**Applications of NL(X) and LLM**

**Names:** Diego Bermudez

**Date:** September 30, 2025

**Ground the Domain – From Naive RAG to Production Patterns**

**Phase 1 – Data Exploration**

I loaded the RAG Mini-Wikipedia passages directly from Hugging Face via Parquet. The raw passages dataset contains 3,200 rows. After normalizing the text (lower-casing, whitespace collapse, ASCII cleanup) and dropping nulls, I identified 16 exact duplicates (0.5%), yielding 3,184 unique passages for indexing. Some profiling features (e.g., token counts) were computed on a cleaned analysis subset of 3,086 rows after excluding blanks and very short artifacts as a result of the normalization process.

**Figure 1. Document Length Distribution**

**A graph with a number of numbers

AI-generated content may be incorrect.**

Length is right skewed with a long tail (Figure 1). The normalized character length follows a distribution with mean 378, median 289, IQR between 104 and 557, with a max of 2,436 words. At the word-level, I observed a mean 64 words and an approximate mean token count of 77 (max observed 500). Only a small tail exceeds typical 300 to 400 tokens, but those long passages justify chunking. Given this distribution, I decided to use sentence-aware chunks of 350 tokens with 50-token overlap to improve recall and reduce prompt bloat, this to consider those right distributed passages. Non-ASCII characters are negligible (mean 0.24% of chars). Punctuation and digits account for 3.6% of our dataset, which is consistent with this encyclopedia-like dataset.

Part-of-Speech (POS) counts are dominated by NOUN and PROPN, confirming entity-dense content. Named-entity distribution is broad: ORG, GPE, PERSON, DATE lead, with additional NORP, LOC, WORK\_OF\_ART, LAW, LANGUAGE (This categorization will be useful further in the project to perform feature reranking across the predictions). The LDA analysis unveiled the heterogeneity present in the topics across the dataset. Particularly, I fund the following topics to be predominant through the documents: 1) wildlife/biology (e.g., bear, beetle, leopard), 2) countries/demography (Romania, Singapore, population), 3) history/law/education (war, court, act), 4) US politics (Roosevelt, president, election), and 5) academia/empire/engineering (university, English, Tesla). This spread deepen in the idea that using reranking will be beneficial for the retrieval process, by improving the topic based precision.

**Figure 2. Distribution of Named Entity Labels (NER)**

A graph of a number of entity labels

AI-generated content may be incorrect.

**Figure 3. Distribution of Part-of-Speech (POS) Tags**

A graph of a number of different types of speech tags

AI-generated content may be incorrect.

**Figure 4. LDA Results**

A screenshot of a graph

AI-generated content may be incorrect.

I worked in Python 3.10 with JupyterLab for EDA, following this set up: Data IO uses pandas, pyarrow==21.0.0, and fastparquet, pulling parquet from Hugging Face via datasets/huggingface-hub. Embeddings are produced with sentence-transformers==5.1.1; generation runs through transformers==4.56.2 on torch==2.5.1 (tokenization utilities via tiktoken). The vector store is Milvus (local/Lite) via pymilvus==2.6.2; we’ll create an HNSW or IVF index and persist IDs for grounded citations. For chunking and orchestration we’ll use langchain, langchain-text-splitters, and (optionally) langsmith for experiment tracking. Linguistic profiling uses spacy==3.8 with en\_core\_web\_md, and topic modeling/visualization uses gensim + pyLDAvis. Evaluation includes ragas==0.3.5; classic baselines and diagnostics use scikit-learn, with plots in matplotlib, seaborn, or plotly.

**Phase 2 – Naive Architecture Building**

In the *utils.py* module, I centralize the shared building blocks to keep the pipeline reproducible, observable, and easy to grade. I seed *Python* and *NumPy* for deterministic runs, and I use a lightweight logger so every major step (load, embed, index, search, generate) is timestamped without print spam. I load passages and queries from local CSVs to avoid network variability during evaluation and to keep the schema explicit. Each passage gets a normalized variant for duplicate/quality checks and quick length metrics; the *tiktoken*‐based token estimate helps me reason about future chunking without adding that complexity into the naive baseline. For embeddings, I use *all-MiniLM-L6-v2* because it offers a strong retrieval signal at low latency and memory cost; I normalize vectors so inner product equals cosine, which matches both common retrieval practice and my Milvus settings.

