Doc-to-Atom: Learning to Compile and Compose Memory Atoms

Anonymous ACL submission

Abstract

Long input sequences are central to document understanding and multi-step reasoning of Large Language Models, yet the quadratic attention cost makes inference memory-intensive and slow. Context distillation compresses contextual information into model parameters, and recent work such as Doc-to-LoRA amortizes this process into a single forward pass to generate a LoRA adapter per document. However, producing a single monolithic adapter for all queries leads to irrelevant-query interference, limited compositional recall, and poor scalability to long-doc reasoning. To address these challenges, we propose Doc-to-Atom (Doc2Atom), a compositional parametric memory framework that decomposes each document into semantically typed knowledge atoms. Each atom is compiled into an independent micro-LoRA adapter and a provenance retrieval key. At inference time, a lightweight query router selects and assembles only the relevant atoms into a query-specific adapter, which is injected into a frozen base model. The entire system is trained end-to-end via a multi-objective distillation framework. Experiments on six QA benchmarks demonstrate that Doc2Atom substantially outperforms existing in-parameter knowledge baselines while maintaining comparable inference efficiency. Our code will be made publicly available upon acceptance.

1 Introduction

Large Language Models have achieved remarkable success across natural language understanding tasks by leveraging in-context learning (ICL), where relevant information is placed directly into the model's context window at inference time (Brown et al., 2020). This paradigm allows the model to condition its outputs on arbitrary textual evidence without parameter updates. However, in-context learning suffers from scalability bottlenecks: the quadratic attention complexity causes

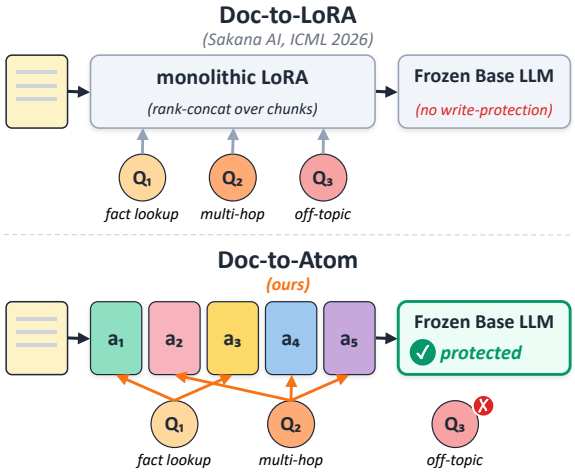

Figure 1: **Doc-to-LoRA vs. Doc-to-Atom.** *Top:* Doc-to-LoRA rank-concatenates fixed-size token chunks into one monolithic LoRA, with no semantic structure, no per-chunk index, and the same merged delta on every query. *Bottom:* Doc2Atom decomposes the document into typed semantic atoms; each a_i carries a key, a micro-LoRA, and an optional micro-KV. A two-stage router uses the keys as a per-atom index and assembles only the relevant atoms into a query-specific adapter, with support for refusing queries outside the document.

both latency and memory consumption to grow rapidly with input length (Liu et al., 2024), and generation quality degrades under longer context lengths due to attention dilution (Hsieh et al., 2024). These limitations motivate an alternative paradigm known as context distillation, which compresses contextual information into model parameters so that the model can reproduce context-conditioned behavior without retaining the original text at inference time (Askell et al., 2021; Snell et al., 2022).

While context distillation (CD) eliminates the runtime cost of long contexts, its standard formulation requires per-prompt supervised fine-tuning, which is prohibitively expensive when information sources change frequently (Zhang et al., 2023). To overcome this limitation, Doc-to-LoRA (D2L) (Charakorn et al., 2026) proposes to meta-learn the context distillation process by training a hyper-

network (Ha et al., 2016) that maps any unseen document to a LoRA adapter (Hu et al., 2022) in a single forward pass. This amortizes CD across documents, enabling fast, parameter-efficient knowledge updates without any per-document training. Nevertheless, D2L inherits a fundamental architectural constraint: it produces a single monolithic adapter per document that is applied uniformly to all downstream queries. This design introduces three interrelated problems. ① When a query is unrelated to the internalized content, the adapter still modifies the model's hidden states, causing irrelevant-query interference. ② For documents containing dozens of heterogeneous facts, a flat low-rank adapter lacks the capacity to selectively surface the evidence required by each query, resulting in limited compositional recall. ③ As document length grows, the fixed-rank adapter must compress increasingly more content into the same parameter budget, which limits scalability on long-document reasoning where the relevant evidence spans many regions of the text.

To address these challenges, we propose Doc-to-Atom (**Doc2Atom**), a framework that fundamentally restructures the document internalization pipeline from one document to one adapter into one document to many composable atoms (as shown in Figure 1). Our key insight is that documents should be decomposed into semantically typed knowledge atoms, which are minimal and self-contained units of information. Each atom is independently compiled into a micro-LoRA adapter, a provenance retrieval key, and an optional compact KV prototype. At query time, a learned two-stage router identifies the small subset of atoms relevant to the current question and assembles their micro-adapters into a query-specific composite LoRA, which is then injected into the frozen base model's designated memory layers. For queries unrelated to the internalized document, the router produces near-zero routing weights, allowing the model to fall back without interference.

Concretely, the **Doc2Atom** pipeline operates in two phases. In the offline memory compilation phase, each input document is decomposed into atoms via an LLM-driven semantic annotation process, and then a shared text encoder followed by an atom-level memory compiler produces per-atom micro-LoRA factors, provenance keys, and sparse memory masks. In the online query-conditioned assembly phase, given a query, a query encoder produces a query embedding that is matched against provenance keys via a two-stage router. The top-ranked atoms are selected, their micro-LoRA factors are composed via routing-weight-based summation and gated by sparse masks, and the resulting assembled adapter is injected into the base model for answer generation. Our main contributions are summarized as follows:

- We introduce atom-level document decomposition for parametric memory, replacing monolithic document-to-adapter compression with fine-grained, semantically typed knowledge units that enable selective retrieval and composition.
- We design a compositional memory compilation and query-conditioned assembly architecture that produces per-atom micro-LoRA adapters, routes queries to relevant atoms, and assembles query-specific adapters dynamically.
- We propose a comprehensive multi-objective training framework with curriculum-based scheduling that jointly optimizes memory generation, routing accuracy, irrelevant-query robustness, knowledge protection of the base LLM, and multi-atom composition consistency.
- Experiments on six diverse QA benchmarks demonstrate the effectiveness of **Doc2Atom** over in-parameter knowledge baselines, offering a step toward compositional parametric memory.

2 Related Work

Hypernetworks for Language Model Adaptation. Hypernetworks (Ha et al., 2016) generate the parameters of a target network conditioned on some input, and have been increasingly applied to efficient LLM adaptation. HyperDecoders (Ivison and Peters, 2022) produce instance-specific decoder parameters for multi-task NLP, while HINT (Ivison et al., 2023) extends hypernetwork instruction tuning for few-shot generalization. HyperLoRA (Lv et al., 2024) uses constrained low-rank adapter generation for cross-task transfer. More recently, Text-to-LoRA (Charakorn et al., 2025) trains a hypernetwork that directly maps context strings to LoRA parameters. Doc-to-LoRA (Charakorn et al., 2026) builds upon this line by meta-training a hypernetwork with the context distillation objective, enabling zero-shot internalization of unseen documents. While these methods demonstrate the promise of hypernetwork-based adaptation, they all produce monolithic adapters

that cannot selectively respond to different queries over the same internalized content.

Context Distillation and Compression. Context distillation trains a model to reproduce context-conditioned behavior without the original context at inference time (Askell et al., 2021; Snell et al., 2022). The core idea of compressing a larger model's knowledge into a smaller one traces back to model compression (Buciluǎ et al., 2006) and knowledge distillation (Hinton et al., 2015). Several recent works extend this paradigm: Caccia et al. (2025) train plug-and-play knowledge modules via deep context distillation; Cartridges (Eyuboglu et al., 2025) leverage sleep-time compute for prefix-tuning-based context distillation; and Generative Adapter (Chen et al., 2025) optimizes a hypernetwork using next-token prediction. Complementary approaches focus on prompt compression: Gisting (Mu et al., 2024) trains gist tokens that compress task instructions via a prefix-tuning (Li and Liang, 2021) parameterization, while LLMLingua-2 (Pan et al., 2024) performs task-agnostic prompt compression via data distillation. MEND (Li et al., 2024) meta-learns demonstration distillation for efficient in-context learning. Our work differs from all these approaches by decomposing documents into atom-level units before compilation, enabling query-conditioned selective assembly rather than monolithic compression.

