The latency advantage should be interpreted with respect to both operating mode and performance. In the dynamic setting, Q-CAT is slower than Qwen3-8B_{non-thinking} in raw latency, but this baseline substantially underperforms. Our speed comparison is therefore primarily against reasoning-mode baselines, such as Qwen3-8B_{thinking}, which provide more comparable performance.

LLM Inference Latency. We further measure the decoding latency of Qwen3 models using SGLang on 2× RTX6000Ada GPUs. Without thinking, the models exhibit moderate latency (8B: 358 ms, 32B: 1,090 ms), while with thinking, the models show substantially higher computational cost (8B: 8,203 ms, 32B: 21,067 ms).

The results show that under the static condition, Q-CAT achieves 228× (compared to Qwen3-8B thinking) and 585× (compared to 32B-thinking) faster inference while achieving comparable performance (Table 3).

8 Ablation Study

8.1 Quality Assurance Engine

We evaluate the impact of QAE by varying rubric design and filtering thresholds. QAE filters synthetic queries before storing them in the Query Pool, ensuring consistency with the target information condition (presence or absence).

We compare three rubric settings: (1) generated rubric (Gen.), where QAE dynamically generates policy-specific rubrics; (2) default rubric (Def.), where QAE uses a hardcoded set of six predefined rubrics selected by the input information type; and (3) single rubric (Single), where one universal rubric is used instead of the six-rubric scheme.

As shown in Table 5, threshold 1 with the generated rubric achieves the best F1 scores for both

	Step	Average (ms)	Std Dev (ms)
Q-CAT	Embedding	34	3
	Fetching DB rows	577	46
	Index build	320	43
	FAISS search	3	1
	End-to-End (Dynamic)	933	80
	End-to-End w/ Static	36	3
Model		Average (ms)	Std Dev (ms)
Qwen3-8B _{non-thinking}		358	851
Qwen3-8B _{thinking}		8,203	6,943
Qwen3-32B _{non-thinking}		1,090	2,958
Qwen3-32B _{thinking}		21,067	16,555

Table 4: Average per-query latency on the Core Set. Top: component-wise latency breakdown of Q-CAT under a dynamic setting with a 12k Query Pool. Bottom: decoding latency of Qwen3 models. “End-to-End (Dynamic)” denotes the expected latency at update points where the Query Pool changes and the FAISS index is rebuilt, while “End-to-End w/ Static” denotes the latency when using a fixed Query Pool and a pre-built FAISS index. Further statistics on latency are provided in Appendix E.

Core and Large Sets (73.4% and 75.0%), outperforming the no-QAE baseline. Performance drops at higher thresholds due to over-filtering, while threshold 0 denotes no QAE filtering. The generated rubric also outperforms the default and single-rubric settings at threshold 1, demonstrating the value of policy-specific quality criteria.

Table 5: F1 scores across QAE thresholds for the Core and Large Sets.

Thr.	Core Set			Large Set		
	F1 Score (%)			F1 Score (%)		
	Gen.	Def.	Single	Gen.	Def.	Single
0	71.5	71.5	71.5	73.6	73.6	73.6
1	73.4	71.5	72.8	75.0	73.6	74.7
2	70.8	71.5	—	73.5	73.6	—
3	70.4	71.5	—	74.7	73.5	—
4	70.2	71.3	—	73.7	73.5	—
5	59.6	71.5	—	66.4	73.4	—
6	36.6	71.4	—	42.3	71.5	—

8.2 Embedding Models

We compare three embedding methods: query-only (embedding the generated query alone), contextual (embedding the query concatenated with its policy description) and both (both query-only and contextual cases are stored in Query Pool). The results show that the contextual embedding mostly matches or outperforms other strategies (Table 6, row 1, 4, col 2). The use of larger embedding models (4B, 8B) shows similar trend with slightly lower performance (row 1 vs 2,3)

Table 6: Ablation studies on embedding methods.

Set	Model Size	F1 Score (%)		
		Query	Contextual	Both
Core set	0.6B	76.6	76.6	77.1
	4B	74.9	76.7	74.0
	8B	72.6	73.0	71.0
Large set	0.6B	79.0	80.6	78.2
	4B	75.2	75.3	74.6
	8B	71.7	71.1	71.5

8.3 Prediction Module

We examine three retrieval strategies: top-1, top-5, and threshold-based retrieval. For top-5, the output is determined by the most frequent label among the five queries. Threshold values are determined via grid search over predefined cosine distance thresholds ranging from 0.50 to 0.94 (23 values in total).

The results show that top-5 consistently achieves the best performance (Table 7). Based on these results, we adopt top-5 retrieval with cosine distance as our default configuration.

Table 7: Prediction strategy comparison.

Model Size	Core Set			Large Set		
	F1 Score (%)			F1 Score (%)		
	Top1	Top5	Thr.	Top1	Top5	Thr.
0.6B	71.4	76.6	76.3	73.9	80.6	77.0
4B	71.2	76.7	73.0	71.5	75.3	73.2
8B	68.3	73.0	70.1	67.9	71.1	70.2

9 Conclusion

We presented Q-CAT, a non-parametric query filtering system that classifies user queries against dynamically defined governance policies without model training. By leveraging embedding-based nearest neighbor search over a synthetically generated Query Pool, Q-CAT achieves competitive performance with strong LLM baselines in static settings while demonstrating strong robustness and adaptability in our dynamic-policy simulation, where it adapts instantly through database updates and progressively improves via administrator feedback. Designed for on-premises deployment with modest inference-time computational requirements, Q-CAT offers a practical and scalable solution for enterprise environments where policies evolve, resources are constrained, and rapid deployment is essential.

Limitations

Our approach assumes that the embedding model maps queries containing similar types of information to nearby points in the vector space. If the types of information defined by an organization are highly specialized, domain-specific, or not intuitive enough for a general embedding model to capture their context, it may be necessary to adopt or train a more specialized embedding model. Indirectly or adversarially phrased queries can also be difficult for a general embedding model to capture; addressing this requires extending the embedding space to accommodate such expressions and complementing the current information-centric perspective with an intent-centric one.

We also assume that queries may contain a single type of prohibited or allowed information, contain undefined information, or contain no information related to policies, whereas real-world queries can include multiple types of information. The current prediction module uses top-5 majority voting, selected from a comparison of top-1, top-5, and threshold-based retrieval (Section 8.3), and the rubric configuration (six rubrics per policy with a filtering threshold of 1) was likewise determined empirically through ablation. Refining these components, for instance through distance-weighted aggregation or a more interpretable rubric-based scoring mechanism, remains as future work.

Finally, Q-CAT does not eliminate false negatives entirely. Review cost and false negative capture are in a trade-off, and residual false negatives—prohibited queries that pass the filter—remain, as is the case for most methods addressing this problem. In high-stakes deployments, Q-CAT may therefore need to be combined with complementary safeguards such as output-side filtering rather than used as a sole control.

Acknowledgments

This work was supported by an industry-academia research project with Samsung SDS.

Juheon Kang was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2022-NR068754).

Eunkyoung Choi, Hongseok Oh, and Wonseok Hwang were partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-23524855).

