QAR-Diff: Atomic Semantic Edit Discrimination via Neuro-Symbolic Question–Answer Alignment

Anonymous ACL submission

Abstract

Maintaining semantic consistency in Retrieval-Augmented Generation (RAG) systems is critical, as source document updates can invalidate downstream behavior. Existing chunk- or summary-level approaches suffer from two unresolved failure modes: LLM *fluency bias*, which omits critical details such as numerical limits, and neural *symbolic blindness*, which assigns near-identical representations to statements differing by a single numeral (e.g., "500 km" vs. "550 km"). Critically, chunk-level equivalence does not imply atomic factual equivalence, since a single paragraph may mix preserved, modified, and new facts.

We propose **QAR-Diff**, which reformulates semantic document revision as a *set alignment problem over Atomic Question–Answer Representations (QARs)*. To improve factual coverage, *Symbolic-Aware Residual Distillation* iteratively surfaces atomic facts missed by single-pass generation. To improve factual precision, *Multi-View Neuro-Symbolic Alignment* combines a *Symbolic Consistency Gate* with a *Retrieve-then-Verify Safety Net* to verify unmatched candidates against source text, decoupling semantic equivalence from surface variation.

Experiments on *Syn-Edits* (97.2% expert-verified), DiffQG, and industrial manuals show that strong baselines capture under half of the underlying atomic changes despite competitive chunk-level accuracy (ACU Coverage: 0.431 vs. **0.791**), establishing fine-grained QAR alignment as essential for reliable, human-verifiable RAG maintenance.

1 Introduction

As Large Language Model (LLM)-powered Retrieval-Augmented Generation (RAG) systems become integral to high-stakes domains such as financial compliance, healthcare, and industrial safety, maintaining consistency between evolving source documents and deployed knowledge systems has emerged as a critical reliability challenge (Lewis et al., 2020; Rim et al., 2025; Chun et al., 2025; Huang et al., 2025). Even seemingly minor document revisions can invalidate downstream responses, making reliable semantic document revision analysis a prerequisite for trustworthy document-grounded NLP systems.

At the core of this problem lies **Semantic Edit Discrimination**: the ability to distinguish *Semantic-Altering Edits* from *Semantic-Preserving Edits*. If subtle factual changes are missed because surrounding text remains highly similar, the system suffers from *silent factual drift*; conversely, treating every structural rewrite as a new fact bloats the retrieval index with redundancy.

Existing approaches fail for complementary reasons. Lexical methods such as `diff` are brittle to structural variation. Embedding-based similarity and NLI pipelines are more robust to paraphrase, but they remain *symbolically blind*: statements differing only in numerals, units, or named entities often receive high similarity scores despite containing critical factual discrepancies (Naik et al., 2018; Wallace et al., 2019; Ravichander et al., 2019). Prior QA-based approaches show promise for factual representation (Wang et al., 2020; Cole et al., 2023; Trienes et al., 2024), but do not address how to align *independently generated* atomic units across structurally different document versions.

More fundamentally, current workflows remain limited by two root-cause problems:

- **P1: Coverage Exhaustiveness (the fluency bias).** Standard single-pass LLM generation prioritizes dominant themes while systematically overlooking peripheral yet critical details such as numerical limits, measurement units, and entity-specific constraints (Maynez et al., 2020).
- **P2: Symbolic Blindness in Neural Align-**

ment. Neural similarity models are structurally ill-equipped for exact symbolic comparison, so even a single numerical change can be semantically decisive while remaining nearly invisible to embeddings or NLI scores (Wallace et al., 2019; Naik et al., 2018).

A key consequence is that *chunk-level equivalence does not imply factual equivalence*: a single paragraph may simultaneously contain preserved, modified, and newly introduced facts, so coarse-grained methods may detect that "something changed" while still failing to identify *what* changed at the atomic level.

To address these limitations, we propose **QAR-Diff**, a framework that reformulates semantic document revision as a *set alignment problem over Atomic Question–Answer Representations (QARs)*. Each document version is decomposed into minimal QA pairs expressing self-contained facts, and revision analysis proceeds by aligning these atomic units globally rather than comparing chunks locally. This decouples semantic equivalence from surface form and enables revision analysis at the level of verifiable atomic facts, as illustrated in Figure 1.

Our contributions are summarized as follows:

- We formalize semantic document revision as a set alignment problem over atomic QARs, providing a principled alternative to sentence-, chunk-, and summary-level comparison.
- We propose **Symbolic-Aware Residual Distillation** (addressing P1), an iterative extraction framework that surfaces peripheral yet semantically critical symbolic facts missed by single-pass generation.
- We introduce **Multi-View Neuro-Symbolic Alignment** with a hard *Symbolic Consistency Gate* and a **Retrieve-then-Verify Safety Net** (addressing P2), combining neural flexibility with strict symbolic verification.
- We introduce *Syn-Edits* and demonstrate that QAR-Diff substantially improves atomic-level factual fidelity on both synthetic and industrial document revisions.

2 Related Work

2.1 Document Difference and Revision Analysis

Document difference analysis has traditionally focused on structural or lexical comparison. Early systems such as `diff` and XML-aware methods (e.g., X-Diff (Wang et al., 2003)) rely on surface-level alignment, making them brittle to reformatting and sentence reordering. Subsequent work introduced topic modeling and distributional similarity to compare document versions at a macro level (Greene and Cross, 2011), but these approaches remain insensitive to fine-grained factual changes.

More recent studies leverage annotated revision corpora such as WikiAtomicEdits (Faruqui et al., 2018) and NewsEdits (Spangher and May, 2021), enabling the classification of edit intents (e.g., fluency vs. factual updates) (Yang et al., 2017; Mita et al., 2022). While useful for edit *categorization*, these methods typically assume pre-aligned sentence pairs and operate as local classifiers. Such assumptions rarely hold in real-world technical documents, where semantically related content may be merged, split, or spread across disjoint areas in subsequent revisions. Moreover, *chunk-level equivalence does not imply atomic factual equivalence*: a single revised paragraph may simultaneously contain preserved, modified, and newly introduced facts.

In contrast, QAR-Diff treats document revision as a *global set alignment problem* over independently extracted atomic units. By decomposing documents into atomic QARs and aligning them across versions without assuming sentence-level correspondence, our approach handles substantial structural divergence and cross-paragraph factual shifts.

2.2 QA-Based Semantic Representation

Question–answer representations have emerged as a powerful tool for interpretable and fine-grained semantic analysis. DiffQG (Cole et al., 2023) introduced the task of generating questions that explicitly capture differences between document versions, showing that QA pairs can effectively surface semantic edits. Similarly, InfoLossQA (Trienes et al., 2024) employs QA pairs to quantify information loss in document summarization, treating them as both atomic factual probes and evaluation signals.

Despite their success, these approaches do not address the central problem targeted here: aligning QA units generated *independently* from separate document versions. DiffQG assumes joint availability of both document versions during question generation, whereas InfoLossQA uses QA pairs purely as a post-hoc evaluation signal.