Knowledge Editing and Protection. Parametric knowledge internalization risks overwriting the base model's existing capabilities. AlphaEdit (Fang et al., 2025) proposes null-space constrained knowledge editing that restricts parameter updates to subspaces orthogonal to previously used directions. Sparse memory finetuning (Lin et al., 2025) encourages different knowledge to be written into different parameter subregions, reducing inter-task interference. Our framework incorporates both L2-based and null-space knowledge protection mechanisms, combined with learned sparse memory masks that route different atom types to different layer-module combinations, thereby minimizing cross-atom interference during composition.

Long-Context Understanding. Handling long documents remains challenging for LLMs due to attention complexity and positional encoding limitations. Lost-in-the-middle phenomena (Liu et al., 2024) and attention noise (Ye et al., 2025) degrade performance on long inputs. Approaches such as activation beacons (Zhang et al., 2025) and infinite ICL (Cao et al., 2025) attempt to extend effective context length, while continual learning with hypernetworks (von Oswald et al., 2020) addresses sequential knowledge accumulation. **Doc2Atom** sidesteps these challenges entirely by compiling documents into parametric memory offline, eliminating the need to process long contexts at inference time while preserving fine-grained access through atom-level routing.

3 Method

We now present **Doc2Atom** in detail. Our goal is to internalize a document into compositional parametric memory in a way that supports selective retrieval and query-conditioned composition. To this end, **Doc2Atom** consists of three components: a data atomization pipeline that decomposes documents into structured knowledge atoms (§3.1); a compositional memory framework that compiles atoms and assembles query-specific adapters (§3.2); and a multi-objective training scheme that optimizes the system end-to-end (§3.3).

3.1 Data Atomization and Processing

The data atomization and processing pipeline, illustrated in Figure 2, transforms raw QA datasets into structured training records through three sequential stages: document-to-atom decomposition, irrelevant probe generation, and question-to-atom annotation.

Document-to-Atom Decomposition. Unlike previous methods which segment documents into fixed-size token chunks regardless of semantic boundaries, **Doc2Atom** requires an LLM to explicitly decompose each context document D into a set of non-overlapping, semantically self-contained knowledge atoms $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$. Each atom a_i represents the minimal independently retrievable and writable knowledge unit within the document, drawn from five semantic types: fact claims, entity attributes, event relations, process steps, and evidence fragments. Beyond the semantic type, each atom carries the following metadata: (i) *content*, a self-contained text preserving critical qualifiers; (ii) *source span*, strictly extracted from the original document; (iii) *retrieval text*, used as the indexing anchor; (iv) an *answer-bearing* flag; (v) an *abstraction level*; (vi) a *conflict group* identifier marking mutually exclusive atoms; (vii) a

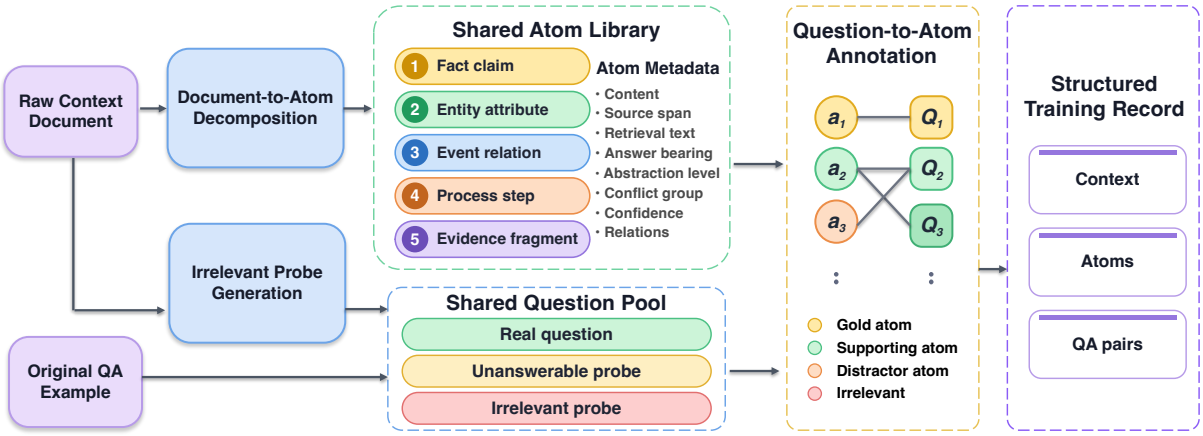

Figure 2: **Doc2Atom data atomization and processing pipeline.** Raw context documents are decomposed into semantically typed atoms stored in a shared atom library with rich metadata (e.g., semantic type, retrieval text, and inter-atom relations). Original QA examples are augmented with irrelevant probes to form a shared question pool. The question-to-atom annotation stage aligns each question to relevant atoms, producing structured training records containing context, atoms, and QA pairs with gold, supporting, and distractor labels.

continuous *confidence* score; and (viii) a set of *inter-atom relations*.

Irrelevant Probe Generation. To train the query router to remain silent on queries unrelated to the internalized content, we synthesize irrelevant probes for each context. These probes are surface-similar questions that mention entities or topics appearing in the document but whose answers cannot be found within it. This forces the router to learn fine-grained answerability boundaries beyond superficial lexical overlap, ensuring that the system falls back to base model behavior when no relevant atoms exist.

Question-to-Atom Annotation. Given the atom library $\mathcal{A}$ and the augmented question pool containing real questions and irrelevant probes, an LLM aligns each question q to atoms by producing a normalized question type, a set of gold atom IDs $\mathcal{A}_{\text{gold}}$ representing the minimal atom set required to answer the question, supporting atom IDs $\mathcal{A}_{\text{supp}}$ that aid reasoning but are not strictly necessary, and distractor atom IDs $\mathcal{A}_{\text{dist}}$ that are surface-similar and most likely to be falsely activated. Boolean signals for irrelevance and conflict presence are also generated.

3.2 Compositional Memory Framework

The **Doc2Atom** training pipeline, depicted in Figure 3, operates through offline memory compilation and online query-conditioned assembly, connected by a teacher–student training objective.

Shared Text Encoder. Rather than introducing a separate encoding model, **Doc2Atom** reuses the first L_{enc} layers of the frozen base model as a shared text encoder for both atoms and queries. Each atom's textual content is tokenized and fed through these layers, with an early-exit mechanism that extracts hidden states at a designated intermediate layer. A learned projection head maps the pooled hidden state to a d-dimensional atom embedding $e_i \in \mathbb{R}^d$. This design avoids introducing additional model parameters for encoding while leveraging the base model's pre-trained representations.

Atom-Level Memory Compiler. The memory compiler takes atom embeddings e_i as input and produces four types of outputs through a shared trunk MLP (LayerNorm followed by two linear layers with a GELU activation in between) and four parallel heads. First, a provenance key head (LayerNorm followed by a linear projection) maps each atom to a retrieval anchor vector $k_i \in \mathbb{R}^{d_k}$ that serves as the index for routing. Second, micro-LoRA heads produce per-atom low-rank factors $A^{(\ell)}_{m,i} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B^{(\ell)}_{m,i} \in \mathbb{R}^{d_{\text{out}} \times r}$ for each target module m and each designated memory layer ℓ, with B initialized to zero to ensure the memory produces no perturbation at the start of training. Third, an optional micro-KV head generates compact key-value prototypes per atom for preserving local evidence and sequential information. Fourth, an optional sparse mask head outputs per-atom logits that, after aggregation via routing weights and sigmoid activation, form a sparse gate $g \in [0,1]^{n_L \times n_M}$ controlling which layer-module