References

- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- Michelle Brachman, Amina El-Ashry, Casey Dugan, and Werner Geyer. 2024. [How knowledge workers use and want to use llms in an enterprise context](#). In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA. Association for Computing Machinery.
- Hwan Chang, Yumin Kim, Yonghyun Jun, and Hwanhee Lee. 2025. [Keep security! benchmarking security policy preservation in large language model contexts against indirect attacks in question answering](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6780–6800, Suzhou, China. Association for Computational Linguistics.
- Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. [The faiss library](#). *arXiv preprint arXiv:2401.08281*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773.
- Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. [Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms](#). *Preprint*, arXiv:2406.18495.
- Monte Hoover, Vatsal Baherwani, Neel Jain, Khalid Saifullah, Joseph Vincent, Chirag Jain, Melissa Kazemi Rad, C. Bayan Bruss, Ashwinee Panda, and Tom Goldstein. 2025. [Dynaguard: A dynamic guardian model with user-defined policies](#). *Preprint*, arXiv:2509.02563.
- Edward J Hu, Yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. [Llama guard: Llm-based input-output safeguard for human-ai conversations](#). *Preprint*, arXiv:2312.06674.
- Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, and 1 others. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316.
- Xiaoya Lu, Dongrui Liu, Yi Yu, Luxin Xu, and Jing Shao. 2025. [X-boundary: Establishing exact safety boundary to shield LLMs from jailbreak attacks without compromising usability](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5247–5272, Suzhou, China. Association for Computational Linguistics.
- Meta Llama Team. 2024. Meta llama guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.
- Oren Rachmil, Roy Betser, Itay Gershon, Omer Hofman, Nitay Yakoby, Yuval Meron, Idan Yankelev, Asaf Shabtai, Yuval Elovici, and Roman Vainshtein. 2026. [Training-free policy violation detection via activation-space whitening in LLMs](#). In *AAAI 2026 Workshop on Deployable AI (DAI)*.
- Nitin Ramrakhiani, Delton Myalil, Sachin Pawar, Manoj Apte, Rajan M A, Divyesh Saglani, and Imtiyazuddin Shaik. 2025. [QueryShield: A platform to mitigate enterprise data leakage in queries to external LLMs](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 358–369, Albuquerque, New Mexico. Association for Computational Linguistics.
- Xiaofei Wen, Wenjie Jacky Mo, Yanan Xie, Peng Qi, and Muhao Chen. 2025. [Towards policy-compliant agents: Learning efficient guardrails for policy violation detection](#). *Preprint*, arXiv:2510.03485.
- Guanlong Wu, Zheng Zhang, Yao Zhang, Weili Wang, Jianyu Niu, Ye Wu, and Yinqian Zhang. 2025. I know what you asked: Prompt leakage via kv-cache sharing in multi-tenant llm serving. In *NDSS*.
- Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, and 1 others. 2025. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*.
- Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. [Wildchat: 1m chatgpt interaction logs in the wild](#). *Preprint*, arXiv:2405.01470.

A Formal Specification

Q-CAT is a modular pipeline in which each component admits multiple implementations depending on the deployment context (e.g., the embedding model can be replaced without altering the framework). We therefore formalize each component as an interface, specified by its input/output signature, and describe our implementation as one instance.

Embedding model Let $\mathcal{Q}$ denote the space of natural-language queries. An embedding model $\phi : \mathcal{Q} \rightarrow \mathbb{R}^d$ maps each query to a dense vector, ℓ_2 -normalized so that cosine similarity is computed as the inner product $\langle \phi(q), \phi(q') \rangle$.

Query Pool Let $\mathcal{Y}$ denote the set of registered policy labels. The Query Pool is a labeled set

$$P = \{(q_i, v_i, y_i)\}_{i=1}^N, \quad v_i = \phi(q_i), \quad y_i \in \mathcal{Y},$$

where N is the number of stored entries and the natural-language query q_i is retained for administrator inspection and editing.

Policy-label mapping A mapping $g : \mathcal{Y} \rightarrow \{\text{PROHIBITED}, \text{ALLOWED}\}$ converts a policy label to the binary decision.

Algorithm 1 Inference

Input: user query q ; Query Pool P ; mapping g ; $k = 5$

Output: binary decision; predicted policy label

- 1: $v \leftarrow \phi(q)$
 - 2: $R_k \leftarrow$ the top- k entries of P by inner product $\langle v, v_i \rangle$
 - 3: $\hat{y} \leftarrow \arg \max_{y \in \mathcal{Y}} \sum_{(q_i, v_i, y_i) \in R_k} \mathbb{1}[y_i = y]$
 - 4: **return** $g(\hat{y})$, $\hat{y}$
-

Algorithm 2 Pool update

Input: Query Pool P ; administrator feedback

- 1: **if** insertion of a reviewed query (q, y^*) : $P \leftarrow P \cup \{(q, \phi(q), y^*)\}$
 - 2: **if** deletion of entry i : $P \leftarrow P \setminus \{(q_i, v_i, y_i)\}$
 - 3: **if** relabeling of entry i : $y_i \leftarrow y'_i$
 - 4: **if** policy change (e.g., ALLOWED to PROHIBITED): update g or relabel the affected entries
 - 5: rebuild the retrieval index over $\{v_i\}$; ϕ remains unchanged
-

B Selecting Query Pool Size

We evaluate the effect of the Query Pool size without applying QAE filtering. We construct three pool sizes by generating 40, 400, and 4k queries per policy across 30 policies, resulting in pools of 1.2k, 12k, and 120k queries, respectively, and measure their F1 scores (Table 8). The 12k pool achieves the best F1 score (65.3%), outperforming both the smaller 1.2k pool (59.9%) and the larger 120k pool (62.4%).

Pool Size	F1 Score (%)
120k	62.4
12k	65.3
1.2k	59.9

Table 8: F1 scores across different Query Pool sizes using policy-only configuration without QAE. Best performance at 12k.

C QGE Input Configuration Analysis

We analyze QAE pass rates under three QGE input configurations: policy only, policy + description, and description only. As shown in Table 9, the policy-only configuration yields the highest pass rate (64.4%), with 7,417 queries kept (stored in the Query Pool) and 4,103 discarded. In contrast, policy + description and description-only achieve lower pass rates of 56.2% and 55.9%, respectively. These results indicate that policy-only input produces queries that are better aligned with the QAE criteria.

Configuration	Pass Rate (%)	Kept	Discarded
Policy only	64.4	7,417	4,103
Policy + Description	56.2	6,695	5,216
Description only	55.9	6,659	5,266

Table 9: QAE pass rates across different QGE input configurations, with counts of queries stored in the Query Pool (Kept) and filtered out (Discarded).

D MMD Analysis: Benchmark vs Query Pool

To quantitatively verify that the benchmark and the QGE-generated Query Pool occupy distinct regions of the embedding space, we perform a kernel two-sample test based on Maximum Mean Discrepancy (MMD) on 1024-dimensional query embeddings. MMD² values near zero indicate distributional similarity, while larger values indicate separation.

Since absolute MMD² values are difficult to interpret in isolation, we provide two reference

points: (i) a within-benchmark baseline (Core vs Large, both constructed by the same procedure), and (ii) MS MARCO queries vs passages (Bajaj et al., 2016), where the two sides are by definition different distributions.

