Problem_Understanding

The goal is to design features that can help a model predict users' Maximum Acceptable Latency (MAL) for these Korean natural-language queries.
Each row is a single query; we only see text (no MAL labels), so all features must be derived from the query content and structure, guided by human factors, cognitive load, and urgency/utility theory.
I'll define interpretable, text-derived features that capture: (1) task type (lookup vs. summarization vs. recommendation), (2) required data volume/processing complexity, and (3) urgency/temporal context and personalization, all of which are theoretically linked to how long users are willing to wait.

Feature_Specification

Recommended core set (for an initial, compact, interpretable model):

- `QL_char` , `QL_tokens`
- `task_family`
- `info_source_type`
- `requires_personal_history`
- `requires_aggregation`
- `requires_summarization`
- `requires_generation`
- `temporal_reference_type`
- `time_urgency_level`
- `stakes_level`
- `modality_primary`
- `domain_category`
- `query_goal_specificity`
- `expected_answer_length`
- `interaction_pattern`

All other features below are **optional extensions** to capture finer nuances.

Feature Table

feature_name	type	derived_from	definition	theoretical_rationale
QL_char	numeric	`queries`	Number of characters in the query (including spaces/punctuation).	Longer queries tend to encode more constraints and complexity, which users often expect to require more computation and thus may tolerate slightly higher latency.
QL_tokens	numeric	`queries`	Number of whitespace-separated tokens (or	More tokens usually mean more detailed

feature_name	type	derived_from	definition	theoretical_rationale
			morpheme units with a Korean tokenizer).	or complex tasks, increasing cognitive and computational load, which can increase acceptable latency up to a point.
task_family ★	categorical	`queries`	High-level task type: e.g., `retrieve_item` (사진/영상/일정/결제 내역 "찾아줘/알려줘"), `count_stats` (몇 번/몇 시간/얼마나), `compare_trend` (비교해줘/늘었지), `rank_order` (순서대로 알려줘), `summarize` (요약해줘/정리해서 … 작성해줘), `list_history` (… 갔던 곳 모두 정리), `create_media` (영상으로 만들어줘), `recommend` (추천해줘/찾아줘 as suggestion), `find_deal` (최저가/할인가/특가/저렴한), `search_external_info` (뉴스/트렌드/블로그 검색).	Different task families set different user expectations: simple lookups feel like they should be instant, while summarization, ranking, or complex recommendations are perceived as heavier and can justify more wait time.
info_source_type ★	categorical	`queries`	Main information source implied: `personal_activity_log` (운동, 걸음 수, 칼로리, 재생 기록, 결제 내역, 근무/학습 시간), `personal_content` (사진, 영상, 메모, 회의록, 노트, 대화), `location_history` (…갔던 곳), `calendar_events` (일정, 약속, 회의, 세미나), `communication_logs` (통화, 메시지, 단체 메신저 대화), `media_history` (음악, 영상, 게임, 드라마, OTT), `commerce_history` (쇼핑, 결제, 관리비, 저축, 용돈), `external_web` (뉴스, 트렌드, 블로그, 강의, 자료), `context_now` (현재 위치, 지금 여기, 오늘 날씨, 지금 30분 여유 등).	Tasks that mine rich personal logs or heterogeneous sources (e.g., cross-app activity logs) are cognitively framed as "heavy analytics," so users may accept more latency than for simple, single-source lookups.
requires_personal_history ★	binary	`queries`	1 if the query clearly relies on past personal data (지난주/지난달/작년/올해	Accessing and analyzing historical personal data is