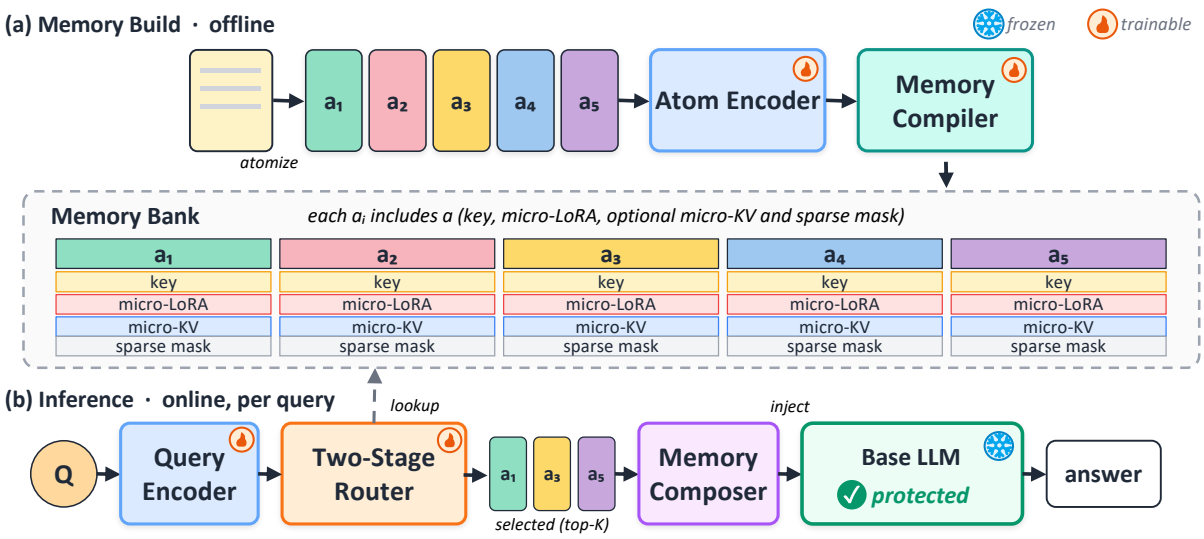

Figure 3: Doc2Atom **pipeline. (a) Memory build (offline).** The input document is decomposed into typed semantic atoms $a_1, \ldots, a_n$. A shared Atom Encoder embeds every atom, and a Memory Compiler produces a per-atom provenance key, a micro-LoRA factor, an optional micro-KV prototype and sparse mask, which together populate the Memory Bank. **(b) Inference (online, per query).** A Query Encoder (sharing weights with the Atom Encoder) embeds the query. A two-stage router consults the provenance keys to retrieve the top-K relevant atoms from the bank; the Memory Composer assembles their micro-LoRA factors into a query-specific adapter, which is injected into the frozen base LLM (together with the micro-KV prototypes when available) to produce the answer.

combinations each atom's knowledge is written to.

Two-Stage Query Router. Given a query embedding $e_q \in \mathbb{R}^d$, the router identifies relevant atoms through two stages. In the first stage, the query is projected and L2-normalized to obtain $k_q \in \mathbb{R}^{d_k}$, then matched against all provenance keys k_i via cosine maximum inner product search to retrieve the top candidates. In the optional second stage, these candidates are reranked by a frozen cross-encoder that scores the query and atom text pairs. A learned metadata bias incorporating atom confidence, answer-bearing status, conflict group membership, and semantic type is added to inject structural priors from the annotation stage. The final score s_i combines the similarity, optional reranking, and metadata bias, from which the top-K atoms are selected and their scores are softmax-normalized into sparse routing weights w_i.

Memory Composer and Injection. The memory composer assembles per-atom parameters into a query-specific adapter. For each layer ℓ and module m, micro-LoRA factors are combined via routing-weight-based summation:

$$A_c^{(\ell,m)} = \sum_{i=1}^{K} w_i A_{m,i}^{(\ell)}, \quad B_c^{(\ell,m)} = \sum_{i=1}^{K} w_i B_{m,i}^{(\ell)}. \tag{1}$$

The composed factors are then element-wise multiplied by the aggregated sparse gate g to enforce selective writing. When micro-KV is enabled, selected atoms' KV slots are scaled by routing weights and concatenated along the sequence dimension as a compact prefix. All writes are restricted to the last n_L decoder layers, following the principle that memory should be injected close to the output to minimize interference with lower-level linguistic processing.

3.3 Training Objective

The training follows a teacher–student paradigm. The teacher forward pass feeds the frozen base model with the full context D concatenated with the query q, producing full-context reference logits y_{teacher}. The student forward pass feeds the same frozen model with only the query q, but with the assembled composite LoRA injected into the designated memory layers, producing student logits y_{student}. The total training loss $\mathcal{L}_{\text{total}}$ is a weighted sum of several objectives designed to jointly optimize memory generation, routing accuracy, irrelevant-query robustness, and multi-atom composition consistency.

Language Modeling and Distillation. The fundamental objective is to reproduce the teacher's answer distribution purely through the parametric memory assembled from selected atoms. We apply

standard cross-entropy loss on the answer tokens:

$$\mathcal{L}_{\text{LM}} = -\sum_{t} \log P_{\text{student}}(y_t \mid y_{<t}, q, \mathcal{A}). \quad (2)$$

We also apply a temperature-scaled Kullback-Leibler divergence between student and teacher logits at answer positions to distill the teacher's knowledge:

$$\mathcal{L}_{\text{distill}} = \text{KL}\left(\sigma(y_{\text{teacher}}/\tau) \parallel \sigma(y_{\text{student}}/\tau)\right), \quad (3)$$

where σ denotes the softmax function and τ is the temperature parameter.

Routing Supervision. To train the router to select the correct atoms, we apply a per-atom weighted binary cross-entropy loss between the routing scores s_i and the gold atom masks $m_i \in \{0, 1\}$:

$$\mathcal{L}_{\text{route}} = -\sum_{i} \alpha_i \big[m_i \log(\sigma(s_i)) + (1 - m_i) \log(1 - \sigma(s_i)) \big], \quad (4)$$

where α_i takes values $\{0.5, 1.0, 1.5\}$ for negative, gold, and distractor atoms respectively, so distractor false positives are penalized more heavily than ordinary negatives. When the second stage is enabled, we apply a listwise softmax cross-entropy loss $\mathcal{L}_{\text{rerank}}$ for the reranking scores.

Irrelevant Query Suppression. For queries that are unrelated to the internalized document, the router should produce near-zero routing weights, and the composed adapter should not perturb the base model. We enforce this via a norm penalty on the composed LoRA factors for irrelevant queries:

$$\mathcal{L}_{\text{irrel}} = \mathbb{E}_{q \sim \mathcal{Q}_{\text{irrel}}} \sum_{\ell, m} \left(\|A_c^{(\ell,m)}\|_2^2 + \|B_c^{(\ell,m)}\|_2^2 \right). \quad (5)$$

Knowledge Protection. To prevent the assembled adapter from overwriting the base model's existing capabilities, we apply an L2 constraint on the composed deltas:

$$\mathcal{L}_{\text{protect}} = \sum_{\ell, m} \|\Delta W^{(\ell,m)}\|_F^2, \quad (6)$$

where $\Delta W^{(\ell,m)} = B_c^{(\ell,m)} A_c^{(\ell,m)}$. Alternatively, a null-space projection penalty can be used to restrict updates to directions orthogonal to the base model's principal components.

Sparse Regularization. We encourage the sparse gates g to activate only a subset of layer-module combinations, pushing activation rates toward a target density ρ:

$$\mathcal{L}_{\text{sparse}} = \sum_{\ell, m} \left(\mathbb{E}[g_{\ell,m}] - \rho \right)^2. \quad (7)$$

Composition Consistency. To ensure that combining multiple atoms does not degrade performance compared to using single atoms, we enforce a symmetric KL divergence between the main forward pass and a forced-routing forward pass using gold masks:

$$\mathcal{L}_{\text{composition}} = \text{KL}_{\text{sym}}\left(y_{\text{student}} \parallel y_{\text{forced}}\right). \quad (8)$$

Additional robustness losses include $\mathcal{L}_{\text{conflict}}$, an intra-group listwise cross-entropy requiring the router to activate only the gold atom within each conflict group, and $\mathcal{L}_{\text{confidence}}$, which aligns routing weights with annotation-side confidence priors.

Training proceeds through a multi-stage curriculum to reduce early coupling conflicts. The curriculum progressively increases the number of retrieved atoms K, activates robustness and consistency losses, and strengthens sparse writing regularization. The detailed configuration of the hyperparameters and curriculum stages is provided in the Appendix §E and §F.

4 Experiments

`Doc2Atom` addresses limitations of monolithic document-to-adapter internalization that prior parametric methods do not handle (§1). First, we compare `Doc2Atom` against three Doc-to-LoRA (D2L) baselines for accuracy and long-context generalization (Table 1). Second, we evaluate refusal on out-of-document queries (Table 2). We then assess per-update compile latency and memory (Table 3).

4.1 Experimental Setup

Models. We evaluate `Doc2Atom` on two frozen instruction-tuned base LLMs from the Doc-to-LoRA setting: Gemma-2-2B-It (Team et al., 2024) and Qwen3-4B-Instruct-2507 (Yang et al., 2025). Atom annotation is performed offline by MiniMax-M2.5 (MiniMax AI, 2026).