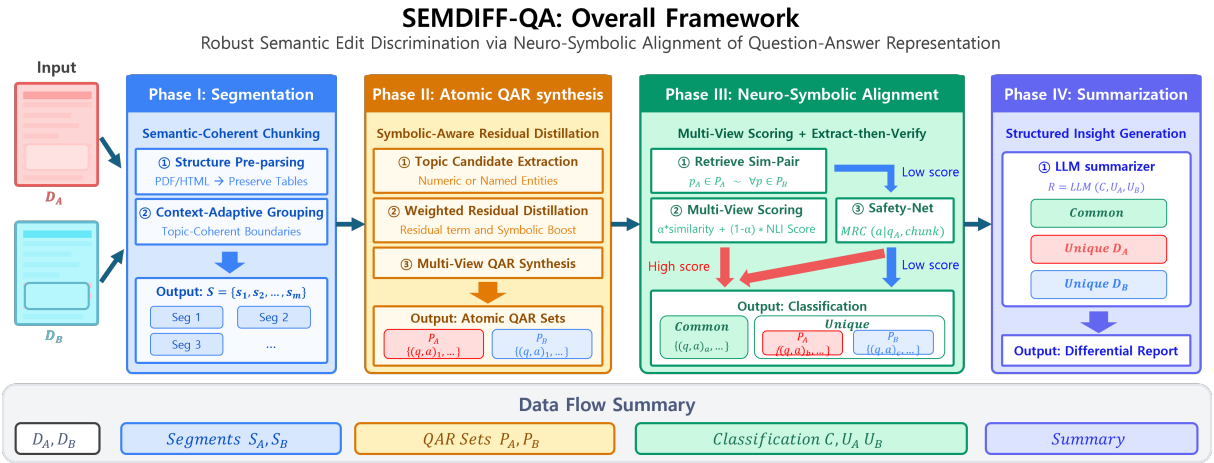

Figure 1: Overview of the **QAR-Diff** framework. The pipeline consists of four phases: (I) Structure-Aware Semantic Segmentation, (II) Atomic QAR Synthesis via Symbolic-Aware Residual Distillation, (III) Multi-View Neuro-Symbolic Alignment, and (IV) Structured Revision Synthesis. This architecture enables precise discrimination between semantic-preserving and semantic-altering edits by explicitly decoupling semantic equivalence from surface realization.

QAR-Diff addresses this gap by elevating QA pairs from auxiliary probes to *first-class atomic semantic units* within a unified alignment framework. Because QARs are extracted independently for each document version, QAR-Diff introduces *Symbolic-Aware Residual Distillation* to improve factual coverage before alignment.

2.3 Neuro-Symbolic Approaches for Factual Precision

Purely neural semantic models, including embedding similarity and NLI-based systems, are known to struggle with symbolic reasoning. They often rely on syntactic heuristics rather than logical entailment (McCoy et al., 2019) and exhibit *symbolic blindness* when comparing numerical values (Wallace et al., 2019; Naik et al., 2018): statements differing by a single numeral (e.g., "100 psi" vs. "120 psi") frequently receive high entailment scores despite carrying factually critical differences. This limitation has motivated QA-based factuality metrics such as QAGS (Wang et al., 2020), which assess consistency by verifying generated answers against source documents.

However, prior work typically applies symbolic reasoning as a *post-hoc evaluation* signal rather than integrating it into the core alignment decision. In document comparison settings, this separation allows high-confidence yet factually incorrect alignments to pass unchallenged.

QAR-Diff instead embeds symbolic constraints directly into the alignment stage. Our *Symbolic Consistency Gate* enforces exact agreement on numerical values and named entities, while neural models handle paraphrasing and lexical variation. Combined with the *Retrieve-then-Verify Safety Net*, this design reduces both symbol-insensitive matches and extraction-induced false positives, making the framework suitable for high-stakes settings such as compliance auditing and self-correcting RAG systems.

3 Methodology

We formalize *Semantic Edit Discrimination* as a set alignment problem over atomic semantic units, identify the key approximation challenges that make this problem non-trivial, and introduce our four-phase framework as a principled solution to each challenge.

3.1 Problem Formulation

We recast document comparison not as a sequence alignment task, but as a *global set operation over Atomic Semantic Units*. Formally, a document D is mapped to a set of Question–Answer Representations (QARs):

$$\mathcal{P}(D) = \{(q_i, a_i)\}_{i=1}^{N}, \tag{1}$$

where each pair (q_i, a_i) encapsulates a minimal, self-contained factual claim. In our implementation, each QAR also includes multiple question views (Section 3.3); we use (q_i, a_i) here for notational simplicity. Given two document versions

$D^{(1)}$ and $D^{(2)}$, we compute an explicit matching set across the entire document scope:

$$\mathcal{M} = \{(p_A, p_B) \mid p_A \in \mathcal{P}^{(1)},\ p_B \in \mathcal{P}^{(2)},\ p_A \equiv p_B\} \tag{2}$$

where $\equiv$ denotes semantic equivalence that is invariant to stylistic variation yet strict over symbolic content. The **Common** and **Unique** subsets are then derived from $\mathcal{M}$ directly:

$$\mathcal{C}^{(k)} = \{p \in \mathcal{P}^{(k)} \mid \exists p' \in \mathcal{P}^{(\bar{k})} : (p, p') \in \mathcal{M}\}, \tag{3}$$

$$\mathcal{U}^{(k)} = \mathcal{P}^{(k)} \setminus \mathcal{C}^{(k)}. \tag{4}$$

This derivation yields a formally valid partition ($\mathcal{P}^{(k)} = \mathcal{C}^{(k)} \cup \mathcal{U}^{(k)}$, $\mathcal{C}^{(k)} \cap \mathcal{U}^{(k)} = \emptyset$) of each QAR set. Units in $\mathcal{C}^{(k)}$ correspond to *Semantic-Preserving Edits* (matched QARs with potentially different surface realizations), while units in $\mathcal{U}^{(k)}$ constitute *Semantic-Altering Edits* reflecting factual additions, deletions, or symbolic modifications.

Approximation Challenges. Computing $\equiv$ precisely and achieving exhaustive extraction are jointly non-trivial due to two core challenges that directly motivate our pipeline:

- **P1 (Coverage Exhaustiveness):** Achieving exhaustive extraction of $\mathcal{P}(D)$ is difficult because standard LLM generation exhibits a fluency bias, systematically overlooking peripheral yet critical details such as numerical values and named entities (Maynez et al., 2020).
- **P2 (Symbolic Blindness):** Approximating $\equiv$ with neural similarity is unreliable for symbolic content, as small changes in numbers or entities produce near-identical embedding representations despite inducing large semantic shifts (Wallace et al., 2019; Naik et al., 2018).

In addition, equivalent facts may be expressed using highly divergent surface forms (Ganitkevitch et al., 2013) or relocated across distant document sections, making local comparison methods insufficient for global alignment.

3.2 Phase I: Semantic-Coherent Segmentation

Before extracting QARs, document D must be partitioned into a sequence of discrete, semantically coherent segments $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$. Standard retrieval pipelines employ fixed-length sliding windows, but this rigid approach frequently fragments logical boundaries—splitting tables, lists, or cross-sentence arguments—which degrades extraction quality (Gao et al., 2023). We instead apply a **Structure-Aware Semantic Segmentation** strategy in two steps.

Step 1: Structural Normalization. Document layout analysis identifies structural elements such as tables and nested lists. These are serialized into structure-preserving Markdown, ensuring that 2D topological relationships (e.g., header–cell and row–column correspondences) are maintained as explicit 1D textual context rather than discarded as unstructured noise. This step is critical for preserving structured facts—such as single-cell updates in specification tables—that are frequently missed by unstructured text extractors.