For storage and retrieval, I wrap *MilvusClient* in a small *MilvusIndex* abstraction. I chose Milvus Lite (file-backed) over a remote service to remove operational dependencies while retaining a production-style schema: id, passage, and embedding, also because the number of documents in our dataset is small, and does not require high computational capabilities. I index with HNSW because it consistently delivers a good recall/latency trade-off on small-to-mid corpora; using the IP metric with normalized vectors gives me cosine similarity without extra transforms. The API is intentionally tiny, so I can swap implementations later (e.g., server Milvus, reranking layers) without touching the rest of the code. On the generation side, I keep the prompt compact and grounded (“use only context; otherwise say you don’t know; cite passage IDs”) to reduce hallucinations and to make answers auditable. I use FLAN-T5-base for wide availability, decent instruction following, and predictable inference; temperature=0.0 ensures reproducible grading.

In the naive\_rag.py module, I keep orchestration minimal to isolate variables for later ablation. *build\_index* embeds once, creates the collection, inserts all rows, and prints entity counts and schema as sanity checks. retrieve embeds questions and returns top-k (id, passage) pairs; those IDs flow into the prompt for traceability. The run function wires the steps end-to-end and writes a compact JSON record {qid, question, prediction, gold, context\_ids} for downstream EM/F1 analysis. I expose all meaningful knobs via CLI—models, batch sizes, top\_k, Milvus options, decoding budget, and evaluation set size—so experiments are easy to script and reproduce.

I deliberately omit chunking, reranking, and query rewriting in this phase to keep the baseline simple and measurable. That clarity lets me attribute future gains (e.g., adding a cross-encoder reranker or sentence-aware chunking) to specific enhancements rather than to pipeline churn.

**Phase 3 – Evaluation Phase**

I ran three prompting strategies on the same naive RAG configuration—MiniLM embeddings, Milvus Lite IVF\_FLAT, top-1 passage retrieval, and FLAN-T5-base at temperature 0—over 200 queries. Using Hugging Face SQuAD for macro EM/F1 and per-question F1 to form normal distributed CIs, I found a clear ordering across performance results: persona > chain-of-thought (CoT) > instruction. Instruction lands at F1 = 0.443 (95% CI [0.378, 0.508]) and EM = 0.370 with a 15.5% “unknown” rate. CoT improves to F1 = 0.489 ([0.424, 0.554]) and EM = 0.415 with only 2% unknown. Persona leads at F1 = 0.497 ([0.432, 0.563]) and EM = 0.425 with a moderate 5.5% unknown rate.

A number of numbers on a white background

AI-generated content may be incorrect.

Persona works best by a small but consistent margin. My analysis from the results is that Persona’s framing (e.g., domain-appropriate, citation-oriented, concise, and explicitly grounded) pushes the model toward faithful extraction rather than speculative synthesis. That balance appears crucial under retrieval constraints: it forces the model to answer when the evidence is reasonably explicit, while still abstaining when context is thin. The gains over instruction are meaningful (0.055 F1/EM) and survive the uncertainty in the CIs; the edge over CoT is narrower and CI-overlapping, so I treat persona vs CoT as effectively tied with persona slightly ahead. In other words, there is not enough statistical evidence to confirm that Persona excels CoT in this task, but, even for a low margin it generate slightly better results.

Error patterns suggest that abstention behavior and recall are the primary drivers. Instruction’s higher unknown rate reflects a conservative reading of the grounding rule; with only one passage, that often suppresses legitimate answers that need light paraphrase. CoT, by contrast, nearly eliminates refusals and lifts accuracy, but it occasionally “fills connective tissue” that isn’t strictly sourced when evidence is borderline—my output-format guardrails kept verbosity down, yet the approach didn’t surpass persona. Persona sits between these extremes, delivering better calibration: fewer unnecessary refusals than instruction, fewer speculative bridges than unconstrained CoT.

The ceiling across all prompts (F1 and EM does not surpass 0.5) signals a retrieval bottleneck, not a generation one. When the single retrieved passage is off-target or incomplete, prompt style yields diminishing returns. To move the needle, I would keep persona as the default system style and strengthen retrieval next: increase to top-k=3, apply max-marginal-relevance to diversify passages, and add a cross-encoder re-ranker to surface the most answerable snippet.