Baselines. On each base LLM, we compare `Doc2Atom` against three D2L baselines and an in-context reference. **D2L$_{\text{ckpt}}$** uses the official Doc-to-LoRA checkpoint (Charakorn et al., 2026) as-is.

Dataset	Gemma-2-2B-It					Qwen3-4B-Instruct				
	Base+ctx	$D2L_{ckpt}$	$D2L_{raw}$	$D2L_{atom}$	Doc2Atom	Base+ctx	$D2L_{ckpt}$	$D2L_{raw}$	$D2L_{atom}$	Doc2Atom
F1										
2WikiMultiHopQA (Ho et al., 2020)	37.69	36.08	49.84	50.39	39.62	45.31	11.25	46.06	47.43	31.74
DROP (Dua et al., 2019)	42.61	29.62	26.98	29.52	28.31	45.18	10.07	27.12	30.66	27.05
QASPER (Dasigi et al., 2021)	48.62	26.34	16.46	18.39	30.78	52.76	27.03	19.17	20.07	31.03
ROPES (Lin et al., 2019)	71.90	65.90	52.94	53.93	53.52	77.63	15.26	53.06	51.95	58.08
SQuAD (Rajpurkar et al., 2016)	86.35	73.20	18.67	19.36	59.71	78.15	29.60	15.88	18.71	59.82
LongBench-2WikiMQA	32.27	27.93	34.63	36.48	29.63	37.47	9.09	34.87	36.14	26.35
LongBench-HotpotQA	41.00	26.79	22.30	19.49	27.96	52.15	11.04	25.88	25.43	24.77
LongBench-MFQA-en	39.41	20.92	7.14	4.68	26.00	47.18	26.69	10.70	9.73	34.29
LongBench-MFQA-zh	7.73	2.05	1.57	0.41	8.57	14.79	5.17	0.70	0.54	9.27
LongBench-MuSiQue	21.62	9.14	6.95	7.39	20.86	18.14	5.24	12.25	6.41	22.32
LongBench-NarrativeQA	22.67	13.31	4.64	6.21	20.58	27.40	11.67	2.83	3.46	17.87
LongBench-QASPER	37.05	19.81	12.29	17.10	29.85	45.14	21.71	14.84	14.56	31.80
LongBench-TriviaQA	86.67	82.14	83.27	83.48	79.54	85.16	50.81	85.99	85.85	47.77
Overall	48.79	37.93	28.55	29.41	**37.99**	52.22	18.70	28.85	29.30	**35.72**
ROUGE-L										
2WikiMultiHopQA (Ho et al., 2020)	37.69	36.08	49.79	50.25	39.62	45.27	11.17	46.03	47.43	31.74
DROP (Dua et al., 2019)	42.61	29.55	26.98	29.52	28.31	45.18	10.02	27.12	30.66	27.05
QASPER (Dasigi et al., 2021)	47.26	25.39	16.25	18.19	30.44	50.72	25.39	18.91	19.83	30.70
ROPES (Lin et al., 2019)	71.90	65.85	52.94	53.93	53.52	77.63	15.25	52.98	51.95	58.08
SQuAD (Rajpurkar et al., 2016)	86.20	72.96	18.57	19.21	59.71	77.89	29.04	15.67	18.48	59.66
LongBench-2WikiMQA	32.27	27.86	34.57	36.48	29.63	37.47	8.92	34.87	36.08	26.24
LongBench-HotpotQA	40.78	26.37	22.21	19.49	27.82	51.78	10.50	25.88	25.24	24.67
LongBench-MFQA-en	38.31	19.70	6.91	4.68	25.83	45.60	24.31	9.92	8.94	33.31
LongBench-MFQA-zh	7.73	2.05	1.37	0.41	8.57	14.79	5.17	0.70	0.54	9.27
LongBench-MuSiQue	21.62	9.14	6.95	7.39	20.86	18.05	5.00	12.25	6.41	22.27
LongBench-NarrativeQA	22.37	13.27	4.64	6.15	20.27	26.56	11.20	2.83	3.46	17.82
LongBench-QASPER	35.58	18.87	11.89	16.68	29.26	42.93	20.35	14.64	14.12	31.46
LongBench-TriviaQA	86.67	82.14	83.27	83.48	79.54	85.14	50.53	85.99	85.85	47.74
Overall	48.43	37.62	28.45	29.32	**37.88**	51.65	18.13	28.74	29.17	**35.58**

Table 1: **Per-dataset F1 and ROUGE-L scores (%).** We compare Doc2Atom against three D2L baselines on two frozen base LLMs. **$D2L_{ckpt}$** uses the official checkpoint released by Sakana AI (Charakorn et al., 2026) (trained on FineWeb-Edu (Lozhkov et al., 2024), PwC (Chevalier et al., 2023), SQuAD, ROPES, and DROP); **$D2L_{raw}$** retrains D2L on our five source datasets; and **$D2L_{atom}$** retrains D2L on the atomized version of these datasets. **Base+ctx** is an ICL reference with full document access at inference time. The first five datasets are in-domain; the LongBench datasets (Bai et al., 2024) are evaluated zero-shot for all methods. Green marks cases where Doc2Atom beats **$D2L_{raw}$**. **Bold** in the *Overall* row marks the best parameterized method.

$D2L_{raw}$ retrains Doc-to-LoRA from scratch on our source datasets. **$D2L_{atom}$** retrains the same architecture on the atomized version of these datasets, isolating atom-level decomposition. **Base+ctx** feeds the full document to the frozen base LLM at inference, serving as an ICL upper bound.

Tasks and Benchmarks. We evaluate on six QA benchmarks across three task categories: short-context reading comprehension (SQuAD (Rajpurkar et al., 2016), DROP (Dua et al., 2019), ROPES (Lin et al., 2019)), long-document and multi-hop QA (2WikiMultiHopQA (Ho et al., 2020), QASPER (Dasigi et al., 2021)), and long-context understanding via eight LongBench (Bai et al., 2024) subsets. The first two categories are used for both training and in-distribution testing; LongBench is evaluated zero-shot for long-context generalization. We report word-level F1 and ROUGE-L as primary metrics.

Configuration. All other details are in Appendix.

4.2 Main Results

Doc2Atom achieves the best overall accuracy. On Gemma-2-2B-It, Doc2Atom reaches 37.99 F1 and 37.88 ROUGE-L overall, outperforming the strongest D2L baseline ($D2L_{atom}$) by 8.58 and 8.56 points. On Qwen3-4B-Instruct, Doc2Atom reaches 35.72 F1 and 35.58 ROUGE-L, exceeding $D2L_{atom}$ by 6.42 and 6.41 points. The gains are most pronounced on the eight LongBench subsets, evaluated *zero-shot* for all methods: Doc2Atom achieves up to 6× higher F1 than $D2L_{raw}$ on MFQA, MuSiQue, NarrativeQA, and QASPER, substantially narrowing the gap to the ICL upper bound (Base+ctx) on tasks unseen during training.

Atomization alone improves D2L. Training the same D2L architecture on atomized documents instead of raw documents ($D2L_{raw} \rightarrow D2L_{atom}$) yields consistent overall gains on both base LLMs (+0.86 on Gemma, +0.45 on Qwen3), confirming the effectiveness of our atomization pipeline.

Dataset	$D2L_{ckpt}$	$D2L_{raw}$	$D2L_{atom}$	Doc2Atom
Gemma-2-2B-It				
2WikiMultiHopQA	8.53	11.31	16.40	**90.01**
DROP	3.14	4.48	4.04	**92.03**
QASPER	1.20	23.49	25.67	**58.05**
ROPES	11.19	5.24	5.48	**94.66**
SQuAD	7.61	1.52	0.87	**85.25**
LongBench-2WikiMQA	8.50	15.83	22.00	**97.50**
LongBench-HotpotQA	8.00	16.00	15.83	**90.17**
LongBench-MFQA-en	2.67	18.00	11.33	**95.22**
LongBench-MFQA-zh	14.50	40.50	11.50	**95.95**
LongBench-MuSiQue	3.00	8.50	12.50	**95.33**
LongBench-NarrativeQA	1.00	7.00	24.50	**84.33**
LongBench-QASPER	1.50	21.15	22.33	**61.50**
LongBench-TriviaQA	1.00	8.50	5.50	**64.00**
Qwen3-4B-Instruct				
2WikiMultiHopQA	17.26	10.12	2.98	**18.88**
DROP	34.80	4.93	1.79	**54.71**
QASPER	43.06	54.93	39.59	**90.68**
ROPES	50.07	12.14	5.48	**76.94**
SQuAD	42.60	2.83	0.65	**54.13**
LongBench-2WikiMQA	13.20	11.00	7.50	**17.64**
LongBench-HotpotQA	15.28	10.50	10.50	**19.08**
LongBench-MFQA-en	17.96	17.33	20.67	**64.00**
LongBench-MFQA-zh	21.47	33.00	21.00	**95.00**
LongBench-MuSiQue	14.38	6.50	6.50	**28.58**
LongBench-NarrativeQA	75.04	5.50	7.00	**92.50**
LongBench-QASPER	38.35	50.50	40.33	**93.25**
LongBench-TriviaQA	6.54	4.00	3.00	**20.00**

Table 2: **F1 scores (%) of refusal on irrelevant queries.** Each document is paired with unrelated questions and the model is prompted to reply `unanswerable` when the answer cannot be grounded in the context; we report F1 against this target, so higher means more reliable refusal. The first five datasets are in-domain (used during training); The first five datasets are in-domain; LongBench rows are held out at training time. Best per row in **bold**.