Step 2: LLM-Guided Semantic Chunking. Inspired by proposition-level retrieval (Chen et al., 2024), an LLM is prompted to identify logical topic boundaries in the normalized text and insert explicit split markers between complete, continuous meaning units. This semantic boundary detection avoids mechanical splitting and ensures each resulting segment s_k is *contextually self-contained*, providing a coherent basis for downstream QAR synthesis.

3.3 Phase II: Atomic QAR Synthesis via Residual Distillation

Phase II directly addresses **P1** by synthesizing an exhaustive set of atomic QARs from each segment. Single-pass LLM generation tends to prioritize salient themes while omitting peripheral yet critical details (Maynez et al., 2020; Huang et al., 2025). Keyword-based extraction, while exhaustive in coverage, lacks semantic grounding and provides weak coverage guarantees (Grootendorst, 2020; Trienes et al., 2024). We propose **Symbolic-Aware Residual Distillation** to overcome this trade-off through iterative, coverage-driven generation.

Candidate Topic Construction. For each segment s, we construct a candidate topic set:

$$\mathcal{T}(s) = \{\tau_1, \tau_2, \ldots, \tau_M\}, \tag{5}$$

deterministically extracted via a hybrid NLP pipeline that combines linguistic signals (noun phrases and named entities) with rule-based pattern matching for numerical expressions, measurement units, and structure-derived header–value

cues. This deterministic extraction improves recall of peripheral symbolic facts while mitigating hallucination at the topic-discovery stage. The purpose of $\mathcal{T}(s)$ is not to replace semantic generation with keywords, but to provide controllable anchors that guide exhaustive QAR extraction.

Topic-Guided Iterative Synthesis via MMR. We maintain a *residual embedding* $\mathbf{r}_t$ (initialized as the segment embedding $\mathrm{Enc}(s)$) representing semantic content not yet covered by previously generated QARs. At each iteration t, we select the topic τ^* that maximizes residual relevance while prioritizing symbolic content:

$$\tau^* = \underset{\tau \in \mathcal{T}(s)}{\operatorname{argmax}} \Big[\alpha \cos(\mathbf{v}_\tau, \mathbf{r}_t) + (1-\alpha)\, \mathbb{I}_{sym}(\tau) \Big], \tag{6}$$

where $\mathbf{v}_\tau$ is the topic embedding and $\mathbb{I}_{sym}(\tau) \in \{0, 1\}$ indicates whether τ contains symbolic entities (numbers, units, named entities). This objective adapts Maximal Marginal Relevance (Carbonell and Goldstein, 1998): using the residual state $\mathbf{r}_t$ as the relevance target implicitly penalizes redundancy with previously extracted content, while the symbolic boost ensures that the details most critical for change detection receive priority.

Multi-View QAR Generation. Conditioned on τ^*, an LLM generates a **Multi-View QAR** $p_t = (q^{std}, q^{nat}, a)$, where q^{std} is a *Standard View* (canonical phrasing aligned with the document's formal register) and q^{nat} is a *Natural View* (conversational paraphrase). This dual representation directly helps bridge the lexical gap identified in Section 3.1, ensuring robustness against stylistic variation in downstream NLI matching.

Residual Update. After generating p_t, we remove its explained semantic component from the residual via orthogonal projection:

$$\mathbf{r}_{t+1} \leftarrow \mathbf{r}_t - \frac{\mathbf{r}_t \cdot \mathbf{v}_{p_t}}{\|\mathbf{v}_{p_t}\|^2}\, \mathbf{v}_{p_t}, \tag{7}$$

ensuring subsequent iterations focus exclusively on semantically uncovered content. In implementation, this update is combined with a dampening mechanism to preserve stability while discouraging redundant generation. The process terminates when $\|\mathbf{r}_t\| \leq \epsilon$, the topic pool is exhausted, or $t \geq T_{max}$. Aggregating over all segments yields the document-level QAR set:

$$\mathcal{P}(D) = \bigcup_{s \in \mathcal{S}} \mathcal{P}(s). \tag{8}$$

Algorithm 1: Symbolic-Aware Residual Distillation

Input: Segment s, candidate topic set $\mathcal{T}(s)$
Output: Atomic QAR set $\mathcal{P}(s)$

Initialize $\mathbf{r}_0 \leftarrow \mathrm{Enc}(s)$,;
$\mathcal{P}(s) \leftarrow \emptyset$;
while $\|\mathbf{r}_t\| > \epsilon$ ***and*** $\mathcal{T}(s) \neq \emptyset$ ***and*** $t < T_{max}$ **do**
  // Select topic via hybrid MMR objective (Eq. 6)
  $\tau^* \leftarrow \operatorname{argmax}_{\tau \in \mathcal{T}(s)} \big[\alpha\, \mathrm{Sim}(\mathbf{v}_\tau, \mathbf{r}_t) + (1-\alpha)\, \mathbb{I}_{sym}(\tau)\big]$;
  // Generate Multi-View QAR conditioned on τ^*
  $p_t \leftarrow \mathrm{LLM_Gen}(s,\ \tau^*)$;
  $\mathcal{P}(s) \leftarrow \mathcal{P}(s) \cup \{p_t\}$;
  // Update residual via orthogonal projection (Eq. 7)
  $\mathbf{r}_{t+1} \leftarrow \mathbf{r}_t - \mathrm{Proj}_{\mathbf{v}_{p_t}}(\mathbf{r}_t)$;
  $\mathcal{T}(s) \leftarrow \mathcal{T}(s) \setminus \{\tau^*\}$;
end

The complete procedure is summarized in Algorithm 1; hyperparameter settings are provided in Appendix B.

3.4 Phase III: Multi-View Neuro-Symbolic Alignment

Phase III directly addresses **P2** by estimating the matching set $\mathcal{M}$ (Eq. 2) between $\mathcal{P}_A$ and $\mathcal{P}_B$. By aligning set-to-set rather than chunk-to-chunk, this phase naturally handles facts that have been structurally relocated across distant document sections. We employ a cascaded alignment strategy consisting of four stages—embedding retrieval, lexical rescue, NLI verification, and symbolic consistency checking—that combines high-recall neural matching with strict symbolic verification, passing units that remain unmatched to a **Retrieve-then-Verify Safety Net** before the final *Unique* classification.

High-recall candidate retrieval. For each query unit $p_A \in \mathcal{P}_A$, we retrieve top-k candidates from $\mathcal{P}_B$ using the multi-view representations (q^{std}, q^{nat}) to maximize recall against lexical variation. This stage is designed to preserve plausible paraphrastic matches before more selective verification is applied.

Neuro-Symbolic Pairwise Scoring. At a conceptual level, the alignment score for each candidate pair is:

$$S_{\text{align}}(p_A, p_B) = \Big[\alpha\,\text{Sim}(p_A, p_B) + (1-\alpha)\,P_{\text{NLI}}(p_A \models p_B)\Big] \times \mathbb{G}_{sym}(a_A, a_B), \tag{9}$$

where $\mathbb{G}_{sym}(a_A, a_B) \in \{0, 1\}$ is the **Symbolic Consistency Gate**: it returns 1 only when the numerical values and named entities in a_A and a_B are symbolically consistent, and 0 otherwise. This gate prevents otherwise high-scoring neural matches from being accepted when they contain decisive symbolic conflicts—a failure mode well-documented in standard NLI systems (McCoy et al., 2019). To optimize computational efficiency at scale, this scoring is realized as a cascaded filtering pipeline rather than a single monolithic ranker. High-similarity candidates are first collected through embedding retrieval; borderline cases are recovered through fuzzy lexical matching; NLI is then applied to validate semantic compatibility; and finally the symbolic gate checks numerical and entity-level consistency. This cascade reduces computational cost while preserving symbolic precision as the final arbiter.