**Phase 4 – Experimentation**

I evaluated the full grid of Step-4 configurations using the naming scheme naive\_results\_{ret\_docs}\_emb\_{emb\_size}\_{strat} (e.g., naive\_results\_top-5\_emb\_384\_persona). The experimental objective was to isolate the impact of testing multiple combinations of embedding dimensionality, retrieval depth, and strategy while keeping the generator and prompt family constant. I reused the Step-3 methodology end-to-end, to ensure interpretability and consistency across the analysis. Every run appends a single summary row to the comparison table, so the analysis here references those tags directly for traceability.

Across the set {256, 384, 512}, the trend is consistent with retrieval physics. 384-D is the most stable Pareto point with high accuracy keeping a steady latency. 256-D shows a small, systematic drop in both EM and F1, which aligns with the information loss expected from dimensionality reduction; cosine similarity is preserved after re-normalization, but the local neighborhood structure suffers just enough to matter at top-k. 512-D is competitive and occasionally wins when paired with deeper retrieval (top-10), but gains are marginal versus 384-D and come with larger collections and longer ANN scans. The CIs on per-question F1 make this concrete: 256-D intervals sit below or barely overlap 384-D, while 512-D intervals typically overlap 384-D, indicating no robust lift commensurate with the extra cost.

Moving beyond top-1 is the primary lever. Naive top-k () improves recall and lifts EM/F1 for 3 and 5, after this point returns diminish and verbosity risks rise. MMR-diverse selection on a larger candidate consistently outperforms naive top-k at the same k, with the largest gap at k=5; the diversified set reduces near-duplicate passages and raises the chance that at least one context contains an extractable span. Where included, cross-encoder reranking yields the most decisive improvements at k=3 (precision jump with minimal prompt bloat), but the advantage narrows at k=10, where long context starts to strain the generator’s focus. These patterns are mirrored in the “unknown” metric: instruction-style prompts abstain less as k increases, but persona retains the best balance—fewer unnecessary refusals than strict instruction, fewer off-context bridges than unconstrained CoT.

A table with numbers and a number of data

AI-generated content may be incorrect.

The strongest, reproducible setting in this sweep is :384-D, top-5, diversified retrieval via MMR, and persona based prompt. It offers the best accuracy/abstention trade-off while keeping latency reasonable. The primary improvements of this setting are: 1) 384-D preserves the encoder’s neighborhood geometry; 2) k=5 is a sweet spot for recall without overwhelming the generator; 3) diversity controls redundancy, raising answerability; and 4) persona’s grounded, concise style turns the improved context mix into cleaner spans and fewer speculative jumps. The confidence Intervals support these claims by separating this setting from 256-D across K’s and showing overlap with 512-D at most K’s.

All runs share fixed seeds, identical query sets, normalized IP similarity, and per-run logging of dimension, k, strategy, and index parameters. Errors (missing files, index/schema mismatches) are caught early with explicit messages, and every result is persisted with its tag for auditability. The results table in the repository (comparison\_analysis.csv) summarizes tag, dim, k, strategy, n, em\_mean (HF), f1\_mean (HF/local), f1\_ci, unknown\_rate, meeting the documentation requirement and enabling one-line deltas between configurations.

**Phase 5 – Advanced RAG Features**

In the enhanced\_rag.py module, I extend the baseline with two targeted upgrades: a Cross-Encoder re-ranker and a confidence scoring with abstention. I kept the orchestration identical to the naive path: load CSV passages and queries, embed with sentence-transformers, and index in Milvus Lite. Since local Milvus Lite is stricter on index options, I default to IVF\_FLAT with inner product over normalized vectors, preserving cosine behavior without operational surprises. For each question I retrieve a candidate pool (e.g., 50), then re-rank those query-passage pairs using cross-encoder. The re-ranker outputs a scalar relevance per pair; I sort, keep the top-k contexts, build the grounded prompt, and generate with the base model setting to maintain reproducibility.

I implement confidence as two interpretable signals: the top score and the margin between the best and second-best re-rank scores. Both are written to the results JSON alongside context\_ids. When --abstain\_on\_low\_conf is enabled, the system returns “I don’t know.” if the top score falls below --min\_conf or the margin below --min\_margin. This yields a calibrated refusal policy that suppresses weakly supported answers while preserving strong, short, citation-bearing generations. Importantly, this confidence logic lives downstream of retrieval but upstream of generation, so it works uniformly across prompts and decoding stages.