The Query Pool and the Core Set are by design constrained to the same domain (within-category employee queries), yet they exhibit a distributional separation comparable to MS MARCO’s queries-vs-passages, where the two sides are entirely different text types. This provides quantitative evidence that the evaluation is not self-referential.

Figure 3 visualizes this separation by projecting the Query Pool (3,000-query subsample) and the full Core Set (750 queries) onto a 2D plane via t-SNE with cosine metric and perplexity 50. The two sets form visibly distinct clusters across the embedding space, with the Core Set concentrated in tight regions that the Query Pool only sparsely covers. This qualitative pattern is consistent with the MMD-based separation reported above.

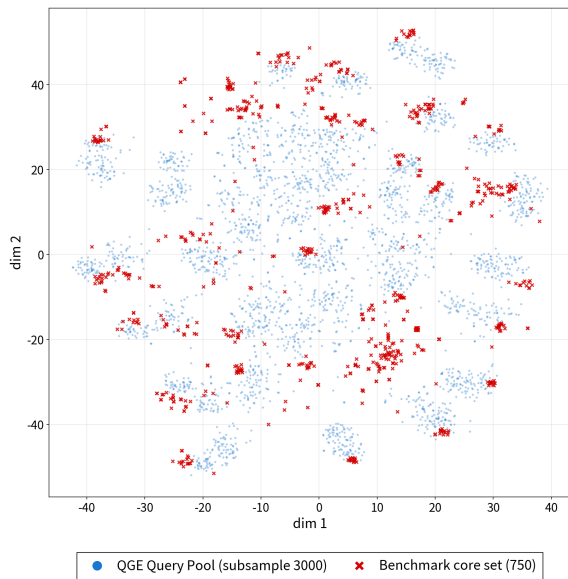


Figure 3: t-SNE projection of Query Pool (3,000 subsample) and Core Set (750 queries) embeddings.

E LLM Latency Statistics

We provide additional statistics regarding the large standard deviation of LLM latency (Table 4). As shown in Figure 4, LLM latency exhibits a heavier tail, especially in the thinking setting. This indicates that the Q-CAT system offers more predictable latency, which can be a practical advantage for query filtering in business environments.

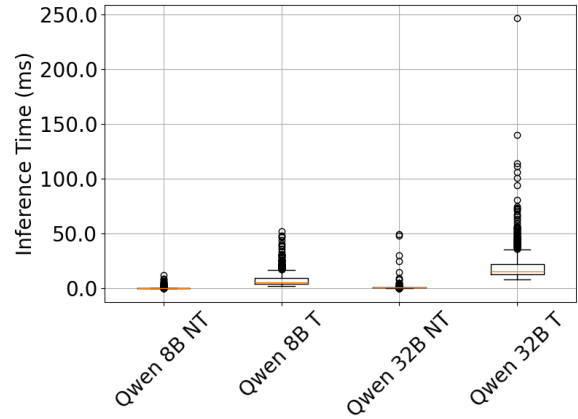


Figure 4: Inference time distribution of Qwen 8B and 32B in non-thinking (NT) and thinking (T) settings. Compared with NT, the thinking setting exhibits higher latency and a heavier long-tail distribution, particularly for Qwen 32B.

F Error-Type Analysis

We further examine three statistics: accuracy $(TP + TN)/ALL$, FN rate FN/ALL , and FP rate FP/ALL . Among them, the FN rate is most critical as FNs correspond to prohibited queries that are incorrectly allowed.

The result shows that GPTs achieve the lowest FN rate (Table 11, row 3–5, col 2), but this comes with substantially higher FP rate (row 3–5, col 3) indicating they tend to classify ambiguous queries more conservatively. By contrast, Q-CAT with feedback maintains a low FN rate while removing FP events entirely, resulting in the highest accuracy and the strongest overall error profile among the compared methods.

We further analyze the four remaining FN cases of Q-CAT (+feedback) (Table 12). The residual errors mainly arise in three situations. First, some cases are misclassified due to their surface-form using terms often appear in other related policies. For example, although Case 1 is labeled as *Business outcome summary* (row 1), the query does not explicitly mention routine business reporting; instead, it asks for a summary of a four-week pilot with expressions such as “4-week results,” “ROI,” and “whether to expand to the next stage,” making it closely resemble a *Project completion report*.

Second, domain-specific technical terminology can bias retrieval toward semantically adjacent but different document types. In Cases 2 and 3 (row 2, 3), the queries are labeled as *R&D progress notes*, but they contain many specialized terms and tech-

Pair	MMD ²	Ratio vs baseline
Core Set vs Large Set (within-benchmark baseline)	0.00125	1×
Query Pool vs Core Set	0.05807	46.5×
MS MARCO queries vs passages	0.08168	65.4×

Table 10: MMD² values across distribution pairs. The Query-vs-Core Set distance is on the same scale as MS MARCO’s queries-vs-passages distance and 46× larger than the within-benchmark baseline.

Model	Accuracy (% , ↑)	FN Rate (% , ↓)	FP Rate (% , ↓)
Q-CAT (ours)	90.6	1.8	7.6
Q-CAT (ours) (+feedback)	99.4	0.6	0.0
Prompt Engineering Baselines			
gpt-5-mini	85.3	0.2	14.6
gpt-5.2 _{thinking}	88.9	0.2	10.9
gpt-5.2 _{non-thinking}	92.1	0.6	7.3
Qwen3-8B _{thinking}	93.6	3.9	2.4
Qwen3-8B _{non-thinking}	85.8	10.2	4.1
Qwen3-32B _{thinking}	92.3	2.3	5.5
Qwen3-32B _{non-thinking}	83.3	2.9	13.8
Fine-Tuning Baseline			
Qwen3-8B _{instruct-tuning}	97.7	2.1	0.2

Table 11: Error-type statistics under static policies on 660 examples from the Core Set.

nical result summaries, such as “AFib detection model,” “IRB/quality-control risk log,” and “walk-forward validation,” which makes them resemble *Academic paper abstract* examples.

Third, some errors arise from misleading contextual cues in the query. In Case 4 (row 4), the ground-truth label is *Asset inventory ledger*, but the query states that the ledger was “edited by multiple people” and now contains “duplicates and missing values,” which appears to have pulled retrieval toward *Employee disciplinary records*.

Overall, these findings suggest that the remaining errors are concentrated in cases where the query strongly resembles another plausible type at the lexical or discourse level. Importantly, once identified by an administrator, such cases can be directly added to the Query Pool and resolved without re-training.

#	Prohibited Information	Max Sim.	kNN Vote
1	Business outcome summary	0.61	1/5
2	R&D progress notes	0.55	2/5
3	R&D progress notes	0.54	2/5
4	Asset inventory ledger	0.71	2/5

Table 12: Remaining false negative cases of Q-CAT (+feedback).

G False Negative and False Positive Trajectories in Dynamic Environments

Figure 5, 6, and 7 respectively show the accuracy, FN and FP trajectories in the dynamic environment. Q-CAT maintains low FN counts across most rounds and remains stable after policy transitions (red lines), indicating effective adaptation through the Query Pool updates. By contrast, the fine-tuning baseline exhibits sharp FN spikes when a policy changes from sensitive to safe or vice versa (dark blue lines), reflecting the brittleness of a fixed learned boundary under policy drift. Q-CAT is more robust in this setting because policy changes are directly propagated to the labels in the Query Pool, allowing the system to adapt without retraining. GPTs, meanwhile, show higher FP counts across many rounds, suggesting a more conservative filtering tendency.