4.3 Refusal on Irrelevant Queries

For each document we pair it with unrelated questions and prompt the model to reply `unanswerable`; Table 2 reports the F1 score against this refusal target. On Gemma-2-2B-It, Doc2Atom consistently outperforms all three D2L baselines on every benchmark, with refusal F1 typically above 85% and peaking at 97.50 on LongBench-2WikiMQA, while D2L baselines stay below 40% on most benchmarks. On Qwen3-4B-Instruct, absolute refusal is harder for every method (Qwen3 tends to attempt an answer even when unsupported), yet Doc2Atom still wins every benchmark, reaching QASPER (90.68), and ROPES (76.94). This advantage carries over to the eight held-out LongBench subsets, where Doc2Atom wins every subset under both base LLMs, confirming that the learned refusal generalizes beyond the training distribution.

4.4 Efficiency Analysis

Table 3 reports the mean per-update compile cost across the same six benchmarks as Table 1.

Additional Memory. Doc2Atom requires substantially less additional GPU memory than D2L: 13.43 GB vs. 24.02 GB on Gemma-2-2B-It (a 44% reduction) and 5.02 GB vs. 34.62 GB on Qwen3-4B (an 85% reduction). The top-K atom budget bounds the activation footprint by the number of selected atoms, independent of document length, whereas D2L's monolithic adapter must accommodate every chunk and scales with document size.

Update Latency. Doc2Atom is slower per update than D2L (2.10 s vs. 0.31 s on Gemma-2-2B-It; 1.10 s vs. 0.51 s on Qwen3-4B-Instruct) because the atom encoder, memory compiler, and two-stage router each add a sequential pass. Since this update is performed offline at document internalization time and decoupled from per-query inference, update latency is not a primary efficiency bottleneck.

	Gemma-2-2B-It		Qwen3-4B-Instruct	
Method	Latency	Memory	Latency	Memory
$D2L_{ckpt}$	**0.31**	24.02	**0.51**	34.62
Doc2Atom	2.10	**13.43**	1.10	**5.02**

Table 3: **Efficiency.** Mean update latency (s) and mean additional GPU memory (GB) of the compile / internalization step, averaged across the same datasets as Table 1. Lower is better (↓); best per column in **bold**.

5 Conclusion

We presented Doc2Atom, a framework that internalizes each document as a compositional library of memory atoms and retrieves and composes the relevant atoms into a query-specific memory for each query. This design addresses three limitations of monolithic document-to-adapter compression: (i) irrelevant-query interference, (ii) limited compositional recall, and (iii) poor scalability to long-document reasoning. Across six diverse QA benchmarks, Doc2Atom achieves higher accuracy than Doc-to-LoRA baselines, more reliably refuses irrelevant queries, and incurs lower per-update memory, offering a step toward scalable, compositional, and selective parametric memory.

6 Limitations

While **Doc2Atom** demonstrates the effectiveness of compositional parametric memory across a wide range of QA and long-context benchmarks, several limitations remain.

First, our current implementation uses a fixed semantic taxonomy and atom decomposition strategy during annotation. Although this design provides stable and interpretable memory units across datasets, different tasks or domains may benefit from alternative atom granularities or more flexible decomposition schemes. Exploring adaptive or task-specific atom structures is an interesting direction for future work.

Second, our experiments focus on relatively small frozen base language models, consistent with recent context distillation studies including Doc-to-LoRA. Training and evaluating substantially larger models would require significantly greater computational resources than the 8-GPU setup used in this work. While this setting highlights the effectiveness and efficiency of compositional parametric memory under constrained compute budgets, extending **Doc2Atom** to larger frontier-scale models and broader multimodal settings remains future work.

Ethical Considerations

All experiments presented in this study were conducted using publicly available datasets and models licensed for academic research purposes. To the best of our knowledge, this work does not present any ethical concerns.

References

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, and 1 others. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*. Available: https://arxiv.org/pdf/2112.00861.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Annual Meeting of the Association for Computational Linguistics*. Available: https://arxiv.org/pdf/2308.14508.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*. Available: https://arxiv.org/pdf/2005.14165.

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Available: https://dl.acm.org/doi/pdf/10.1145/1150402.1150464.

Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulić, and Alessandro Sordoni. 2025. Training plug-and-play knowledge modules with deep context distillation. In *Second Conference on Language Modeling*. Available: https://arxiv.org/pdf/2503.08727.

Bowen Cao, Deng Cai, and Wai Lam. 2025. Infinite-icl: Breaking the limit of context window size via long short-term memory transformation. In *Findings of the Association for Computational Linguistics: ACL 2025*. Available: https://aclanthology.org/2025.findings-acl.595.pdf.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert T. Lange. 2025. Text-to-lora: Instant transformer adaptation. In *International Conference on Machine Learning*. Available: https://arxiv.org/pdf/2506.06105.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. 2026. Doc-to-lora: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*. Available: https://arxiv.org/pdf/2602.15902.

Tong Chen, Hao Fang, Patrick Xia, Xiaodong Liu, Ben Van Durme, Luke Zettlemoyer, Jianfeng Gao, and Hao Cheng. 2025. Generative adapter: Contextualizing language models in parameters with a single forward pass. In *International Conference on Learning Representations*. Available: https://arxiv.org/pdf/2411.05877.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Conference on Empirical Methods in Natural Language Processing*. Available: https://aclanthology.org/2023.emnlp-main.232.pdf.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Available: https://arxiv.org/pdf/2105.03011.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Available: https://arxiv.org/pdf/1903.00161.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, and 1 others. 2025. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*. Available: https://arxiv.org/pdf/2506.06266.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained model editing for language models. In *International Conference on Learning Representations*. Available: https://arxiv.org/pdf/2410.02355.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*. Available: https://arxiv.org/pdf/1609.09106.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. Available: https://arxiv.org/pdf/1503.02531.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *International Conference on Computational Linguistics*. Available: https://arxiv.org/pdf/2011.01060.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*. Available: https://arxiv.org/pdf/2404.06654.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. Available: https://arxiv.org/pdf/2106.09685.

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew E Peters. 2023. Hint: Hypernetwork instruction tuning for efficient zero-and few-shot generalisation. In *Annual Meeting of the Association for Computational Linguistics*. Available: https://aclanthology.org/2023.acl-long.631.pdf.

Hamish Ivison and Matthew E Peters. 2022. Hyperdecoders: Instance-specific decoders for multi-task nlp. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Available: https://aclanthology.org/2022.findings-emnlp.124.pdf.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics*. Available: https://aclanthology.org/2021.acl-long.353.pdf.

Yuyang Li, Xinyao Ma, Shuhe Lu, K Lee, X Liu, and C Guo. 2024. Mend: Meta demonstration distillation for efficient and effective in-context learning. In *International Conference on Learning Representations*. Available: https://arxiv.org/pdf/2403.06914.

Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. 2025. Continual learning via sparse memory finetuning. *arXiv preprint arXiv:2510.15103*. Available: https://arxiv.org/pdf/2510.15103.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *2nd Workshop on Machine Reading for Question Answering*. Available: https://arxiv.org/pdf/1908.05852.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. In *Transactions of the Association for Computational Linguistics*. Available: https://aclanthology.org/2024.tacl-1.9.pdf.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Chenghao Lv, Lei Li, Shuo Zhang, Gang Chen, Fei Qi, Ningyu Zhang, and Hai-Tao Zheng. 2024. Hyperlora: Efficient cross-task generalization via constrained low-rank adapters generation. In *Findings of the Association for Computational Linguistics: ACL 2024*. Available: https://aclanthology.org/2024.findings-emnlp.956.pdf.