By using re-ranking, I managed to improve precision at the top of the context set (eg., k between 3 and 5), which is reasonable for seq2seq models with small context windows. Because I re-rank only the ANN shortlist, the cost scales with the candidate count, not corpus size. The CLI mirrors the baseline by adding --retrieve\_candidates, --rerank\_model, and confidence thresholds, so parameter sweeps remain scriptable.

There are some pitfalls to acknowledge from this framework, particularly, cross-encoders add latency proportional to the candidate pool; on CPU, scoring 50 pairs per query took 2x the time the naive model did. The model is also trained on MS MARCO; on out-of-domain data its calibration can drift, which affects both ranking quality and confidence margins. The confidence rule itself is heuristic, not a calibrated probability, so the thresholds may inflate abstentions and depress EM, or invite hallucinations if defined higher or lower to the optimum. Finally, IVF\_FLAT in Milvus Lite trades some recall versus HNSW on large collections; if the corpus grows, I would revisit index choice or move to a server Milvus with HNSW/IVF-PQ. In general, my enhanced pipeline preserves the baseline’s simplicity, but adds a precision boost to the prediction itself (top-k), and introduces an auditable safety valve via abstention.

**Phase 6 – RAGAs Evaluation**

I evaluated both the naive and enhanced RAG pipelines with RAGAs, using a single LLM judge (gpt-4o-mini) and the same held-out set (approx. 200 items). The notebook calls a hardened wrapper (ragas\_evaluation.eval\_one) that converts each run’s results JSON into a RAGAs-compatible dataset with fields {question, response/answer, contexts, reference}, I restricted the number of contexts per item to 5 and trimmed each to 1800 chars to control token growth, then evaluates in batches of 25 observations to mitigate judge timeouts. I logged per-batch failures and aggregated successful batches with a weighted mean. This produced four summaries: three naive top-1 variants (instruction, persona, CoT) and one enhanced configuration (top-5 retrieve, cross-encoder rerank, persona prompting).

With this setting I found that among naive RAG’s, CoT yields the highest answer relevancy (0.70) and the best context precision (0.032), while persona has the strongest context recall (0.036); the differences are modest but consistent with my earlier EM/F1 analysis where CoT/persona outperformed plain instruction. Second, the enhanced run shows a clear increase in context recall (0.0542 vs 0.023, and 0.036) as expected from retrieving more documents and re-ranking, but faithfulness and context precision collapsed to 0.0 and answer relevancy dropped sharply (0.2881). In other words, the enhanced system retrieved more potentially relevant text but the judged alignment between answer and grounded evidence was poor and answers themselves were judged less relevant.

Interpretation requires caution because this stage encountered substantial API timeouts from the LLM judge. Despite sharding and retries, several batches failed and were excluded. This can bias summary means in two ways: 1) if failures are not uniformly distributed across “easy” and “hard” questions, the surviving shards may over-represent harder cases; 2) longer contexts (top-5) increase token sizes and LLM latency, making enhanced runs more susceptible to timeout and truncation artifacts. Additionally, persona outputs can be concise and evidence-lean; RAGAs faithfulness penalizes answers that do not explicitly ground to the provided contexts, so even correct but terse responses can be marked unfaithful if the judge cannot connect them back to spans within the trimmed contexts.

That said, between naive architectures, CoT/persona superiority on answer relevancy suggests generation style matters when it comes to recall over the passages; CoT’s planning tends to produce on-topic answers that the judge deems helpful. The enhanced pipeline improved context recall, indicating the reranker is surfacing more answer-bearing text overall, but the simultaneous drop in precision/faithfulness points to either over-inclusive context windows, re-ranker miscalibration, or prompt-answer formatting that fails the judge’s grounding test (e.g., insufficient span anchoring or citations).

**Phase 7 – Summary Report**

This project builds a transparent, reproducible RAG pipeline over a Mini-Wikipedia corpus and progresses from a naive baseline to an enhanced system with re-ranking and confidence-based abstention. The baseline uses Sentence-Transformers (all-MiniLM-L6-v2) for embeddings, Milvus Lite (IVF\_FLAT, IP on normalized vectors) for approximate search, and FLAN-T5-base for answer generation under grounded prompts. Step-3 experiments compared three prompting strategies under top-1 retrieval (n=200): persona and chain-of-thought (CoT) outperformed a plain instruction style on both F1 and EM, with persona slightly ahead (F1 0.497 vs 0.489; EM 0.425 vs 0.415) and a moderate abstention rate that tempered hallucinations. Step-4 widened the design space (embedding sizes 256/384/512; top-k=3/5/10). The strongest, stable trade-off was 384-D with k=5.