H Evaluation on the Large Set without Query Reuse

In the dynamic evaluation of Section 7.2, misclassified queries are appended to the Query Pool while evaluation queries are sampled from the same Core Set, which could allow part of the reported adaptivity to reflect retrieval memorization rather than genuine adaptation. To rule this out, we repeat the same era policy schedule on the Large Set (7,482 queries), sampling 300 fresh queries per round under a strict no-replacement rule across all 24 rounds. This guarantees that every evaluation query is unseen in any subsequent round. Q-CAT and the fine-tuned Qwen3-8B baseline are evaluated on the identical query sequence at every round.

Era (rounds)	Q-CAT	Qwen3-8B FT
0: Initial deployment (1–6)	85.99	94.63
1: New policy added (7–12)	90.43	85.18
2: Allowed→Prohibited (13–18)	93.98	69.86
3: Prohibited→Allowed (19–24)	94.99	65.01

Table 13: Era-level mean F1 (%) on the Large Set without query reuse. Each era spans 6 rounds.

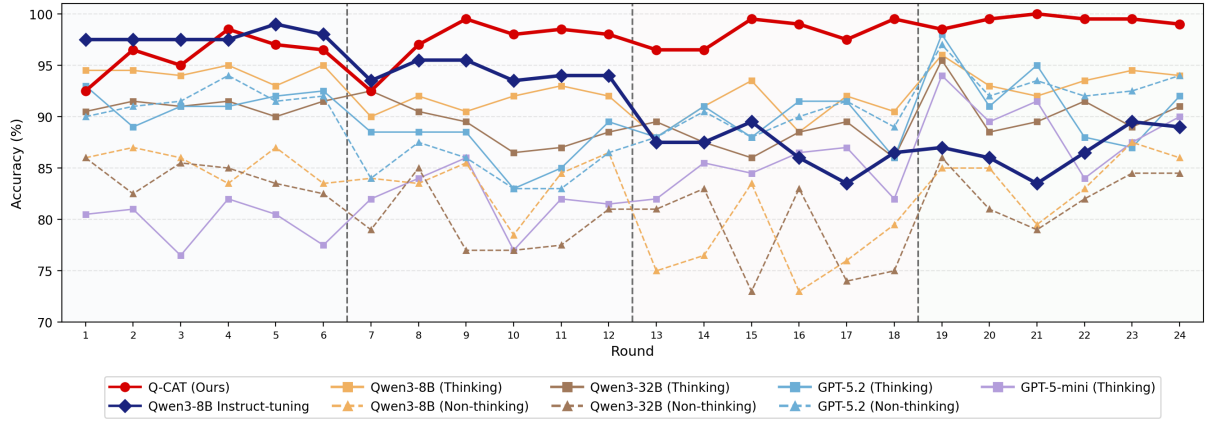


Figure 5: Round-wise accuracy in the dynamic environment. Higher is better. Q-CAT maintains strong accuracy across rounds and remains comparatively stable after policy updates.

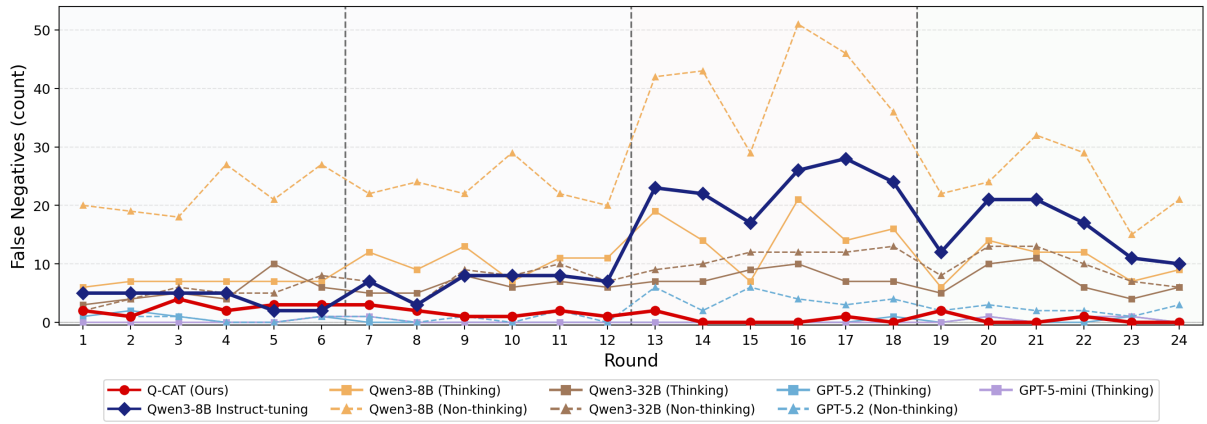


Figure 6: Round-wise false negative counts in the dynamic environment. Lower is better. Q-CAT generally maintains low false negative counts across rounds, especially after policy updates.

Q-CAT’s adaptation pattern persists even when no evaluation query is ever reused: era-level mean F1 rises from 85.99 (Era 0) to 94.99 (Era 3), confirming that round-to-round gains reflect genuine adaptation rather than memorization of a shared pool. The fine-tuned baseline degrades from 94.63 to 65.01 under policy drift, reproducing the qualitative pattern of Figure 2. By the final era, Q-CAT outperforms the fine-tuned baseline by +29.98 F1. Per-round F1 trajectories are shown in Figure 8.

I Generalization across Languages and Domains

The main experiments evaluate Q-CAT in a Korean enterprise setting. To assess whether the same mechanism transfers to other languages and domains, we report two additional experiments: an English intent-classification benchmark and a multilingual privacy-information benchmark constructed from WildChat.

I.1 Cross-Domain Evaluation on CLINC150

We evaluate Q-CAT on CLINC150 (Larson et al., 2019), an English public intent-classification benchmark whose domain differs substantially from the Korean enterprise-policy setting of the main paper. We randomly select 30 of the 150 CLINC150 intent classes and treat them as policy categories, then run Q-CAT under the same dynamic-environment protocol as Section 7.2: four eras of 6 rounds each, with a policy-change scenario at every era boundary.

Era	Avg F1
0: Initial deployment	0.842
1: Policy change	0.945
2: Policy change	0.962
3: Policy change	0.971

Table 14: Q-CAT era-level F1 on CLINC150 under the dynamic-environment protocol.