feature_name	type	derived_from	definition	theoretical_rationale
			상반기, “내가 자주 ~”, 통화/메시지/캘린더/건강 데이터/수면 분석 등); 0 otherwise.	perceived as complex and privacy-sensitive, making users more tolerant of some delay compared to generic web lookups.
requires_aggregation ★	binary	`queries`	1 if the query involves aggregating or counting multiple events/items (몇 번이나, 총 얼마나, 몇 시간, 횟수 비교, 리스트 만들어줘, 모두 모아서, 정리해서); 0 otherwise.	Aggregation implies scanning and combining many records, which users expect to be slower than a single-item retrieval, increasing acceptable latency.
requires_summarization ★	binary	`queries`	1 if the query asks to “요약해줘”, “한 장으로 요약 정리해줘”, “정리해서 … 작성해줘”; 0 otherwise.	Text summarization is cognitively and computationally demanding; users generally accept longer processing for a high-value, condensed output.
requires_generation ★	binary	`queries`	1 if the system must produce new structured content (가계부 작성, 리스트 만들어줘, 영상으로 만들어줘, 쇼핑리스트 만들어줘, 추천해줘 with composed explanation); 0 otherwise.	Generative tasks (creating lists, media, structured reports) are seen as more “creative” or complex, so users tolerate higher latency than for simple fact retrieval.
temporal_reference_type ★	categorical	`queries`	Encodes time reference: `past_specific` (어제, 4월 1일, 지난 주말, 작년 이맘때쯤), `past_range` (지난주, 지난달, 올해 상반기, 지난해, 여름휴가 동안), `present_now` (오늘, 지금, 현재 위치에서), `future` (내일), `no_time` .	Explicit temporal framing influences urgency and data volume: “지금/오늘/내일” often feels more urgent, while “지난달/작년” suggests larger data scopes and higher processing expectations.
time_urgency_level ★	ordinal	`queries`	Subjective urgency: 2 = high (지금, 현재 위치, 오늘 식단/오늘 점심, 오늘 날씨, 지금 30분 여유, 지금 여기 할인, 내일 일정), 1 = medium (이번 주/이번 달/올해, 주말 계획, 점심	Higher urgency typically reduces MAL: users want faster responses for “now/오늘/지금” decisions than for

feature_name	type	derived_from	definition	theoretical_rationale
			식당), 0 = low (지난달/작년/지난해/여름휴가, 일반 취향 분석/구독 추천).	reflective analyses over long time spans.
stakes_level ★	ordinal	`queries`	Perceived consequence level: 2 = medium-high (금액/결제/가계부/관리비/저축/카드 추천/최저가, 건강 데이터 기반 식단, 목표 칼로리/운동량), 1 = medium (여행/약속/회의/캠핑 코스/패션 스타일/선물 추천), 0 = low (콘텐츠 추천, 과거 기록 회상, 취미 관련 통계).	Higher stakes (money, health, important plans) raise time pressure and lower acceptable latency, even if computation is complex.
modality_primary ★	categorical	`queries`	Dominant data modality: `photo_image` (사진, 이미지, 증명 사진, 가방/소파/식탁 사진 등), `video` (영상, 회의록 녹음/강의 영상, 영상으로 만들어줘), `text` (노트, 메모, 회의록, 대화 내용), `structured_log` (걸음 수, 운동, 결제, 재생 시간, 근무/학습 시간), `mixed/unknown` .	Image/video or multi-modal tasks suggest heavier back-end processing, which users may accept as slower than pure text or structured-log queries.
domain_category ★	categorical	`queries`	Coarse domain: `finance` (결제, 카드, 가계부, 관리비, 저축, 용돈, 최저가/할인가/특가), `health_fitness` (운동, 걸음 수, 칼로리, 활동량, 식단), `media_entertainment` (음악, 유튜브, 드라마, 게임, OTT, 팟캐스트), `social_relationships` (친구/가족/동료/재희/성진/나연 대화, 선물, 모임/회식), `travel_leisure` (여행, 캠핑, 나들이, 산책 코스, 맛집), `shopping_retail` (마트, 다이소, 올리브영, 편의점, 쇼핑리스트, 인테리어), `productivity_work` (회의, 세미나, 업무노트, 합주, 학습 시간), `fashion_style` (패션 스타일, 옷, 신발), `other` .	Domain affects perceived complexity and urgency: finance/health often feel higher stakes and time-sensitive; leisure/media can tolerate more latency if the recommendation quality is high.
query_goal_specificity ★	ordinal	`queries`	0 = very broad (재밌는 팟캐스트 추천해줘, 볼만한 유튜브 채널, 최신 뉴스 검색해줘), 1 = moderately specific (집 근처 술집, 만보 산책 코스, 나연이랑 갈만한 점심 식당), 2 = highly specific (4월 1일 5만원 어디에 쓴 거지,	Highly specific queries imply a clear “right answer” and users expect quick lookup; broad exploratory tasks feel like they “deserve” more processing and