The Step-5 enhancement introduced a CrossEncoder re-ranker and a simple confidence signal (top-score and margin) with optional abstention. While this improved context recall, our RAGAs evaluation (Step-6, gpt-4o-mini judge) indicated drops in faithfulness and answer relevancy for the enhanced run, likely driven by over-inclusive contexts, re-ranker calibration, and judge timeouts on longer prompts. The system is deployment-ready for a small corpus with clear observability and evaluation harnesses; to raise trust and accuracy at scale I recommend a re-ranker sanity check (span-match), tighter context budgets, and stabilized LLM-judge infrastructure for automated QA.

The codebase separates concerns to keep iteration auditable. utils.py provides: deterministic seeding; a minimal logger; data loaders for passages/queries (CSV to avoid network variability); Sentence-Transformers encoders with normalization so IP≈cosine; and a tiny MilvusIndex wrapper over Milvus Lite that owns the schema (id, passage, embedding) and ANN configuration (IVF\_FLAT for local mode). This wrapper avoids ORM foot-guns, exposes only add() and search(), and prints collection stats for sanity.

The naive orchestrator, naive\_rag.py, is intentionally linear: load→embed→index once; embed queries→ANN retrieve (top-k)→prompt→generate; then persist a compact JSON per query: {qid, question, prediction, gold, context\_ids}. The default prompt is short and defensive: “answer only from context; otherwise say you don’t know; cite passage IDs.” Generation uses FLAN-T5-base at temperature 0 for determinism.

The enhanced orchestrator, enhanced\_rag.py, keeps the same outer contract and adds two layers. First, it retrieves a larger candidate pool (e.g., 50) via ANN and re-ranks pairs (query, passage) using a CrossEncoder (cross-encoder/ms-marco-MiniLM-L-6-v2). Only the top-k re-ranked contexts are kept for prompting, increasing precision at the top of the window. Second, it computes confidence from the re-rank scores (top score and top-2 margin). With --abstain\_on\_low\_conf, low-confidence items yield “I don’t know.” rather than weakly supported answers. Both signals are written to results for later analysis.

Evaluation is handled in two layers. evaluation.py computes per-example F1 (SQuAD-style) plus macro EM/F1 via Hugging Face evaluate, and appends runs to comparison\_analysis.csv. ragas\_evaluation.py converts any run into a RAGAs dataset and computes faithfulness, context precision/recall, and answer relevancy with a configurable LLM judge; to survive judge instability it shards the dataset, trims contexts, retries, and aggregates weighted means to ragas\_summary.csv.

Persona and CoT beat the plain instruction prompt. Instruction’s F1=0.443 (95% CI [0.378, 0.508]) and EM=0.370 came with the highest “unknown” rate (15.5%), which reflects a conservative but over-strict reading of “answer only from context.” CoT reduced abstentions to 2% and lifted both F1 to 0.489 ([0.424, 0.554]) and EM to 0.415; its planning nudges the model to “fill connective tissue” when the span is implicit. Persona yielded the best F1=0.497 ([0.432, 0.563]) and EM=0.425 with a moderate 5.5% unknown rate—evidence that a domain-appropriate, concise, citation-oriented voice aligns well with retrieval-bounded QA. The F1 confidence intervals for persona and CoT overlap, so the advantage is small but consistent. The shared ceiling (F1≈0.5) points to a recall bottleneck: when the single passage lacks the answer, prompt style alone cannot rescue accuracy.

Across {256, 384, 512} dimensions, 384-D is the best stability/cost point. 256-D suffered modest F1/EM drops consistent with weaker neighborhood geometry after compression; 512-D rarely exceeded 384-D and added latency. Increasing top-k improved recall up to k≈5; beyond that, redundancy and generator distraction counterbalanced additional evidence. Diversifying the context set (MMR) consistently helped vs naive top-k at the same budget—less topical duplication increased the odds of including at least one extractable span. These trends matched abstention behavior: instruction abstained less as k grew (more chances to find a span), but persona preserved the best precision/recall trade-off.