MiniMax AI. 2026. Minimax-m2.5.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*. Available: https://arxiv.org/pdf/2304.08467.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, and 1 others. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*. Available: https://aclanthology.org/2024.findings-acl.57.pdf.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*. Available: https://aclanthology.org/D16-1264.pdf.

Charlie Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *arXiv preprint arXiv:2209.15189*. Available: https://arxiv.org/pdf/2209.15189.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*. Available: https://arxiv.org/pdf/2408.00118.

Johannes von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. 2020. Continual learning with hypernetworks. In *International Conference on Learning Representations*. Available: https://arxiv.org/pdf/1906.00695.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*. Available: https://arxiv.org/pdf/2505.09388.

Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. 2025. Differential transformer. In *International Conference on Learning Representations*. Available: https://arxiv.org/pdf/2410.05258.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zheng Dou. 2025. Long context compression with activation beacon. In *International Conference on Learning Representations*. Available: https://arxiv.org/pdf/2401.03462.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaowei Sun, Shuhe Wang, Jiwei Li, Runqi Hu, Tianwei Zhang, Fei Wu, and 1 others. 2023. Instruction tuning for large language models: A survey. In *arXiv preprint arXiv:2308.10792*. Available: https://arxiv.org/pdf/2308.10792.

Contents of Appendix

A Baselines **11**

B Benchmarks **11**

C Atomization Pipeline **12**
- C.1 Context Decomposition 13
- C.2 Irrelevant Probe Generation 13
- C.3 Question-to-Atom Annotation . . 14
- C.4 Failure Handling and Quality Logging 14

D Detailed End-to-End Architecture **14**

E Implementation Details **15**

F Curriculum **15**

G Atomization Example **16**

A Baselines

We compare **Doc2Atom** against three Doc-to-LoRA (D2L) baselines and an in-context-learning (ICL) reference on two frozen base LLMs. All methods use identical prompts, chat-template wrapping, token budgets, greedy decoding settings and truncation policies. The base LLMs are kept frozen across all methods, so any per-method difference is attributable to the parametric-memory mechanism alone.

Base+ctx. A pure in-context-learning reference. The full document and the question are concatenated into a single chat-formatted prompt and fed directly to the frozen base LLM, serving as the ICL upper bound.

D2L$_{\text{ckpt}}$. The official Doc-to-LoRA checkpoint released by Sakana AI (Charakorn et al., 2026), trained on FineWeb-Edu (Lozhkov et al., 2024), PwC (Chevalier et al., 2023), SQuAD, ROPES, and DROP. Used as-is, without any further fine-tuning on our atomised data. This baseline isolates how much of the gap on our evaluation is due to training data versus the algorithm itself.

D2L$_{\text{raw}}$. The Doc-to-LoRA architecture retrained from scratch on our five source datasets, using the original token-chunk decomposition (no atom labels). This is the apples-to-apples comparison: same training data pool, same data caps, same base LLM, and same optimizer settings as **Doc2Atom**.

D2L$_{\text{atom}}$. The same Doc-to-LoRA architecture trained on the *atomized* version of our datasets: the fixed-size token chunks are replaced with the typed atom strings produced by our annotation pipeline. This ablation isolates the effect of the atom decomposition step alone.

B Benchmarks

- **SQuAD** (Rajpurkar et al., 2016): SQuAD is a large-scale reading comprehension benchmark constructed from Wikipedia articles. Each example consists of a paragraph, a question written by crowdworkers, and an answer that appears as a contiguous span in the corresponding passage. The dataset evaluates whether models can locate and extract the correct answer from a given context rather than selecting from predefined answer choices.

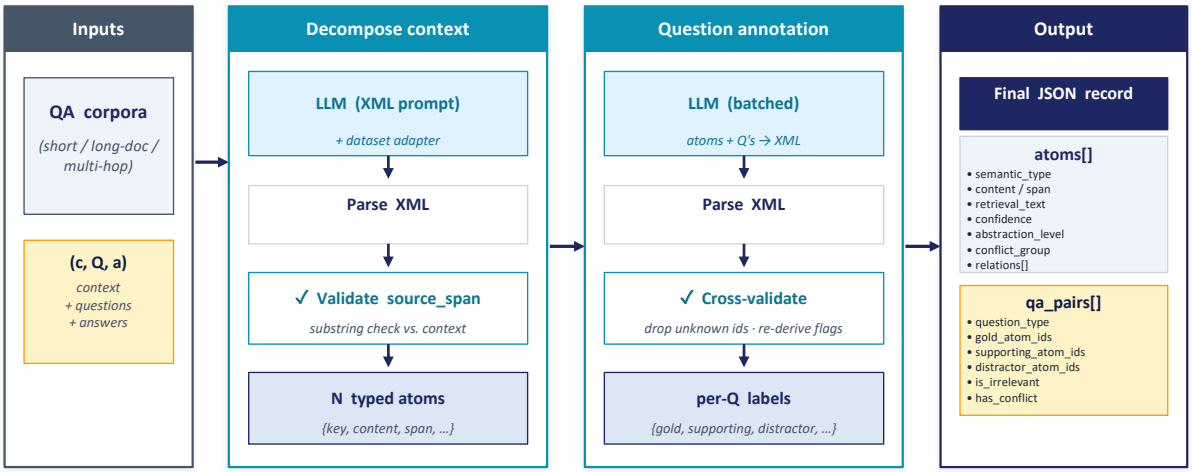

Figure 4: **Atom annotation pipeline.** An XML-driven annotator with a dataset-specific adapter decomposes each context into typed atoms; outputs are parsed and each `source_span` is verified against the original context. A second batched call then takes the validated atom bank with the question list and emits per-question role labels, which are cross-validated deterministically (dropping unknown `atom_ids` and re-deriving `has_conflict` from atom-level fields). The merged output is serialised as a JSON record containing `atoms[]` and `qa_pairs[]` arrays; the complete field schema is given in Table 4.

- **DROP** (Dua et al., 2019): DROP is a reading comprehension benchmark requiring discrete reasoning over paragraphs. It contains over 96K questions crowdsourced from Wikipedia, where models must perform operations such as addition, counting, sorting, and comparison, going beyond span extraction to test numerical and compositional reasoning abilities.

- **ROPES** (Lin et al., 2019): ROPES is a reading comprehension benchmark focusing on reasoning over paragraph effects in situations. It provides a background passage describing causal relationships, a novel situation, and questions that require applying the inferred cause–effect relations to the new context, testing models' ability to transfer knowledge beyond the original text.

- **2WikiMultihop** (Ho et al., 2020): 2WikiMultihop is a multi-hop question answering dataset constructed from Wikipedia and Wikidata, designed to require reasoning over multiple paragraphs. Each example includes not only the answer and supporting sentences, but also structured evidence in the form of triples that explicitly capture the reasoning path from the question to the answer, enabling more comprehensive evaluation of multi-hop reasoning.

- **QASPER** (Dasigi et al., 2021): QASPER is an information-seeking question answering benchmark built on full-text NLP research papers. Questions are written after annotators read only the title and abstract, while answers are derived from the full paper with supporting evidence. This setting requires document-level understanding and often involves reasoning over multiple paragraphs, tables, or figures.

- **LongBench** (Bai et al., 2024): LongBench is a bilingual, multi-task benchmark designed to evaluate long-context understanding in large language models. It consists of 21 datasets across 6 task categories, including single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion, with context lengths extending to thousands of tokens, enabling comprehensive evaluation of long-range reasoning and comprehension.

C Atomization Pipeline

This appendix details the annotation pipeline that produces the *atomized supervision* used in §3.1. We expand on several aspects deferred from the main text, including the prompt format, the post-processing validation rules, and the failure handling policy. The full output schema is given in Table 4, and the pipeline is illustrated in Fig. 4.

Annotator and prompt format. All annotation is performed by MiniMax-M2.5 (MiniMax AI, 2026) driven by structured XML prompts; all post-processing downstream of the annotator is deter-

Table 4: Output schema produced by our annotation pipeline. Atom-level fields are emitted once per atom during context decomposition, and question-level fields are emitted once per question during question-to-atom annotation. Field names match the codebase identifiers consumed by the training collator.