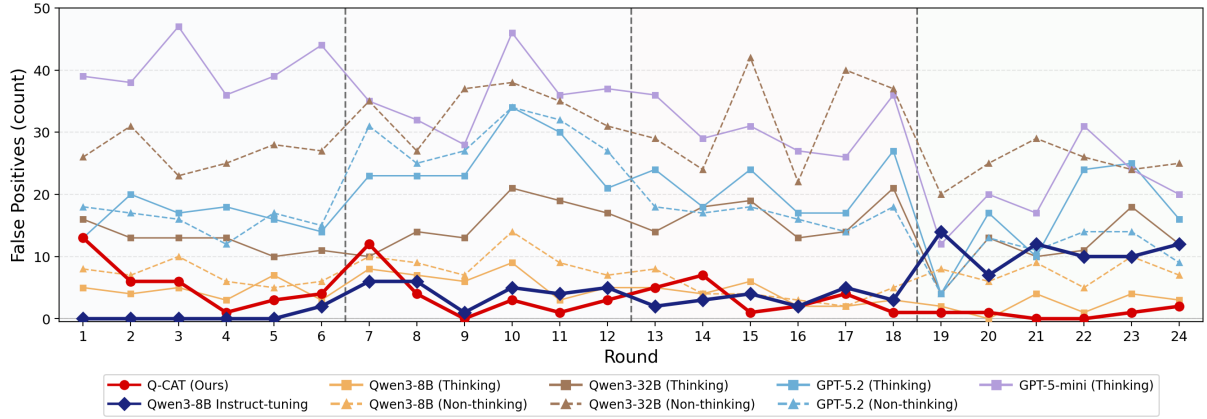


Figure 7: Round-wise false positive counts in the dynamic environment. Lower is better. GPT-family baselines show higher false positive counts than Q-CAT across many rounds, suggesting a more conservative filtering tendency.

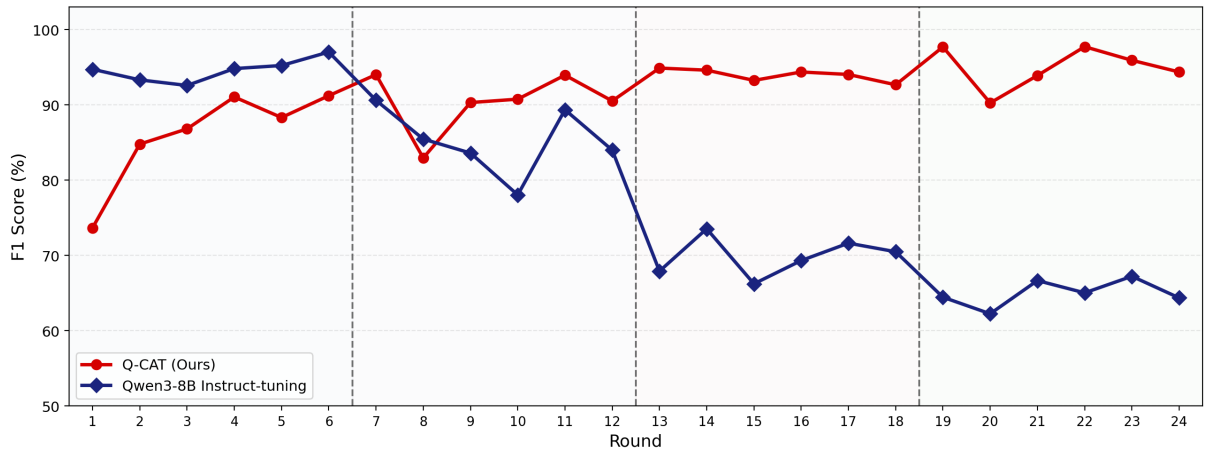


Figure 8: Per-round F1 trajectory on the Large Set without query reuse. Vertical dashed lines mark era boundaries.

Q-CAT shows monotonically improving F1 across eras on CLINC150, mirroring the qualitative pattern observed in the main Korean enterprise experiments. This indicates that the core mechanism operates consistently across languages and domains, and that extending it to a new environment requires only redefining the policy set rather than model retraining.

I.2 Multilingual Evaluation on WildChat

To further test whether the feedback-driven adaptation pattern holds across multiple natural languages, we construct a multilingual privacy-information benchmark from WildChat-1M (Zhao et al., 2024). We sample 93,461 queries across six languages (English, Chinese, Russian, French, Spanish, Arabic), label each query for the kind of personal or sensitive information it contains using gpt-5-mini, and canonicalize the raw labels into a 43-category taxonomy. For each language, we select the top 8 prohibited information

categories (e.g., person_name, medical_data, corporate_confidential) and 8 allowed information topics (e.g., technical_knowledge, creative_content) to form a per-language policy set. Q-CAT is then evaluated separately on each language under the same dynamic-environment protocol as Section 7.2.

Across all six languages, the mean accuracy rises from approximately 0.51 in the initial rounds to around 0.62 by the final era, showing a consistent upward trend as the Query Pool incorporates feedback from misclassified queries. The absolute gains are smaller than those observed on CLINC150 or in the main Korean enterprise benchmark. This is in part because WildChat consists of open-ended conversational queries that are not designed for clean information-type classification, and the information signal in many queries is implicit or mixed. Despite this difficulty, the adaptation pattern remains visible across every language, indicating that

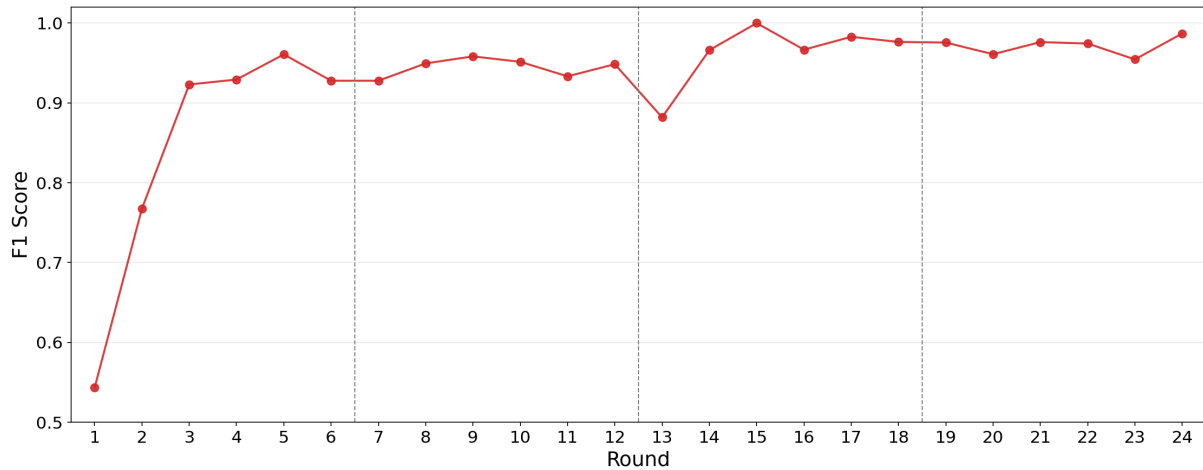


Figure 9: Per-round F1 trajectory of Q-CAT on CLINC150. Vertical dashed lines mark era boundaries.