Retrieve-then-Verify Safety Net. A unit p_A may remain unmatched not because it represents a true semantic addition or deletion, but because its corresponding QAR was not generated during Phase II. Classifying such units as *Unique* would produce spurious false positives. To prevent this, unmatched units are subjected to a **Retrieve-then-Verify Safety Net** before the final decision.

We broaden the search to raw text chunks of $D^{(2)}$ using semantic retrieval over chunk representations, supplemented by lexical rescue signals from the earlier matching stage. A cross-encoder MRC model then extracts a candidate answer a'_{pred} from the retrieved contexts $\mathcal{K}$. This answer is verified against the original question using NLI, gated by the same symbolic consistency check:

$$S_{\text{safe}}(p_A, \mathcal{K}) = \max_{c \in \mathcal{K}} P_{\text{NLI}}(c \models p_A) \cdot \mathbb{G}_{sym}(a_A, a'_{pred}). \tag{10}$$

A unit is classified as *Unique* only if both direct QAR alignment and safety-net verification fail. This conservative two-stage criterion separates true semantic changes from extraction misses, ensuring that the final sets $\mathcal{U}^{(1)}$ and $\mathcal{U}^{(2)}$ reflect genuine factual differences. Model architectures and decision thresholds are detailed in Appendix A and Appendix B.

3.5 Phase IV: Structured Insight Synthesis

Phase IV translates the algorithmically precise outputs of Phase III into a human-verifiable revision report. Using an LLM synthesizer, we aggregate $\mathcal{C}$, $\mathcal{U}^{(1)}$, and $\mathcal{U}^{(2)}$ and organize detected changes into two interpretable categories:

1. **Value Updates:** Semantically corresponding units whose symbolic values differ (e.g., "Maximum driving distance: 500 km" → "550 km"), representing in-place factual revisions.
2. **Information Gaps:** Units existing exclusively in one document version with no matched counterpart, representing true factual additions or deletions.

The final output is a structured natural-language report that abstracts away low-level QAR details while preserving the atomic facts necessary for human verification. This allows domain experts and RAG maintainers to audit semantic shifts and perform surgical index updates without inspecting raw document text, directly supporting the human-verifiable maintenance workflow motivating this work.

4 Experiments

We evaluate **QAR-Diff** to answer the following research questions:

- **(RQ1) Semantic Edit Discrimination:** Can atomic QAR alignment discriminate semantic edits more faithfully than sentence- or summary-level baselines?
- **(RQ2) Generation Efficacy:** Does Symbolic-Aware Residual Distillation improve factual coverage during QAR extraction?
- **(RQ3) Scalability:** Does QAR-Diff remain robust on large-scale industrial documentation?

4.1 Experimental Setup

Datasets. We evaluate on three datasets. **(1) SynEdits (Ours)** is a controlled benchmark spanning six domains (Academic, Essay, Medical, Patent,

Table 1: Human evaluation of Syn-Edits label quality. Overall 97.2% label consistency (105/108 pairs). Underlined scores indicate rare cases where semantic-preserving edits unintentionally introduced minor factual shifts, typically due to complex structural reordering (T1).

Domain	T1	T2	T5	T3	T4	T6
Academic	2	3	3	3	3	3
Essay	3	3	3	3	3	3
Medical	3	3	3	3	3	3
Patent	2	3	3	3	3	3
Manual	3	3	3	3	3	3
Wiki	3	3	2	3	3	3

Manual, Wiki), comprising **180 document pairs** generated by applying six transformation types (T1–T6) to 30 seed documents (5 per domain). Seed documents were collected from publicly available, domain-representative corpora (e.g., arXiv, PubMed, Google Patents, Wikipedia), deduplicated, and filtered by human annotators for suitable length and content complexity. **(2) DiffQG** (Cole et al., 2023) is used to evaluate generation recall on **500 sampled document pairs**. **(3) Industrial Manuals** consists of three pairs of vehicle owner's manuals (400–700 pages each) with dense structured content.

Human Verification of Syn-Edits. To validate label reliability, domain experts evaluated a stratified random subset of **108 pairs** (3 per domain-type configuration), yielding a **97.2% label consistency rate** (105/108), as shown in Table 1. Disagreements were rare and mostly arose from structural reordering cases (T1) that introduced unintended minor semantic shifts. Full generation protocols and prompt templates are provided in Appendix E.

Baselines. We compare against two document-comparison baselines and two generation ablations. **Sentence-NLI** is an NLI-based sentence-pair comparison method, where sentences are matched by embedding retrieval and classified as common or unique. **Summary-Embedding** is a chunk-level embedding baseline, where documents are split into fixed chunks, summarized, and compared via embedding similarity. For RQ2, we additionally compare against **LLM-Direct** (single-pass QAR generation) and **KeyBERT-Guided** (keyword-driven generation). Full model specifications are given in Appendix A.

Evaluation Protocol. We adopt a G-Eval-style LLM-as-a-Judge protocol (Liu et al., 2023a) using GPT-4.1. A **Common** prediction is correct only if its factual claim is verifiable from *both* versions, while a **Unique** prediction must be verifiable from *exactly one*. To assess judge reliability, we manually verified a random subset of 100 judgments and observed **89%** human–judge agreement. We therefore use the judge as a scalable verification layer rather than as a substitute for human annotation.

Table 2: **Semantic Edit Discrimination on Syn-Edits.** QAR-Diff achieves the best Common Accuracy. Summary-Embedding's high Unique Accuracy is not matched by faithful atomic-level revision tracking (see Table 3).

Method	Common Acc.	Unique Acc.
Sentence-NLI	0.907	0.772
Summary-Embedding	0.857	**0.932**
QAR-Diff (Ours)	**0.941**	0.834

Table 3: **Fine-Grained Information Recall (ACU Coverage).** High Unique Accuracy does not imply high factual fidelity. Summary-Embedding captures less than half of the actual atomic changes.

Method	ACU Cov.	T3 Rec.	T4 Rec.
Sentence-NLI	0.721	0.855	0.692
Summary-Embedding	0.431	0.490	0.338
QAR-Diff (Ours)	**0.791**	**0.889**	**0.779**

Metrics. We report **Common Accuracy** and **Unique Accuracy**. To assess fine-grained factual fidelity, we additionally report **ACU Coverage** (Liu et al., 2023b), defined as recall of ground-truth atomic factual changes within the detected unique set. For DiffQG, we also report **ROUGE-L** (Lin, 2004), **BLEURT** (Sellam et al., 2020), **QSIM** (Risch et al., 2021), and **Recall**.

4.2 Semantic Edit Discrimination (RQ1)

Table 2 shows that QAR-Diff achieves the highest Common Accuracy (0.941), indicating strong robustness to semantic-preserving rewrites such as paraphrasing and structural reorganization. The more important finding is in Table 3: although Summary-Embedding attains the highest Unique Accuracy (0.932), its ACU Coverage is only **0.431**, meaning that it often detects that "something changed" without accurately recovering *which facts* changed.