Field	**Description**
Atom-level	
`atom_id`	Stable identifier (`atom_0`, `atom_1`, ...) used for cross-referencing within the sample.
`semantic_type`	One of {`fact_claim`, `entity_attribute`, `event_relation`, `process_step`, `evidence_fragment`}.
`content`	Self-contained natural-language statement of the atom.
`source_span`	Verbatim contiguous substring of the original context, validated post-hoc.
`retrieval_text`	Query-friendly form (entity plus attribute plus key qualifiers), used as the indexing anchor for routing.
`is_answer_bearing`	Whether the atom can directly carry an answer, as opposed to only providing context.
`abstraction_level`	One of {`abstract`, `evidence`, `hybrid`}, used by the router's metadata bias.
`conflict_group`	Cluster identifier shared by mutually inconsistent atoms; `null` otherwise.
`confidence`	Annotator's calibrated confidence in $[0, 1]$.
`relations`	Typed edges to other atoms (`supports`, `elaborates`, `causes`, `precedes`, `follows`, `contradicts`, `same_entity`, `same_event`, `part_of`).
Question-level	
`question_type`	One of ten types (`single_hop`, `multi_hop`, `comparison`, `temporal`, `causal`, `procedural`, `aggregation`, `field_lookup`, `evidence_grounded`, `irrelevant`).
`gold_atom_ids`	Minimal set of atoms strictly required to answer the question.
`supporting_atom_ids`	Atoms aiding reasoning but not strictly required.
`distractor_atom_ids`	Hardest negatives: atoms a router would most plausibly mis-activate.
`relevant_atom_ids`	Convenience union `gold` $\cup$ `supporting`, re-derived after validation.
`is_irrelevant`	True iff the document does not contain the answer; forces `gold_atom_ids` $= \emptyset$.
`has_conflict`	True iff any `gold` or `supporting` atom shares a `conflict_group` with another; re-derived from atoms.
`is_generated_irrelevant`	True iff the question was synthesised by the irrelevant-probe generator rather than originating from the source corpus.

ministic. We use XML for both input and output because, compared with JSON, XML is more tolerant of partial output: a single malformed tag does not invalidate the rest. It also supports finer-grained per-tag validation and degrades gracefully when the annotator truncates mid-response. We verified empirically that XML parsing recovers a strictly larger fraction of samples than JSON parsing under the same prompts.

C.1 Context Decomposition

Given a context c, the annotator emits an XML-tagged list of atoms, each a self-contained, minimally redundant unit of knowledge. The five semantic types and eight metadata fields per atom are described in §3.1 and tabulated in Table 4 (top half). Atoms can additionally link to one another through nine *relation* types encoding bridge entities, supporting evidence, contradictions, and procedural ordering.

The decomposition prompt is composed of a base system prompt and a dataset-specific adapter that captures the structural priors of the source corpus. On multi-hop corpora the adapter forces the annotator to surface bridge entities and to link parallel mentions via `same_entity` relations; on numeric-reasoning corpora it requires every comparable number to be split into its own atom and preserved verbatim in `source_span`.

Source-span validation. Every atom must carry a `source_span` field: a contiguous substring of the original context that grounds the atom. Immediately after parsing, we run a substring search (optionally fuzzy with whitespace normalisation) against c. Atoms whose span is not located receive a `span_valid`=FALSE flag and are surfaced to downstream filtering. This single guard rule eliminates the most common annotator hallucination, namely fabricated quotations, and we find it sufficient in practice without requiring a second annotator call for verification.

C.2 Irrelevant Probe Generation

To support the irrelevant-suppression objective $\mathcal{L}_{\text{irrel}}$ (§3.3), we augment each context with synthesised *irrelevant probes* as described in §3.1: surface-similar questions that mention entities or topics from the context but cannot be answered from it. A separate prompt conditioned on the con-

text asks the annotator to produce n such plausible-but-unanswerable questions. Generated probes are tagged `is_generated_irrelevant=true`, and the subsequent question-to-atom annotation step forces `gold_atom_ids`$= \emptyset$ on them.

C.3 Question-to-Atom Annotation

Given the validated atom bank and the question list Q (real questions plus the irrelevant probes generated above), the annotator labels every question with the role fields listed in Table 4 (bottom half). The two most consequential are `gold_atom_ids`, defined as the *minimal* set of atoms strictly necessary to answer the question, and `distractor_atom_ids`, the hardest atoms a router would most plausibly mis-activate. Dataset-specific adapters again shape what counts as a hard distractor: on multi-hop corpora, atoms that bind the right attribute to the wrong entity; on field-lookup corpora, atoms with the same entity but a different field, or the same field with a different entity.

Cross-validation. We do not blindly trust the annotator's question-level output. After parsing, we apply three deterministic checks: (i) drop any `atom_id` that does not appear in the validated atom bank; (ii) recompute `relevant_atom_ids`=`gold`∪`supporting` from the surviving lists; (iii) re-derive `has_conflict` from the atoms' own `conflict_group` fields rather than trusting the annotator-emitted flag. When the annotator marks a question `is_irrelevant`, we additionally clear its `gold_atom_ids` and force `question_type=irrelevant` to enforce flag consistency.

C.4 Failure Handling and Quality Logging

We adopt a graceful-degradation policy. A failure during context decomposition causes the sample to be dropped. A failure during question-to-atom annotation keeps the validated atom bank but emits empty question annotations. An irrelevant-probe generation failure is logged and ignored. Per-sample warnings (invalid semantic types, dropped relation targets, span mismatches, unknown atom ids, etc.) are preserved in an `annotation_meta` field on each sample so downstream consumers can filter on annotation quality without re-running the annotator.

D Detailed End-to-End Architecture

This appendix walks through the complete **Doc2Atom** architecture shown in Figure 5, which combines the offline memory compilation phase, the online query-conditioned assembly phase, and the teacher–student training loop into a single end-to-end diagram.

Offline memory compilation (left). For each document, the atom annotation pipeline (§C) emits a list of typed atoms. The shared text encoder (§3.2) maps each atom into a latent embedding e_i, and the atom-level memory compiler routes e_i through four parallel heads to produce (i) a provenance key k_i, (ii) per-atom micro-LoRA factors $\{A^{(\ell)}_{m,i}, B^{(\ell)}_{m,i}\}$ for every target module m and memory layer ℓ, (iii) an optional micro-KV prototype, and (iv) an optional sparse layer-module mask. These four outputs jointly form the *memory bank* for that document, indexed by atom ID.

Online query-conditioned assembly (right). At inference time, the same shared encoder embeds the query, and the two-stage router consults the provenance keys to retrieve the top-K relevant atoms. The memory composer aggregates the selected atoms' micro-LoRA factors into a query-specific adapter $\Delta W^{(\ell,m)} = B_c^{(\ell,m)} A_c^{(\ell,m)}$, optionally gated by the aggregated sparse mask, and injects it into the base LLM's memory layers along with the corresponding micro-KV prototypes when enabled. The base LLM itself remains frozen, so the only inference-time computation that **Doc2Atom** adds on top of a vanilla forward pass is the two-stage routing step and the rank-r adapter multiplication.

Teacher–student training (bottom). Each (c, q, a) tuple is paired with two forward passes through the same frozen base LLM. The *teacher* pass conditions on the full context c concatenated with the query q and produces reference logits y_{teacher}. The *student* pass conditions only on q but has the assembled adapter injected into the memory layers, producing y_{student}. The trainable modules (the atom encoder projection head, the memory compiler, and the router) are optimized against the joint objective described in §3.3, which combines language modeling, teacher distillation, routing supervision, irrelevant-query suppression, knowledge protection, sparse regularization, and composition consistency.