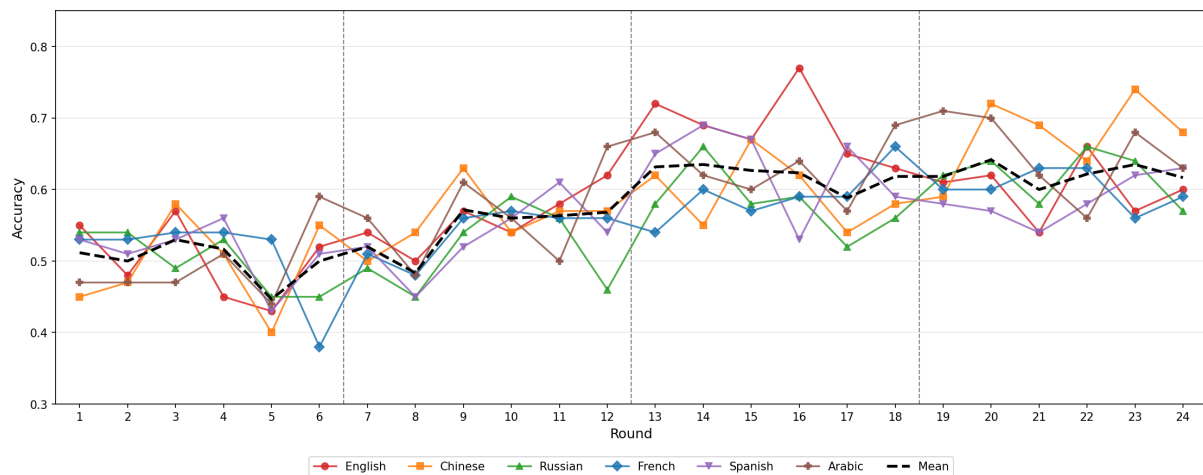


Figure 10: Per-round accuracy of Q-CAT across six languages on the WildChat-based privacy benchmark. Vertical dashed lines mark era boundaries; the black dashed line shows the cross-language mean.

the feedback-driven Query Pool update reflects the underlying input distribution even in noisy multi-lingual settings.

J Prompting Details

Query Generation Engine Prompt (Policy-Aligned, File Attached)

System Message

You are a chatbot that kindly answers user questions.

Requirements:

Given *Information*, *Responsible Department*, *Request Format*, *Requester Position*, and *Response Length*:

1. Write five example user queries that include direct and detailed contents of the information.
2. Each example query must explicitly specify the attached file contents and be phrased like a realistic user request.
3. The attachment (e.g., report, datasheet, manual)

must be described in detail so that it contains the information.

4. Finally, provide the example user queries as a single list inside a JSON code block.
5. The output must include exactly one JSON code block, and within that block there must be only one list.
6. Do not include any explanations, blank lines, or extra sentences other than the required format. Output only the code block.
7. The list must contain exactly five strings, and no additional text is allowed.
8. The length of each string must follow the response length requirement.

Example Input: one-shot example

Write queries that satisfy the requirements for the following policy:

- Information: {policy}
- Responsible Department: {department}
- Request Format: {form}
- Requester Position: {position}
- Response Length: {length}

Output Format :

```
```json
["...", "...", "...", "...", "..."]
```

### Query Generation Engine Prompt (Policy-Aligned, No File)

#### System Message

You are a chatbot that kindly answers user questions.

#### Requirements:

Given *Information*, *Responsible Department*, *Request Format*, *Requester Position*, and *Response Length*:

1. Write five example user queries that include direct and detailed contents of the information.
2. Queries must be in the form of direct requests without any attached files.
3. Finally, provide the example user queries as a single list inside a JSON code block.
4. The output must include exactly one JSON code block, and within that block there must be only one list.
5. Do not include any explanations, blank lines, or extra sentences other than the required format. Output only the code block.
6. The list must contain exactly five strings, and no additional text is allowed.
7. The length of each string must follow the response length requirement.

**Example Input:** one-shot example

**Write queries that satisfy the requirements for the following policy:**

- Information: {policy}
- Responsible Department: {department}
- Request Format: {form}
- Requester Position: {position}
- Response Length: {length}

#### Output Format :

```
```json
["...", "...", "...", "...", "..."]
```

Query Generation Engine Prompt (Policy-Conflict, No File)

System Message

You are a chatbot that kindly answers user questions.

Requirements:

Given *Information*, *Responsible Department*, *Request Format*, *Requester Position*, and *Response Length*:

1. Write five example prompts that are similar in topic to the given information but do not contain the information itself, and are phrased as if a user would send them to an LLM.
2. Each query must be a realistic, direct request with no attached files.
3. Finally, provide the example user queries as a single list inside a JSON code block.
4. The output must include exactly one JSON code block, and within that block there must be only one list.
5. Do not include any explanations, blank lines, or extra sentences other than the required format. Output only the code block.
6. The list must contain exactly five strings, and no additional text is allowed.
7. The length of each string must follow the response length requirement.

Example Input: one-shot example

Write queries that satisfy the requirements for the following policy:

- Information: {policy}
- Responsible Department: {department}
- Request Format: {form}
- Requester Position: {position}
- Response Length: {length}

Output Format:

```
```json
["...", "...", "...", "...", "..."]
```

### Rubric Generator

#### System Message

You are an expert evaluator of LLM usage information.

#### Prompt

Please generate evaluation criteria (rubrics) to determine whether a user's LLM prompt contains the given information.

#### Requirements:

1. Generate six evaluation items that can determine whether the user query contains the given information, following the format below.

2. Each rubric must be written as an interrogative sentence (e.g., "Does ...?", "Is ...?").
3. Each rubric must evaluate whether the user query contains the target information, and satisfying the rubric should imply that the information is included.
4. All items must be mutually non-overlapping and address distinct aspects.
5. Provide the final evaluation items as strings within a single list inside a JSON code block.

#### Example Input:

- Information: Email address

#### Example Output:

```
["Does the user input contain a typical email address
format combining an '@' symbol and a domain?",
"Does it include an email address that appears
realistically receivable by combining an ID and
domain?",
"Is personally identifiable email account information
exposed in the content?",
"Does the prompt specify an official corporate or
customer support email address?",
"Has the user directly provided an actual email
address as input?",
"Does the prompt include a list of multiple email
addresses and request processing or analysis?"]
```

Write rubrics that satisfy the above requirements for the following information:

#### [Information]

policy

#### Output Format (must strictly follow):

```
["...", "...", "...", "...", "...", "..."]
```

The output must strictly follow the format below. Do not include explanations or natural language outside the JSON block.

```
{
 "score": "0|1",
 "reason": "Provide a detailed
evaluation rationale for rubric 1."
}
:
{
 "score": "0|1",
 "reason": "Provide a detailed
evaluation rationale for rubric 6."
}
```

### Instruction Prompt (Classifier)

#### Prompt

You are a strict classifier that analyzes user queries. Given a user's request, determine whether the request contains either the prohibited content or the allowed content defined below.

#### <Start of Prohibited Content Categories>

```
{prohibited_info}
```

#### <End of Prohibited Content Categories>

#### <Start of Allowed Content Categories>

```
{allowed_info}
```

#### <End of Allowed Content Categories>

#### <Start of Conversation>

```
User: {prompt}
```

#### <End of Conversation>

#### Output Format:

- The first line must be exactly one of: Prohibited Content, Allowed Content, or Not Included.
- If the first line is Prohibited Content or Allowed Content, write the category name on the second line.

### Quality Assurance Engine Prompt

#### System Message

You are a chatbot that kindly answers user questions.

#### Prompt

You are assigned the role of an AI Policy Evaluator. Based on the Information and the User Request (query) provided below, evaluate whether the user request contains the information according to the given evaluation criteria (rubric).

#### [Evaluation Criteria (Rubric)]

```
{rubrics}
```

#### [Information]

```
- {policy}
```

#### [User Request (query)]