In contrast, QAR-Diff achieves **0.791 ACU Coverage**, substantially outperforming both baselines in fine-grained factual recovery. This supports our

Table 4: **Generation Recall on DiffQG.** Symbolic-Aware Residual Distillation achieves the highest recall (0.906), substantially outperforming both single-pass and keyword-guided generation. Lower surface-form scores are expected: multi-view generation prioritizes factual coverage over reference-mimicking.

Method	ROUGE-L	BLEURT	QSIM	Recall
Reference (DiffQG)	**0.535**	**0.565**	**0.506**	0.848
LLM-Direct	0.227	0.269	0.158	0.480
KeyBERT-Guided	0.309	0.381	0.192	0.772
QAR-Diff (Distill)	0.371	0.473	0.261	**0.906**

central claim that chunk-level accuracy alone is insufficient for reliable revision analysis. In revision-heavy settings, edits are typically sparse relative to large invariant sections (Ihori et al., 2022; Mita et al., 2022); coarse baselines over-flag these regions, whereas QAR-Diff isolates atomic factual updates more precisely.

QAR-Diff's Unique Accuracy (0.834) remains below Summary-Embedding's (0.932). Error analysis shows that the main failure mode is extraction on highly abbreviated structured content, where a single-cell update in a dense specification table constitutes a complete semantic-altering edit. In such cases, Phase II residual distillation may fail to anchor a topic candidate on the updated cell value, producing a missed QAR that the Safety Net cannot recover.

4.3 Ablation: Generation Efficacy (RQ2)

Table 4 validates the contribution of Symbolic-Aware Residual Distillation in overcoming the LLM fluency bias (P1) on the DiffQG generation task. QAR-Diff achieves the highest Recall (0.906), substantially outperforming both LLM-Direct (0.480) and KeyBERT-Guided (0.772).

This evaluation is structurally conservative for QAR-Diff: DiffQG reference questions are generated jointly from both document versions, whereas QAR-Diff generates independently from each document with no cross-version visibility. The resulting trade-off—lower ROUGE-L, BLEURT, and QSIM alongside higher recall—is expected and by design: multi-view representations (q^{std}, q^{nat}) prioritize factual coverage over reference-mimicking.

4.4 Industrial Scalability (RQ3)

Table 5 presents results on large-scale vehicle owner's manuals. Summary-Embedding degrades substantially (Common: 0.793, Unique: 0.612) under structural noise and document length,

Table 5: **Scalability on Industrial Manuals.** QAR-Diff maintains robust accuracy on 400–700-page vehicle manuals, while Summary-Embedding degrades substantially. Sentence-NLI is excluded due to prohibitive runtime on this setting; see Appendix C.

Method	Common Acc.	Unique Acc.
Summary-Embedding	0.793	0.612
QAR-Diff (Ours)	**0.887**	**0.724**

whereas QAR-Diff maintains robust accuracy (Common: 0.887, Unique: 0.724), demonstrating that structure-aware segmentation (Phase I) effectively preserves table and list semantics and that set-level QAR alignment scales to lengthy, structurally complex documents without the brittleness of chunk-level methods.

Furthermore, the cascaded alignment strategy (Phase III) improves practical scalability by avoiding exhaustive all-pairs QAR comparison. By filtering candidates through embedding retrieval and fuzzy matching before expensive NLI verification, QAR-Diff bounds verification to a small candidate set per query rather than the full Cartesian product of cross-version QAR pairs (details in Appendix C). This makes atomic-level semantic tracking tractable for large-scale RAG maintenance.

5 Conclusion

We presented **QAR-Diff**, a framework that reformulates semantic document revision as alignment over Atomic Question–Answer Representations (QARs), enabling fine-grained discrimination between Common and Unique content even under substantial structural and stylistic divergence.

Our central finding is that high change-detection accuracy alone is insufficient for reliable revision analysis: strong coarse-grained baselines achieve high *Unique* accuracy yet capture less than half of the underlying atomic changes (ACU Coverage: 0.431 vs. **0.791**). QAR-Diff addresses this gap through **Symbolic-Aware Residual Distillation** (overcoming fluency bias, P1) and **Multi-View Neuro-Symbolic Alignment** (overcoming symbolic blindness, P2), providing the atomic-level factual fidelity required for reliable, human-verifiable RAG maintenance.

6 Limitations and Future Work

Structured and Multimodal Content. QAR-Diff handles standard tables via Markdown serial-

ization, but highly abbreviated or multi-page nested tables—common in industrial specifications—remain a known failure mode, as identified in our error analysis (Section 4.2). A single-cell update in a dense specification table can constitute a complete semantic-altering edit that Phase II residual distillation occasionally misses. Extending atomic QAR synthesis to incorporate Vision-Language Models (VLMs) for multimodal, cell-level extraction from complex layouts is a priority for future work, as is support for diagrams and figures.

Evaluation Cost and LLM Dependency. The primary computational bottleneck in our setup is not QAR generation or NLI-based alignment—which are practically efficient—but rather *evaluation*. Verifying whether extracted QARs are faithfully grounded requires inserting full document text into an LLM judge, and determining whether detected differences accurately cover ground-truth changes requires a second round of long-context LLM calls. These evaluation-side costs were the primary driver of our reliance on commercial GPT-series models, as they offer reliable long-context grounding at scale. Developing more efficient evaluation protocols—such as retrieval-augmented verification or locally deployable judge models—would substantially lower this barrier and enable broader validation across open-weight LLMs and resource-constrained environments.

Continuous Revision Histories. QAR-Diff is currently designed for pairwise document comparison. A natural extension is to support incremental revision tracking over multi-version histories, treating the $\mathcal{U}^{(k)}$ sets from each comparison as structured update patches. Applied to downstream RAG systems, this would enable a *self-correcting knowledge base* that propagates only changed atomic facts into the retrieval index without requiring full re-indexing—directly addressing the RAG maintenance challenge that motivates this work.

References

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–342, Melbourne, Australia.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 13532–13546.

Changwoo Chun, Daniel Rim, and Juhee Park. 2025. LLM ContextBridge: A hybrid approach for intent and dialogue understanding in IVSR. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track (COLING)*, pages 794–806, Abu Dhabi, UAE. Association for Computational Linguistics.

Jeremy R. Cole, Palak Jain, Julian Martin Eisenschlos, Michael J. Q. Zhang, Eunsol Choi, and Bhuwan Dhingra. 2023. DiffQG: Generating questions to summarize factual changes. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 3088–3101. Association for Computational Linguistics.

Manaal Faruqui, Ellie Pavlick, Ian Tenney, and Dipanjan Das. 2018. WikiAtomicEdits: A multilingual corpus of Wikipedia edits for modeling language and discourse. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 305–315.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 758–764.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Derek Greene and James Cross. 2011. Comparative document analysis for large text corpora. In *Proceedings of the 22nd Irish Conference on Artificial Intelligence and Cognitive Science (AICS)*, pages 62–71.

Maarten Grootendorst. 2020. KeyBERT: Minimal keyword extraction with BERT. https://doi.org/10.5281/zenodo.4461265.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Mana Ihori, Hiroshi Sato, Tomohiro Tanaka, and Ryo Masumura. 2022. Multi-perspective document revision. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, pages 6128–6138.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023a. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2511–2522.

Yixin Liu, Alexander R Fabbri, Pengfei Liu, Yilun Zhao, Niklas Muennighoff, and Arman Cohan. 2023b. Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4140–4170.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1906–1919. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3428–3448. Association for Computational Linguistics.