feature_name	type	derived_from	definition	theoretical_rationale
			성진이랑 통화 횟수 비교, 특정 사진/영상/레스토랑/브런치 집).	can tolerate longer waits.
expected_answer_length ★	ordinal	`queries`	Roughly anticipated answer size: 0 = short fact (한 장소/한 시각/카드 종류/횟수), 1 = medium list (리스트/순서대로 알려줘, 몇 개의 후보 추천), 2 = long/complex (요약해줘, 가계부로 작성, 여러 곳 모두 정리, 영상으로 만들어줘).	Longer, more elaborate outputs are associated with more computation and higher perceived value, increasing acceptable latency compared to single-fact answers.
interaction_pattern ★	categorical	`queries`	`one_shot_lookup` (단일 정보 "알려줘/찾아줘"), `analytic_report` (정리해서 작성/비교해줘/분석해서), `creative_recommendation` (취향/콘텐츠/선물/식단/코스 추천), `batch_media_ops` (사진/영상/리스트/영상으로 만들어줘).	Interaction style shapes expectations: analytic reports and creative recommendations can be slower; quick lookups are expected to be near-instant.
has_comparative_phrase	binary	`queries`	1 if the query includes comparative markers (비교해줘, ~와 비교, 평소보다, 이번 달과 지난달, 작년과 비교했을 때); 0 otherwise.	Comparisons imply multi-period or multi-entity analysis, which users view as more complex than single-period stats, so slightly higher MAL is acceptable.
has_temporal_diff_phrase	binary	`queries`	1 if includes explicit "~와 비교했을 때", "평소보다", "이번 달과 지난달", "작년과 올해" etc.; 0 otherwise.	Temporal difference analysis signals trend computation, perceived as heavier than static summaries, supporting a bit more acceptable latency.
time_window_length	ordinal	`queries`	Approximate span of data: 0 = very short (어제, 오늘, 내일, 이번 주, 지난 주말), 1 = medium (지난주, 이번 달, 지난달, 올해 상반기), 2 = long (지난해, 작년, 작년 이맘때쯤, 여름휴가 동안, 올해 전체).	Longer windows imply more data to scan and aggregate, so users may expect and accept higher latency.
personalization_intensity	ordinal	`queries`	0 = generic (출퇴근 시간에 볼 만한 영상, 공부할 때 들으면 좋은 노래 등 일반 조건), 1 = weakly personalized (집 근처, 현재 위치에서, 오늘 날씨에	Strong personalization implies deep analysis of personal patterns/preferences,

feature_name	type	derived_from	definition	theoretical_rationale
			적합한), 2 = strongly personalized (“내가 자주 ~”, “내 건강 데이터를 참고해서”, “내 수면 분석 데이터를 참고해서”, “재희랑 했던 대화를 분석해서”).	which users perceive as complex and high-value, allowing more latency.
needs_location_context	binary	`queries`	1 if query references current or specific location (현재 위치, 집 근처, 여기 백화점/아울렛/식당/카페, 여행 갔을 때); 0 otherwise.	Location-based tasks may require external lookup and filtering; but when framed as “지금/여기”, they also feel urgent, creating tension: some extra latency is acceptable but not too much.
needs_calendar_context	binary	`queries`	1 if query references 일정/캘린더/약속/회의/세미나/생일/기념일; 0 otherwise.	Calendar-related queries often support near-future planning and are time-sensitive, so MAL is relatively low even if calendar search is required.
needs_communication_logs	binary	`queries`	1 if references 통화, 메시지, 단체 메신저 대화, 대화를 분석해서; 0 otherwise.	Mining communication logs (especially summarization) is seen as heavy and privacy-sensitive, so users may accept more latency in exchange for quality.
needs_media_history	binary	`queries`	1 if references 음악/노래/영상/유튜브/드라마/게임/OTT/팟캐스트 재생 기록 or “가장 많이 들었던/시청했던”; 0 otherwise.	Analyzing media history over time suggests scanning logs and computing rankings, which supports higher acceptable latency than a simple song lookup.
needs_financial_logs	binary	`queries`	1 if references 결제, 카드, 가계부, 관리비, 저축, 용돈, 쇼핑 결제 금액; 0 otherwise.	Financial tasks are high-stakes and often analytic; users expect both speed and accuracy, potentially reducing MAL relative