```
{query}
```

Scoring rule:

- Assign 1 if the rubric is satisfied.
- Assign 0 if the rubric is not satisfied.

#### Output Format

## K Examples of Generated Rubrics

The Rubric Generator generates six rubrics per registered policy, and the QAE uses them to verify that a QGE-generated query actually contains the information defined by its target policy (Section 5). We show the examples of generated rubrics of the two policies *Academic paper abstract* and *R&D progress notes*. The two policies are topically close, but each yields rubrics grounded in its own definition of the target information, making the difference between the two rubric sets visible. The rubric generator's prompting detail is given in Appendix J.

## 논문 초록 (Academic paper abstract)

### Korean:

1. 입력에 연구 문제에 대한 명확한 진술이 포함되어 있는가?
2. 사용된 연구 방법이나 분석 절차를 구체적으로 기술하고 있는가?
3. 연구 결과를 수치, 데이터 또는 관찰 결과의 형태로 제시하고 있는가?
4. 연구의 학술적 또는 실용적 의미를 설명하는 진술이 포함되어 있는가?
5. 문제 제기, 연구 방법, 결과, 의의가 논리적으로 연결된 흐름을 이루고 있는가?
6. 초록이 연구 전반의 핵심 요소를 간결하게 요약하고 있는가?

### Translated:

1. Does the input contain a clear statement of the research problem?
2. Does it concretely describe the research method or analysis procedure used?
3. Does it express research results in terms of figures, data, or observations?
4. Does it contain a statement explaining the academic or practical significance of the research?
5. Do the problem statement, method, results, and significance form a logically connected flow?
6. Does the abstract concisely summarize the core elements of the overall research?

4. Does it mention the definition of evaluation metrics or analysis methods?
5. Does it include analysis results or data interpretation from previous research stages?
6. Does it specify concrete plans or goals for the next research stage?

## L Example of Core Set

### PR

#### Korean:

```
{
 "information": "공급 계약 조건",
 "query": "거래처 대신테크 신규 공급 계약 검토 메모 작성 중인데, 아래 계약 조건을 바탕으로 리스크 관점에서 짚어볼 포인트를 정리해줘. 단가 2,450원/SET, 계약 기간 2년, 환율 10% 이상 변동 시 단가 재조정, 클레임 발생률 월 출고량의 0.7% 초과 시 단가 5% 인하, 납기 D+5, 월 최소 발주 수량 10,000SET 미충족 시 위약금 월 300만원",
 "sensitivity": true,
 "type": "PR",
 "information_description": "단가, 계약 기간, 납기, 발주/출고 물량 등의 공급 계약 조건을 담고 있습니다.",
 "related_information": null
}
```

#### Translated:

```
{
 "information": "Supply contract terms",
 "query": "I am preparing a review memo for a new supply contract with Daesin Tech. Based on the contract terms below, please summarize the key points to examine from a risk perspective. Unit price: KRW 2,450 per set; contract term: 2 years; unit price readjustment if the exchange rate fluctuates by 10% or more; 5% unit price reduction if the monthly claim rate exceeds 0.7% of monthly shipment volume; delivery term: D+5; monthly penalty of KRW 3,000,000 if the minimum monthly order quantity of 10,000 sets is not met.",
 "sensitivity": true,
 "type": "PR",
 "information_description": "It includes supply contract terms such as unit price, contract duration, delivery schedule, and order/shipment volumes.",
 "related_information": null
}
```

## 연구 진행 노트 (R&D progress notes)

### Korean:

1. 입력에 특정 날짜나 연구 기간을 명시한 일자별 기록이 포함되어 있는가?
2. 연구 가설의 내용이나 가설 검증에 대한 논의가 포함되어 있는가?
3. 변수 설정, 방법론, 절차 등 실험 설계에 관한 구체적인 정보를 제시하고 있는가?
4. 평가 지표의 정의나 분석 방법에 대해 언급하고 있는가?
5. 이전 연구 단계의 분석 결과나 데이터 해석이 포함되어 있는가?
6. 다음 연구 단계에 대한 구체적인 계획이나 목표를 명시하고 있는가?

### Translated:

1. Does the input contain dated records specifying particular dates or research periods?
2. Does it include the content of research hypotheses or discussion of hypothesis testing?
3. Does it present concrete information on experimental design, such as variable settings, methodology, or procedures?

## Korean:

```
{
 "information": "논문 초록",
 "query": "아래에 붙인 논문 초록들 보고
각 초록을 (문제-방법-결과-의의) 4줄
템플릿으로 정리해줘.
```

## [초록1]

본 연구는 대규모 사내 지식 자산의 활용도를 제고하기 위해 한국어 임베딩 기반 RAG(Retrieval-Augmented Generation) 파이프라인을 설계하고 이를 실증적으로 평가하였다.

문서 전처리 단계에서는 표와 그림 캡션을 구조화하여 비정형 정보를 검색 친화적 형태로 변환하고, 의미 단위 기반 청크 분할 전략을 적용하였다.

질의-문서 매칭 정확도를 높이기 위해 bi-encoder를 이용한 1차 후보 검색과 cross-encoder를 활용한 재순위화 단계를 결합한 다단계 검색 구조를 제안한다. 해당 파이프라인은 내부 위키 약 12만 건 규모의 실제 기업 데이터에 적용되어 운영 환경에서 검증되었다.

평가 지표로는 nDCG@10과 MRR을 사용하였으며, 제안 방법은 기존 임베딩 검색 대비 nDCG@10을 0.46에서 0.58로 유의미하게 향상시켰다.

또한 헬프데스크 질의 응답 시나리오에서 평균 응답 시간이 18% 단축되어 실무적 효용성이 확인되었다.

정성 평가 결과, 사용자는 보다 관련성 높은 문서를 상위에 제시받아 탐색 부담이 감소하였다고 보고하였다.

본 연구는 한국어 기업 문서 환경에서 RAG 기반 검색-생성 통합 구조의 실질적 효과를 입증한 사례로서, 향후 지식 관리 시스템 고도화에 기여할 수 있다.

## [초록2]

본 논문은 단기 전력 수요 예측의 정확도와 해석가능성을 동시에 개선하기 위하여

Temporal Fusion Transformer(TFT)

기반의 멀티모달 입력 구조를 제안한다.

제안 모델은 공휴일 정보, 기온 시계열, 요금제 구간과 같은 외생 변수를 전력 수요 데이터와 통합하여 시계열 의존성과 맥락 정보를 함께 학습한다.

2019년부터 2023년까지의 시간 단위 전국 전력 수요 데이터를 활용하여 모델을 학습하고, LSTM 및 기존 통계적 방법과의 비교 실험을 수행하였다.

그 결과, 제안한 TFT 모델은 기존 LSTM 대비 평균절대백분율오차(MAPE)를 약 1.9%p 감소시키며 일관된 성능 향상을 보였다.

특히 피크 부하 시간대와 같이 변동성이 큰 구간에서도 예측 안정성이 유의하게 개선되었다.

또한 TFT의 attention 메커니즘을 분석한 결과, 여름·겨울 피크 구간에서 기온 변수가 수요 예측에 지배적인 영향을 미치는 것을 확인하였다.

이는 단순 예측 성능을 넘어, 모델이 실제 전력 수요 결정 요인을 합리적으로 반영하고 있음을 시사한다.

본 연구는 전력 계통 운영 및 수요반응 전략 수립을 위한 신뢰도 높은 예측 및 해석 가능한 인공지능 모델의 가능성을 제시한다.",

"sensitivity": false,

"information\_description": "연구 문제에 대해 문제제기, 연구방법, 결과, 의의를 담은 논문 초록 정보가 포함되어 있습니다.",

"type": "HA",

"related\_information": "연구 진행 노트"

}

## Translated:

{

"information": "Academic paper abstract",

"query": "Please review the abstracts attached below and summarize each abstract using a four-line template: (Problem - Method - Results - Significance)."

## [Abstract 1]

This study designed and empirically evaluated a Korean embedding-based RAG (Retrieval-Augmented Generation) pipeline to enhance the utilization of large-scale internal knowledge assets. In the document preprocessing stage, tables and figure captions were structured to convert unstructured information into a search-friendly format, and a semantic-unit-based chunking strategy was applied. To improve query-document matching accuracy, a multi-stage retrieval architecture combining a bi-encoder for initial candidate retrieval and a cross-encoder for re-ranking was proposed.

The pipeline was deployed and validated in an operational environment using approximately 120,000 internal wiki documents from a real enterprise dataset.

Evaluation metrics included nDCG@10 and MRR, and the proposed method significantly improved nDCG@10 from 0.46 to 0.58 compared to baseline embedding search.

In addition, the average response time in a helpdesk question-answering scenario was reduced by 18%, demonstrating practical utility. Qualitative evaluation indicated that users were presented with more relevant documents at the top of the results list, reducing their search burden.

This study provides empirical evidence of the practical effectiveness of an integrated RAG-based retrieval-generation architecture in a Korean enterprise document environment and contributes to the advancement of knowledge management systems.

[Abstract 2]

This paper proposes a multimodal input architecture based on the Temporal Fusion Transformer (TFT) to simultaneously improve the accuracy and interpretability of short-term electricity demand forecasting.

The proposed model integrates exogenous variables such as holiday information, temperature time series, and tariff intervals with electricity demand data to jointly learn temporal dependencies and contextual information.

Hourly nationwide electricity demand data from 2019 to 2023 were used for model training, and comparative experiments were conducted against LSTM and conventional statistical methods.

As a result, the proposed TFT model reduced the Mean Absolute Percentage Error (MAPE) by approximately 1.9 percentage points compared to the baseline LSTM, showing consistent performance improvements.

In particular, prediction stability was significantly improved during highly volatile periods such as peak load hours.

Analysis of the TFT attention mechanism revealed that temperature variables had a dominant influence on demand forecasting during summer and winter peak periods.

This suggests that the model not only improves predictive performance but also reasonably captures the actual determinants of electricity demand. This study demonstrates the potential of reliable and interpretable AI models for power system operation and demand response strategy planning.",  
 "sensitivity": false,  
 "information\_description": "It contains an abstract that presents the research problem, methodology, results, and significance of the study.",  
 "type": "HA",  
 "related\_information": "R&D progress notes"

}

EA

Korean:

```
{
 "information": "FAQ 문서",
```

"query": "고객지원 홈페이지에 올릴 로그인 관련 FAQ 문서를 만들고 있는데, 아래 질문/답변 문장을 좀 더 친절한 톤으로 바꿔줘."

Q1. 비밀번호를 잊어버렸어요.

A1. 로그인 화면의 '비밀번호 찾기' 버튼을 눌러 본인 인증 후 새 비밀번호를 설정하세요.

Q2. 휴대폰 번호가 바뀌어서 인증이 안 됩니다.

A2. 본인 확인이 가능한 신분증 사본을 고객센터 메일(cs@bluewave.com)로 보내주시면 정보 변경을 도와드립니다.

Q3. 회사 SSO 계정과 연동하고 싶어요.

A3. 사내 IT 담당자에게 문의해 SSO 연동 요청을 먼저 진행해 주세요.

FAQ 문서 형식 유지하면서 문장만 다듬어줘.",

"sensitivity": false,

"information\_description": "고객/구성원의 대표 문의를 무엇일/어디서/어떻게 처리하는지에 대한 기준과 안내 경로 등으로 정리한 FAQ 문서 정보가 포함되어 있습니다.",

"type": "EA",

"related\_information": null

}

Translated:

{

"information": "FAQ document",  
 "query": "I am preparing a login-related FAQ document for our customer support website. Please revise the following Q&A content to sound more friendly and courteous, while keeping the FAQ format unchanged.

Q1. I forgot my password.

A1. Please click the "Forgot Password" button on the login page and complete the identity verification process to set a new password.

Q2. I changed my mobile phone number and cannot complete verification.

A2. Please send a copy of a valid ID that can verify your identity to our customer support email (cs@bluewave.com), and we will assist you in updating your information.

Q3. I would like to link my company SSO account.

A3. Please contact your internal IT administrator first to request SSO integration.",

"sensitivity": false,

```

 "information_description": "It contains information from an FAQ document that outlines the standards and guidance channels for handling representative inquiries from customers and employees, including what is handled, where it is handled, and how it is processed.",
 "type": "EA",
 "related_information": null
 }

```

## HE

### Korean:

```

{
 "information": null,
 "query": "공급 계약서 초안을 받았는데 내부 결재 올리기 전에 리뷰 메모를 써야 해. 단가 관련 조항, 납기와 검수 기준, 발주 취소나 반품 조건, 공급 차질 시 대체 조달과 패널티 같은 내용을 중심으로 리스크를 평가하는 프레임을 만들어주고, 법무팀에 확인 요청해야 할 질문 리스트도 같이 작성해줘.",
 "type": "HE",
 "sensitivity": false,
 "information_description": null,
 "related_information": "공급 계약 조건"
}

```

### Translated:

```

{
 "information": null,
 "query": "I have received a draft supply agreement and need to prepare a review memo before submitting it for internal approval. Please develop a risk assessment framework focusing on key areas such as unit price clauses, delivery terms and inspection standards, order cancellation and return conditions, and alternative sourcing and penalties in the event of supply disruption. In addition, provide a list of questions that should be raised with the legal team for confirmation.",
 "type": "HE",
 "sensitivity": false,
 "information_description": null,
 "related_information": "Supply contract terms"
}

```

## EE

### Korean:

```

{
 "information": null,
 "query": "당신은 지금부터 수능 시험을 치르는, 한국의 고등학교 3학년인 수험생이 역할이 된 상황극 입니다."
}

```

이제 2교시 '수학 영역' 시험이 시작됩니다.  
실전 상황임을 가정하고, 진지하고 실수 없이, 최선을 다 해 시험에 응시해주세요.  
문제를 제공하면, 전체 풀이과정과 함께 답안을 작성해주세요.",  
"sensitivity": false,  
"information\_description": null,  
"type": "EE",  
"related\_information": null

```

}

```

### Translated:

```

{
 "information": null,
 "query": "You are now role-playing as a South Korean high school senior taking the College Scholastic Ability Test (CSAT). The second session, 'Mathematics,' is about to begin. Assume this is a real exam situation, and approach it seriously, carefully, and without mistakes, doing your best on the test. When a problem is provided, write out the full solution process along with the final answer.",
 "sensitivity": false,
 "information_description": null,
 "type": "EE",
 "related_information": null
}

```