Masato Mita, Keisuke Sakaguchi, Masato Hagiwara, Tomoya Mizumoto, Jun Suzuki, and Kentaro Inui. 2022. Towards automated document revision: Grammatical error correction, fluency edits, and beyond. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3875–3896. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 2340–2353. Association for Computational Linguistics.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose, and Graham Neubig. 2019. SQuAD adversarial examples that fool reading comprehension models. In *Proceedings of the 2019 Workshop on Machine Reading for Question Answering*, pages 56–63. Association for Computational Linguistics.

Daniel Rim, Minsoo Cho, Changwoo Chun, and Jaegul Choo. 2025. To chat or task: a multi-turn dialogue generation framework for task-oriented dialogue systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 576–592, Vienna, Austria. Association for Computational Linguistics.

Julian Risch, Timo Möller, Julian Gutsch, and Malte Pietsch. 2021. Semantic answer similarity for evaluating question answering models. In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, pages 149–157.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Findings of the Association for Computational Linguistics: ACL 2020*, pages 7881–7892.

Alexander Spangher and Jonathan May. 2021. NewsEdits: Tracking real-world news article revisions and their effects. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2533–2544.

Jan Trienes, Sebastian Joseph, Jörg Schlötterer, Christin Seifert, Kyle Lo, Wei Xu, Byron Wallace, and Junyi Jessy Li. 2024. InfoLossQA: Characterizing and recovering information loss in text simplification. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4263–4294.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. 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 5307–5315.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5008–5020.

Yuan Wang, David J. DeWitt, and Jin-Yi Cai. 2003. X-Diff: An effective change detection algorithm for XML documents. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 519–530. IEEE.

Yong Yang and 1 others. 2017. Semantic taxonomies for revision identification. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 1586–1596.

A Model Specifications

Table 6 summarizes the neural and LLM components used in **QAR-Diff**. Local neural models were run on GPU with dynamic batching (batch size = 64) to maximize throughput during alignment. All

LLM calls used temperature $T = 0.0$ to ensure deterministic outputs for both QAR generation and evaluation.

Table 6: Model specifications used in QAR-Diff.

Component	Model
Neural Network Models	
Sentence Embed.	BAAI/bge-m3 (568M)
NLI Classif.	mDeBERTa-v3-base-xnli-multilingual-nli-2mil7 (278M)
Extractive MRC	mdeberta-v3-base-squad2 (278M)
Large Language Models (Azure OpenAI)	
Syn-Edits Gen.	GPT-4.1
Phase I (Seg.)	GPT-4o-mini
Phase II (QAR)	GPT-4o-mini
Phase IV (Sum.)	GPT-4.1
Eval. (Judge)	GPT-4.1

B Implementation Details and Hyperparameters

This section details the implementation choices and hyperparameters used in QAR-Diff. Where the main paper presents conceptual scoring functions, the actual system is implemented as a staged pipeline with deterministic thresholds and filtering steps.

B.1 Phase I: Segmentation Implementation

Phase I consists of two practical steps: structure-preserving text extraction and LLM-based semantic chunking.

Structured Extraction. PDF documents are parsed with PyMuPDF. We reconstruct lines and blocks heuristically, preserve page boundaries, and serialize detected table-like structures into Markdown-style rows. XML documents are flattened with an XML parser, while plain-text files are loaded directly. The intermediate output is stored as `manual_text.md`.

Semantic Chunking. Chunking is performed by an LLM prompt that inserts an explicit split marker (`SPLIT`) between semantically coherent units. The implementation processes long documents in segments constrained by a maximum character budget. The resulting chunked output is stored as `manual_text_chunked.md`.

Chunking Parameters. The default configuration used in the main experiments is:

- Maximum input characters per LLM chunking call: 7,000
- Chunk overlap: 0
- Main pipeline chunk size parameter: 6,000

B.2 Phase II: Topic Extraction and Residual Distillation

Phase II is implemented as a hybrid topic extraction and iterative QAR generation pipeline.

Hybrid Topic Extraction. Candidate topics are extracted from each segment using a mixture of:

- linguistic signals (noun phrases, general nouns, named entities),
- rule-based symbolic patterns (numbers, years, units, monetary expressions, legal/article references),
- structure-derived cues from Markdown tables and bullet lists.

For Korean text, noun and entity candidates are extracted with Kiwi when available; for English, spaCy noun chunks and entities are used when available. Additional symbolic candidates are mined with regular expressions. Topics are then normalized and lightly deduplicated.

LLM-Based Topic Refinement. When enough raw candidates are available, a second LLM pass refines the topic list by removing duplicates, filtering meaningless items, and keeping only segment-relevant topics. This refinement improves topic quality without replacing the deterministic symbolic extraction stage.

Iterative Coverage Loop. Given the refined topic set, QAR generation proceeds topic by topic. Topics are mapped to their most relevant chunks using embedding similarity, and the LLM generates question–answer pairs in batches of 10. The final extracted QAR set is then deduplicated using:

1. exact and fuzzy string matching,
2. embedding similarity,
3. bidirectional NLI checks over questions and answers.

Table 7 lists the hyperparameters for the topic selection objective (Eq. 6) and residual update (Eq. 7). The symbolic score boost is applied as a multiplicative factor to the similarity score of any topic containing a numerical expression, measurement unit, or named entity.

Table 7: Hyperparameters for Phase II topic extraction and residual distillation.

Parameter	Value
Topic Selection	MMR-based greedy
MMR Diversity (λ)	0.4
Symbolic Score Boost	1.2× ($\mathbb{I}_{sym}$ term)
Residual Dampening	0.7
Min Similarity Thresh.	0.15
Max Topics / Segment	100
Max Residual Loops	2
Min Topic–Chunk Sim.	0.60
Max Chunks / Topic	3
QAR Batch Size	10

B.3 Phase III: Cascaded Alignment, Safety Net, and Deduplication

The conceptual alignment score in the main paper is implemented as a cascaded filtering pipeline. Rather than scoring all QAR pairs exhaustively with a single model, the system progressively narrows the candidate set. Table 8 provides the decision thresholds for each gate.

Gate 1: Embedding Retrieval and Fuzzy Rescue. QAR claims are embedded using BAAI/bge-m3 and indexed with FAISS (Inner Product). Candidate pairs above the hard similarity threshold (0.82) are retained. Borderline pairs (similarity between 0.40 and 0.82) can be rescued when their Levenshtein fuzzy matching ratio exceeds 70.

Gate 2: NLI Verification. Remaining candidate pairs are batch-scored with the multilingual NLI model. A pair is retained only if its entailment score exceeds 0.70.

Gate 3: Symbolic Consistency Check ($\mathbb{G}_{sym}$). A symbolic guard normalizes strings, extracts numbers, and enforces exact numerical consistency. If one answer contains a number and the other does not, or if the numbers differ, the pair is rejected as a semantic conflict rather than a semantic match.

Safety Net. Unmatched QARs are passed to the Retrieve-then-Verify Safety Net. The system builds FAISS dense retrieval indices over raw document chunks, retrieves top-3 contexts from the opposite document, runs extractive MRC to predict a candidate answer, and verifies the result with NLI and the symbolic gate. If second-stage verification succeeds, the item is rescued as an implicit match rather than classified as *Unique*.

Pre-processing Deduplication. Before final reporting, duplicate or near-duplicate QARs are merged using fuzzy matching, embedding similarity, and bidirectional NLI-based semantic equivalence.

Table 8: Decision thresholds for Phase III alignment, safety-net verification, and pre-processing deduplication.