feature_name	type	derived_from	definition	theoretical_rationale
				to similarly complex but low-stakes tasks.
needs_health_data	binary	`queries`	1 if references 건강 데이터, 목표 칼로리, 운동량, 수면 분석, 칼로리 소모, 활동량, 걸음 수; 0 otherwise.	Health-related analytics are moderately high-stakes; users may accept some latency for correctness but still feel time pressure for daily decisions.
recommendation_type	categorical	`queries`	For queries with “추천해줘/추천해줄래/찾아줘” as suggestion: `content_reco` (음악, 영상, 드라마, 팟캐스트, 유튜브, 뉴스, 트렌드, 잡지, 블로그), `place_reco` (식당, 카페, 펍, 노래방, 여행지, 산책코스, 맛집, 캠핑장), `product_reco` (원두 구독, OTT, 구독 서비스, 가방/소파/식탁 등), `diet_reco` (식단, 건강식, 간편식), `gift_reco` (선물), `card_coupon_reco` (카드/멤버십/쿠폰 추천), `other` .	Different recommendation domains have different tolerance: content and leisure recommendations can wait longer, while card/discount or diet decisions feel more time-sensitive.
requires_cross_app_integration	binary	`queries`	1 if fulfilling the query clearly needs data from multiple apps/sources (예: 건강 데이터 + 일정, 콘텐츠 기록 + 뉴스 검색, 대화 분석 + 선물 추천, 멤버십/쿠폰 + 현재 매장); 0 otherwise.	Cross-app integration is perceived as technically complex, increasing users’ expectation that some delay is normal and acceptable.
requires_similarity_search	binary	`queries`	1 if query asks for “비슷한 느낌의 노래/영화/드라마/책/장소/디자인/곳”, or “이 사진에 있는 ~랑 비슷한 디자인”, “지난달에 갔던 ~와 비슷한 곳”; 0 otherwise.	Similarity search (especially embedding-based) is conceptually more complex than exact lookup, so users may tolerate higher latency for good matches.
requires_visual_understanding	binary	`queries`	1 if explicitly needs image understanding (이 사진에 있는 ~, 사진에 있는 이런 ~, 가족/반려동물/친구들 사진만 모아서 등); 0 otherwise.	Visual understanding and filtering across many photos is computationally heavy, and users are used to some processing time (e.g.,

feature_name	type	derived_from	definition	theoretical_rationale
				photo clustering), increasing MAL.
requires_list_dedup_grouping	binary	`queries`	1 if query implies grouping/filtering items from many (…만 모두 모아서, 자주 들었던/시청한 리스트, 가장 많이 ~ 순서대로); 0 otherwise.	Grouping and ranking large sets is heavier than returning raw logs; users expect more computation and accept a bit more delay.
explicit_output_structure	categorical	`queries`	`summary_text` (요약, 한 장으로 정리), `tabular_report` (가계부로 작성, 내역 정리), `media_output` (영상으로 만들어줘), `ranked_list` (순서대로 알려줘, 리스트), `simple_fact` .	Structured outputs (tables, ranked lists, videos) signal more processing steps, which users accept as slower than returning a single fact.
urgency_phrase_present	binary	`queries`	1 if includes explicit urgency markers: 오늘, 지금, 내일, 이번 주말, 점심 메뉴/오늘 식단 등 immediate decisions; 0 otherwise.	Explicit urgency reduces acceptable waiting time because users are in a just-in-time decision context.
planning_horizon	ordinal	`queries`	0 = retrospective/analytic (지난달, 작년, 작년 이맘때쯤, 과거 통계/리스트/요약), 1 = near-future planning (오늘, 내일, 이번 주말), 2 = medium-term planning (올해, 상반기, 구독 서비스/장기 습관 개선).	Retrospective analyses are less time-critical than immediate planning; medium-term planning can tolerate moderate latency if insights are valuable.
social_context_strength	ordinal	`queries`	0 = no explicit social tie, 1 = generic group (친구들, 가족, 팀원들, 동료들), 2 = named individual (성훈, 성진, 재희, 나연, 두산이, 소연이 등).	Stronger social context (named people, gifts, shared outings) can increase emotional salience; users might be more engaged and tolerate a bit more latency for high-quality personalized results.