The HF SQuAD macro EM/F1 tracked our per-example F1 means closely, indicating that generation noise is not the dominant error driver at this scale; retrieval coverage is. That motivated the Step-5 re-ranking enhancement to raise the quality of top-k contexts without growing the prompt excessively. The strongest naive regime couples 384-D embeddings with k=5 and persona prompting. The mechanism is straightforward: 384-D preserves enough local structure for ANN, and five diverse contexts are usually sufficient to include an answer span without overwhelming the generator. Persona’s disciplined style converts that evidence into short, auditable answers with acceptable abstention.

I introduced two features: 1) a robust CrossEncoder re-ranker, and 2) a simple confidence-based abstention. The re-ranker directly optimizes query-passage relevance at inference time, correcting ANN’s geometric approximations; it is especially effective at small k (3–5), where one mislabeled passage can dominate a generation. The confidence signals come for “free” from re-rank scores. Using both the top score and the margin makes the abstention rule interpretable: I only answer when the best candidate is individually strong and clearly better than the runner-up.

Mechanistically, this should increase context precision (fewer off-topic contexts in the top-k), maintain or raise context recall (I still retrieve a broad candidate pool), and improve faithfulness (better evidence → tighter answers). However, our RAGAs results for the enhanced run (top-5, persona, rerank) showed higher context recall (0.054) but collapsed faithfulness and context precision (both 0.00) with lower answer relevancy (0.288) compared to naive top-1. Three interacting factors likely explain the disparity:

1. Context budget and trimming. I capped at five contexts and 1,800 chars each to keep judge tokens in check. If the re-ranker surfaces long, semantically relevant but non-extractive passages, the trimmed windows may lack explicit spans, causing the judge to mark answers as unfaithful—even when the answer is actually grounded in the untrimmed text.
2. Re-ranker calibration. The MiniLM MS-MARCO model works well on passage retrieval benchmarks, but on encyclopedic snippets its score scale can be flat. A flat distribution reduces the top-2 margin and destabilizes our confidence rules; it also allows near-ties to propagate into top-k, reintroducing redundancy.
3. Judge instability. I observed frequent timeouts on longer prompts; shards with long contexts fail more often and are excluded, biasing means. Surviving enhanced shards may over-represent harder cases. This is consistent with the drop in answer relevancy despite better recall.

Two simple mitigations already evident from the analysis are: cut context to k=3 with MMR diversity before re-ranking (less redundancy, tighter spans), and add a span-match sanity check after re-ranking (discard candidates that do not contain a lexical/semantic match for the predicted answer). Both should improve faithfulness without growing compute.

Milvus Lite is perfect for this assignment and small deployments; a server Milvus with HNSW/IVF-PQ is advised once collections or QPS grow. The indexing API is intentionally thin, so migration is a configuration change. Re-ranking adds linear cost in candidates; cap retrieve\_candidates to 50 and prefer GPU inference (or async CPU pools) if latency matters.

Persist the JSON outputs with context\_ids, conf, and margin as I do now; these support audit trails and online shadow evaluations. Add simple health metrics—ANN latency, re-rank throughput, abstention rate, and “empty context” rate—to catch drifts.

Keep persona as the default prompt and enforce the “use only context” guard. Confidence-based abstention is a practical safety valve; tune thresholds on a validation slice for your domain. For external judge-based QA (RAGAs), isolate credentials, implement exponential-backoff with jitter, and schedule overnight runs to avoid peak rate limits.

All runs used fix seeds, log parameters, and write rows to comparison\_analysis.csv and ragas\_summary.csv. To rerun: 1) build the index with naive\_rag.py or enhanced\_rag.py CLI flags; 2) evaluate with evaluation.py (macro EM/F1 and per-example F1); 3) run ragas\_evaluation.eval\_one() with the same environment knobs (batch, context caps, judge model). The repository includes template commands and environment variables.

Limitations. The current confidence is heuristic and not a calibrated probability; thresholds can trade EM against abstention in non-linear ways. Judge-based metrics are sensitive to token budgets and API stability; CI on F1 provides complementary signal that is less brittle. Finally, chunking is basic; sentence-aware splitting plus overlap would likely raise recall and faithfulness simultaneously.

**Appendix**

**A1. OpenAI API Errors**

A screenshot of a computer program

AI-generated content may be incorrect.