Parameter	Value
Deduplication	
Fuzzy Match Thresh.	95 (Levenshtein)
Q Similarity Thresh.	0.60
A Similarity Thresh.	0.85
NLI Thresh. (Q)	0.50
NLI Thresh. (A)	0.60
Cascaded Alignment (Eq. 9)	
Hard Sim. Thresh.	0.82 (Gate 1)
Fuzzy Rescue Ratio	70 (Gate 1)
NLI Entail. Thresh.	0.70 (Gate 2)
Symbolic Gate	Exact match (Gate 3)
Safety-Net (Eq. 10)	
MRC Conf. Thresh.	0.60
NLI Safety Thresh.	0.85

C Cascaded Alignment Efficiency

The cascaded alignment design (Section 3.4) avoids exhaustive all-pairs NLI comparison. For a document pair with QAR sets $|\mathcal{P}_A|$ and $|\mathcal{P}_B|$, a brute-force approach requires $|\mathcal{P}_A| \times |\mathcal{P}_B|$ NLI calls. The cascade bounds NLI invocations to a small top-k candidate set per query unit, reducing the effective NLI verification space from the full Cartesian product to a bounded top-k candidate set per query to $O(k \cdot |\mathcal{P}_A|)$, where $k \ll |\mathcal{P}_B|$. Concretely, Gate 1 (embedding retrieval) filters out the vast majority of cross-document pairs before the NLI model is ever invoked, and Gate 2 further narrows the set before the symbolic check is applied. On the industrial manual pairs evaluated in Section 4.4, this design made per-pair alignment tractable within practical time budgets. The Sentence-NLI baseline was excluded from industrial manual evaluation because its exhaustive sentence-level NLI comparison became prohibitively expensive at the 400–700-page document scale.

D Human–LLM Agreement Analysis

To validate the reliability of our G-Eval-style LLM-as-a-Judge protocol, we conducted a human agreement study. Two domain experts manually annotated a stratified random sample of 100 QAR alignment judgments (50 *Common*, 50 *Unique*) drawn from the system's output. Experts evaluated whether each QAR was faithfully grounded in

the source documents according to our strict criteria (Common requires grounding in both versions; Unique requires grounding in exactly one). The agreement rate between human consensus and the GPT-4.1 judge was **89%**. Disagreements primarily occurred in highly ambiguous paraphrase cases where the LLM judge was slightly more conservative than human annotators. These results confirm that the LLM judge serves as a sufficiently reliable, scalable verification layer for large-scale comparative evaluation, while motivating the conservative threshold design described in Appendix B.

E Syn-Edits Dataset: Construction Pipeline

The Syn-Edits benchmark was generated through a four-stage process: web collection, LLM-based normalization, plan–edit–verify generation, and human validation.

E.1 Source Document Collection

Seed documents were collected from publicly available, domain-representative corpora spanning six domains: academic papers (arXiv), essays (educational essay repositories), medical documents (CDC/MedlinePlus), patents (Google Patents), product manuals, and Wikipedia. The original crawling pipeline covered a broader set of domains (including fairytales and news articles) during initial data collection; the final benchmark was scoped to six domains that offered a consistent mix of structured and unstructured technical content. After crawling, candidate documents were filtered for appropriate length and topical coherence, and duplicates were removed before normalization.

E.2 LLM-Based Normalization

Raw crawled documents often contain boilerplate, HTML artifacts, navigation menus, and formatting noise. An LLM-based normalization stage (P0, Section H) addresses this by:

- removing web artifacts and malformed formatting,
- rewriting content into a cleaner canonical structure with clear introduction–body–conclusion flow,
- strictly preserving all named entities, numbers, and factual content,
- normalizing document length to a target range of 1,000–3,000 characters.

The normalization output is stored as structured JSON containing a single `clean_text` field.

E.3 Plan–Edit–Verify Generation

Each normalized seed document is transformed under all six edit types (T1–T6) using a three-phase agentic loop:

Phase A – Plan. Given the edit type definition and a target number of edits scaled proportionally to document length, GPT-4.1 generates a structured modification plan listing specific target segments and intended actions (Prompt P1).

Phase B – Edit. A second GPT-4.1 call executes the plan against the original document and produces the transformed text along with a structured change log (Prompt P2). After generation, the original text field is overridden with the exact source document to prevent silent hallucination of the source.

Phase C – Verify. A two-stage verification is applied: (i) an exact string matching validator confirms that all `original` fields in the change log appear verbatim in the source document, detecting hallucinated edits; (ii) a GPT-4.1 judge evaluates semantic correctness against the edit type definition (Prompt P3). Instances failing either check are discarded and retried up to three times before being excluded from the final benchmark.

This Plan–Edit–Verify loop produced the 97.2% label consistency rate reported in Section 4.2.

Table 9: Syn-Edits transformation types and expected classification labels.

Type	Description	Label
T1	Structural Reordering	Common
T2	Lexical Paraphrasing	Common
T5	Meaning-Preserving Mix (T1+T2)	Common
T3	Terminological Revision	Unique
T4	Content Addition/Deletion	Unique
T6	Meaning-Altering Mix (T3+T4)	Unique

The final benchmark comprises 180 document pairs. Human verification on a stratified random subset of 108 pairs (three per domain-type configuration) yielded a 97.2% label consistency rate.

F Prompt Templates

This section provides the prompt templates used in the main QAR-Diff pipeline.

F.1 Phase I: Semantic Segmentation

Listing 1: Phase I: Semantic Segmentation Prompt

```
In order to index the document, you must divide the
     document into paragraphs (chunks) by the main
     content unit of the document.
Please divide the input document provided below into
      'complete paragraphs with continuous meaning'
     units.
Make the itemized or numbered items, such as the
     table of contents or order, exist as a single
     chunk.
Do not change the original order and content, and
     output only the    SPLIT    mark between each
     paragraph while maintaining the content of the
     input document as it is.
However, if there are spacing, line breaks, or
     formatting errors, you can make minimal
     corrections to the normal state.
Instead of putting complex content into a single
     chunk, work towards breaking it down into
     Atomic Semantic Units.
Connect it to one of the preceding or following
     chunks.
If there is a part of the document where the content
      is cut off at the end, do not put the
        SPLIT    mark at the end. Since the chunk
     may be connected to the next page document, we
     will separate the chunk considering the next
     document.
```

F.2 Phase II: Topic Refinement

Listing 2: Phase II: Topic Refinement Prompt

```
You are an expert document analyst. Refine the topic
      list extracted from chunk #{chunk_id}.

Tasks:
1. Remove duplicates/similar topics
2. Remove meaningless topics
3. Normalize terminology
4. Keep ONLY topics discussed in THIS chunk

Chunk:
{chunk_text}

Raw Topics ({count}):
{topics}

Output (JSON array only, no markdown):
["topic1", "topic2", ...]
```

F.3 Phase II: QAR Generation

Listing 3: Phase II: QAR Generation Prompt

```
You are an expert analyst.
Generate Question-Answer pairs from the provided
     Document Chunk based on the list of Keywords.

Rules:
1. For each Keyword, verify if it appears in the
     Chunk.
2. Generate TWO versions of the question:
   - "question": Standardized, formal style (Who/
        What/When/Where/How+Noun).
   - "question_natural": Natural questions that
        utilize expressions from the original text.
3. Answer should be keyword itself or key element
     within it.
4. Language: Write in the SAME LANGUAGE as the
     document (Detected: {lang}).
5. Output JSON:
[
  {
    "question": "Standardized Question...",
    "question_natural": "Natural context-aware
         Question...",
    "answer": "Exact Answer..."
  }
]

Document Chunk:
{chunk}

Keywords:
{topics}
```