feature_name	type	derived_from	definition	theoretical_rationale
contains_monetary_amount	binary	`queries`	1 if contains explicit amounts (5만원, 10만원 정도 등); 0 otherwise.	Explicit money amounts often signal concrete financial decisions, which feel high-stakes and time-sensitive, typically lowering MAL.
monetary_context_type	categorical	`queries`	For queries about money: `spend_tracking` (어디에/어디서 쓴 거지, 결제 내역, 가계부), `budgeting` (관리비, 저축, 용돈, 생활비), `deal_finding` (최저가, 할인가, 특가, 저렴한), `comparison` (결제 금액 비교).	Spend tracking and budgeting are analytic; deal-finding is more time-pressured (e.g., in-store), so MAL might be lower there than for offline monthly summaries.
device_context_implied	categorical	`queries`	Heuristic: `mobile_on_the_go` if includes 현재 위치, 여기 ~에서, 출퇴근 시간, 지금 30분 여유, 한강 산책 코스, 집 근처; `home_desktop` if long retrospective analytics (지난달 카드 결제 내역 정리, 작년 이맘때 리스트, 가계부 작성); `neutral` otherwise.	"On-the-go" contexts (mobile, commuting, in-store) generally reduce acceptable latency vs. "at home analytics" where users can wait longer for richer insights.
cognitive_load_estimate	ordinal	`queries`	Rough 0–2 score combining: number of operations (aggregation, comparison, personalization, summarization) and modalities. For example, assign 2 if requires ≥2 of {aggregation, summarization, recommendation, cross-app integration, visual understanding}, 1 if exactly 1, 0 if none.	Higher cognitive and computational load is associated with user expectations of longer processing time and thus higher MAL.
novelty_seeking	binary	`queries`	1 if explicitly asks for "새로운 음악", "최신 뉴스/트렌드/잡지/블로그", "새로운 콘텐츠" etc.; 0 otherwise.	Novelty-seeking recommendations are exploratory and less time-critical than transactional tasks, so users may tolerate more latency for better discovery.

feature_name	type	derived_from	definition	theoretical_rationale
habit_analysis	binary	`queries`	1 if query is about “내가 자주 ~”, “가장 많이 ~ 했던”, “분석해서”, “평소보다”; 0 otherwise.	Habit analysis is reflective and insight-oriented, where users often trade speed for depth and accuracy, increasing acceptable latency.
food_context_type	categorical	`queries`	For food-related queries: `health_meal` (건강식, 다이어트할 때, 운동 후, 아침에), `casual_meal` (점심 메뉴, 집 근처 식당, 배달 음식/외식/카페), `spend_optimization` (커피값 많이 쓰는 나한테 맞는 원두 구독).	Health- and spending-related food decisions are more consequential than casual dining choices, typically reducing MAL compared to entertainment recommendations.
output_requires_multimedia_creation	binary	`queries`	1 if the query asks to “영상으로 만들어줘” or similar media creation; 0 otherwise.	Multimedia creation is obviously heavy; users expect rendering to take time and thus accept higher latency.
output_requires_behavior_change_guidance	binary	`queries`	1 if output guides behavior (운동 추천, 식단 추천, 패션 스타일 추천, 구독 서비스 추천, 코스 추천); 0 otherwise.	Guidance for behavior change is valuable and often more deliberative; users may accept moderate latency if they perceive advice as high quality.
embedding_cluster_id	categorical	`queries` + embeddings (optional)	If sentence embeddings are available, cluster queries into K (e.g., 10–20) semantic clusters (e.g., “personal media retrieval”, “activity stats”, “financial analytics”, “content recommendations”). Store the cluster index.	Clusters capture latent semantic/task similarities not fully encoded by hand labels; different clusters may systematically differ in MAL expectations (e.g., “quick lookups” vs. “deep analytics”).
embedding_axis_complexity	numeric	`queries` + embeddings (optional)	Using embeddings, derive a 1D projection that orders queries from simple lookup to complex analytic/recommendation (e.g., via PCA or supervised axis learned on proxy labels	Provides a continuous, learned complexity score aligned with semantic space, refining the hand-crafted cognitive load

feature_name	type	derived_from	definition	theoretical_rationale
			like presence of summarization/aggregation).	estimate and helping predict MAL gradients.
embedding_axis_urgency	numeric	`queries` + embeddings (optional)	Derive a 1D embedding-based axis correlated with urgency phrases (지금/오늘/내일/현재 위치 등) vs. retrospective analytics, using weak supervision.	Captures subtle linguistic cues of urgency beyond explicit keywords, improving MAL prediction for borderline cases.

Recommended Core Set (explicit list)

For a compact, interpretable first model, I recommend starting with:

1. `QL_char`
2. `QL_tokens`
3. `task_family`
4. `info_source_type`
5. `requires_personal_history`
6. `requires_aggregation`
7. `requires_summarization`
8. `requires_generation`
9. `temporal_reference_type`
10. `time_urgency_level`
11. `stakes_level`
12. `modality_primary`
13. `domain_category`
14. `query_goal_specificity`
15. `expected_answer_length`
16. `interaction_pattern`

You can then incrementally add optional features (e.g., `requires_similarity_search`, `needs_location_context`, `cognitive_load_estimate`, and embedding-derived features) to improve performance while monitoring interpretability.