F.4 Phase IV: Difference Summarization

Listing 4: Phase IV: Difference Summarization Prompt

```
Here's a result of the Question-Answer-
     Representation comparison from the two
     documents {codeA} and {codeB}.

1) Q&A pairs with identical questions and answers (
     common):
{common_txt}

2) Unique Q&A pairs containing only {codeA}:
{a_unique_txt}

3) Unique Q&A pairs containing only {codeB}:
{b_unique_txt}

Based on the above information, please concisely
     summarize the commonalities and differences (
     specific contents to each document) between the
      two documents.
If there is duplicate content in the Q&A pairs, you
     don't need to list them all. Simply clearly
     summarize the similarities and differences
     between the two documents.
Write in the SAME LANGUAGE as the document.

Example output:
Similarities:
- ...
Differences:
- {codeA}: ...
- {codeB}: ...
```

G Evaluation Prompts

The evaluation pipeline uses GPT-4.1 as an LLM judge to verify the grounding and factual correctness of extracted QARs against source documents.

G.1 QAR Verification Prompt

Listing 5: QAR Verification Prompt (LLM-as-Judge)

```
You are a professional auditor comparing two
     documents (Document A, Document B).

[Document A]
{doc_a_truncated}
...(omitted)...

[Document B]
{doc_b_truncated}
...(omitted)...

[Question List]
ID: 0
Question: {question_0}
Target Answer: {answer_0}
---
ID: 1
Question: {question_1}
Target Answer: {answer_1}
---
...

[Tasks]
For each ID, verify the facts and output a JSON list
     :
```

```
1. value_A: Value found in Doc A (or null)
2. value_B: Value found in Doc B (or null)
3. judgment:
   - "common": Semantically identical
   - "conflict": Semantically different (Changed
       value, Negation, etc.)
   - "missing_a": Exists in B, missing in A
   - "missing_b": Exists in A, missing in B
   - "not_found": Cannot find in both

[Output Format]
JSON List only. Example:
[{"id": 0, "value_A": "...", "value_B": "...", "
     judgment": "conflict"}]
```

G.2 Ground Truth Coverage Assessment Prompt

Listing 6: Ground Truth Coverage Assessment Prompt

```
[Ground Truth Changes]
{gt_changes_json}

[Detected QARs]
ID 0: Q: {question_0} | A: {value_A_0} -> {value_B_0
     }
ID 1: Q: {question_1} | A: {value_A_1} -> {value_B_1
     }
...

[Task]
Determine if each GT Change is represented or
     covered by any of the Detected QARs.
One GT change can be covered by multiple detected
     items.

JSON Output:
{
  "coverage_mapping": [
    {
      "gt_index": 0,
      "covered": true,
      "covered_by_ids": [0, 2]
    }
  ]
}
```

H Syn-Edits Generation Prompts

This section provides the prompts used in the Plan–Edit–Verify generation pipeline for Syn-Edits.

H.1 Document Normalization Prompt (P0)

Listing 7: Syn-Edits: Document Normalization Prompt (P0)

```
You are an expert editor. Your task is to clean up
     and rewrite the provided raw text into a high-
     quality, canonical document suitable for a
     reading comprehension dataset.

[Guidelines]
1. Noise Removal: Remove all web artifacts such as
     navigation menus, advertisements, copyright
     footers, irregular line breaks, and html tags.
2. Coherence: Rewrite the text to have a clear
     structure (Introduction - Body - Conclusion)
     and smooth logical flow.
3. Fact Preservation: You must strictly preserve all
      factual information from the original text,
     including Named Entities (people, organizations
     , products), dates, and numerical values. Do
     not hallucinate new facts.
4. Length Control: Adjust the length of the text to
     be approximately between {min_len} and {max_len
     } characters.
5. Language: The output must be in the SAME LANGUAGE
      as the input text.

[Input Raw Text]
{raw_text}

[Output Format]
Return ONLY a valid JSON object. Do not include any
     markdown code fences or explanations.
{"clean_text": "<Your cleaned and rewritten text
     here>"}
```

H.2 P1: Edit Planning

Listing 8: Syn-Edits: Modification Plan Prompt (P1)

```
You are a meticulous content planner. Your goal is
     to plan modifications for a document according
     to a specific edit type.

[Edit Type Definition: {edit_type}]
{type_description}

[Target]
- Total Number of Edits: Exactly {num_edits}
- Input Document:
{clean_text}

[Task]
Analyze the input document and list exactly {
     num_edits} specific points to modify.
Do not rewrite the document yet. Just output the
     plan.

[Output Format]
Return ONLY a valid JSON list of objects.
[
  {
    "id": 1,
    "target_segment": "<The specific sentence or
         phrase in the original text to change>",
    "planned_action": "<Specific description of what
          to do>"
  },
  ...
]
```

H.3 P2: Edit Execution

Listing 9: Syn-Edits: Document Edit Prompt (P2)

```
You are an expert editor. Execute the following
     modification plan on the document.

[Edit Type]: {edit_type}
[Input Document]:
{clean_text}

[Modification Plan]:
{plan_json}

[Task]
1. Apply the changes described in the plan to the
     Input Document.
2. Ensure the rest of the text remains unchanged.
3. Generate the structured label describing the
     changes.

[Allowed Actions]
- "reorder": Move text (T1)
- "paraphrase": Replace words/phrases (T2)
- "terminology": Change Named Entities or Numbers (
     T3)
- "added": Insert new content (T4, T6)
- "deleted": Remove content (T4, T6)

[Output Schema]
Return ONLY a valid JSON object.
{
  "original_text": "<Copy the full Input Document
       exactly>",
  "transformed_text": "<The full document after
       edits>",
  "label": {
```

```
    "type": "{edit_type}",
    "changes": [
      {
        "action": "<one of allowed actions>",
        "original": "<Exact text segment from Input
             Document (null if added)>",
        "new": "<Exact text segment from Transformed
              Document (null if deleted)>",
        "description": "<Short explanation>"
      }
    ]
  }
}
```

H.4 P3: Edit Verification

Listing 10: Syn-Edits: Verification Prompt (P3)

```
You are a strict quality assurance auditor. Evaluate
      if the modified document and its label
     correctly correspond to the intended Edit Type.

[Target Edit Type: {edit_type}]
{type_description}

[Data to Verify]
- Original Text: {original_text_excerpt}
- Transformed Text: {transformed_text_excerpt}
- Recorded Changes (Label):
{label_json}

[Checklist]
1. Consistency: Do the recorded 'original' texts
     actually exist in the Original Document?
2. Intent Match:
   - If T1 (Reorder), is the meaning preserved? Did
        any text disappear?
   - If T2 (Paraphrase), are Named Entities/Numbers
        PRESERVED? (FAIL if numbers changed)
   - If T3 (Terminology), did the meaning CHANGE by
        swapping entities/numbers?
   - If T4 (Add/Del), is the meaning altered?
3. Language: Is the language of the transformed text
      same as original?
4. Hallucination: Are there any unintended changes
     not listed in the change log?

[Output]
Return ONLY a valid JSON object.
{
  "pass": true or false,
  "reason": "<If false, explain why>",
  "confidence": 0.0-1.0
}
```