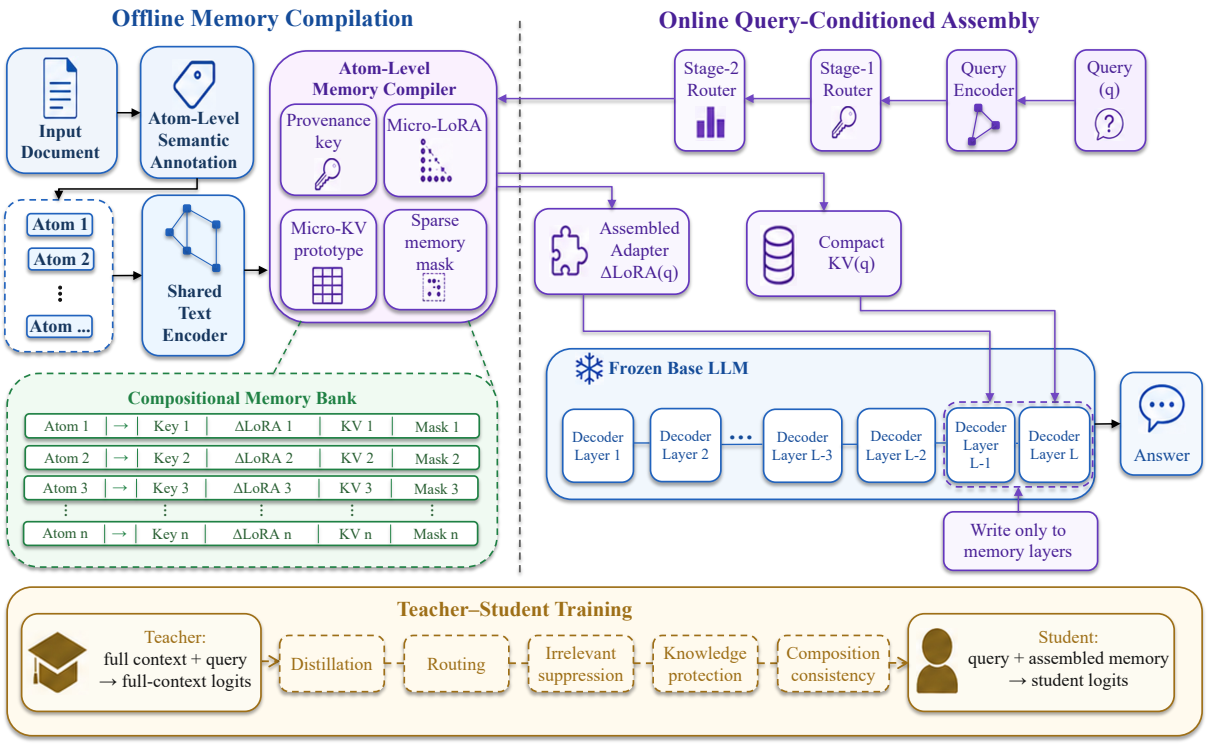

Figure 5: **More detailed architecture of the Doc2Atom. Left:** Input documents undergo atom-level semantic annotation, and a shared text encoder followed by the atom-level memory compiler produces per-atom provenance keys, micro-LoRA factors, micro-KV prototypes, and sparse memory masks. **Right:** Given a query, a two-stage router selects relevant atoms, whose micro-LoRA factors are composed into an assembled adapter and injected into the frozen base LLM's memory layers. **Bottom:** The teacher processes full context plus query to produce reference logits, while the student uses only the query plus assembled memory, trained via distillation, routing, irrelevant suppression, knowledge protection, and composition consistency losses.

E Implementation Details

All experiments are conducted on a cluster of 8 NVIDIA A100 80GB GPUs interconnected via NVLink, using PyTorch 2.1 with DeepSpeed ZeRO-2 for distributed training. We use FlashAttention-2 for efficient attention computation throughout both training and inference. Both base models are kept entirely frozen in bfloat16 precision, while all trainable modules including the atom encoder projection head, memory compiler trunk and heads, and query router are maintained in float32 for optimization stability.

For the memory compiler, we set the latent dimension $d = 256$, LoRA rank $r = 8$ with scaling factor $\alpha = 16$ and zero dropout, targeting the query, value, output, and down projection modules in the last $n_L = 4$ decoder layers. The shared text encoder extracts hidden states from the first $L_{\text{enc}} = 4$ layers of the frozen base LLM for both base models. The two-stage router uses a high-recall retrieval with $K_1 = 32$ and a final selection of $K = 8$ atoms. The learned 4-dimensional metadata bias is enabled, while the second stage cross-encoder reranking is disabled by default and used only in ablation studies. We optimize with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$) at a learning rate of 1×10^{-4} on Gemma-2-2B-It and 8×10^{-5} on Qwen3-4B-Instruct-2507, with cosine scheduling and 500 warm-up steps, weight decay of 0.01, and gradient clipping at 1.0. The effective global batch size is 128. The distillation temperature τ is set to 1.5. Maximum context length is 8192 tokens, with atom texts truncated to 96 tokens and answers to 128 tokens. Each sample contains at most 64 atoms and 8 questions. Evaluation uses greedy decoding with a maximum of 128 new tokens. The full curriculum schedule (loss-weight transitions, top-K ramps, and stage boundaries) is provided in §F; the complete per-LLM hyperparameter configuration is listed in Table 5.

F Curriculum

Training proceeds through a four-stage curriculum that gradually increases task difficulty and activates

losses in dependency order. Each stage is specified by a step range, the top-K atom budget passed to the router, and the set of active losses.

Stage 1. Single-atom memory (steps 0–2000, $K{=}1$). The router selects exactly one atom per query. Only the core losses are active: $\mathcal{L}_{\text{LM}}$, $\mathcal{L}_{\text{distill}}$, $\mathcal{L}_{\text{route}}$, $\mathcal{L}_{\text{irrel}}$, and $\mathcal{L}_{\text{protect}}$.

Stage 2. Multi-atom composition (steps 2000–6000, $K{=}4$). The router selects four atoms; $\mathcal{L}_{\text{composition}}$ and $\mathcal{L}_{\text{sparse}}$ become active.

Stage 3. Robustness (steps 6000–9000, $K{=}8$). The router selects eight atoms; the auxiliary robustness losses $\mathcal{L}_{\text{conflict}}$ and $\mathcal{L}_{\text{confidence}}$ are added, and the weight on $\mathcal{L}_{\text{irrel}}$ is increased.

Stage 4. Full composition (steps 9000+, $K{=}8$). All losses remain active and the weight on $\mathcal{L}_{\text{sparse}}$ is strengthened to push the gate density closer to the target ρ.

Training runs for up to 50 epochs with early stopping on the validation language-modeling loss.

G Atomization Example

To illustrate the pipeline of §C, Fig. 6 walks through one complete record from 2WikiMultiHopQA. Part 1 shows the raw 10-passage input document; Parts 2 and 3 list the 17 atoms extracted by the annotator together with their metadata; and Part 3 also gives the three Stage-2 questions with their gold/supporting/distractor labels.

Annotation case study (Part 1 of 3) — 2WikiMultiHopQA, id `2wiki_000b2e3d098711ebbdb0ac1f6bf848b6`, type *comparison*, 17 atoms / 3 questions

(a) Input document (10 Wikipedia stub passages, full text) passages relevant to q_0 and q_1 are highlighted in soft yellow

Passage [Telephone numbers in Ascension Island]: Country Code:+ 247
 International Call Prefix: 00 Ascension Island does not share the same country code(+290) with the rest of St Helena.

Passage [Satellite tournament]: A satellite tournament is either a minor tournament or event on a competitive sporting tour or one of a group of such tournaments that form a series played in the same country or region.

Passage [Qaserdalu]: Qaserdalu(also Romanized as Qaṣerdālū) is a village in Tolbozan Rural District, Golgir District, Masjed Soleyman County, Khuzestan Province, Iran. At the 2006 census, its population was 97, in 23 families.

Passage [Lubnowy]: Lubnowy is part of the name of two villages, both located in Gmina Susz, within Iława County, Warmian- Masurian Voivodeship, Poland:

Passage [Radzice]: Radzice is part of the name of two villages, both located in Gmina Drzewica, within Opoczno County, Łódź Voivodeship, Poland:

Passage [Limestone Coast]: The Limestone Coast is a name used since the early twenty- first century for a South Australian government region located in the south east of South Australia which immediately adjoins the continental coastline and the Victorian border. The name is also used for a tourist region and a wine zone both located in the same part of South Australia.

Passage [Tunstall, Virginia]: Tunstall is an unincorporated community in New Kent County, Virginia, United States. Foster's Castle and Hampstead, both located in Tunstall, are listed on the National Register of Historic Places.

Passage [Quyujoq]: Quyujoq(also Romanized as Qūyūjoq; also known as Kūjīkh and Qūjūq) is a village in Anzal -e Jonubi Rural District, Anzal District, Urmia County, West Azerbaijan Province, Iran. At the 2006 census, its population was 93, in 14 families.

Passage [Jawty]: Jawty is part of the name of two villages, both located in Gmina Susz, within Iława County, Warmian- Masurian Voivodeship, Poland:

Passage [Greenbury Point Light]: Greenbury Point Light was the name of two lighthouses in the Chesapeake Bay, both located at the mouth of the Severn River in Annapolis, Maryland.

Figure 6: **Annotation case study (Record 3) — Part 1 of 3.** The complete input document, a 10-passage Wikipedia bundle from 2WikiMultiHopQA. Three passages are relevant to the Stage-2 questions and are highlighted in soft yellow : *Telephone numbers in Ascension Island* (gold for the field-lookup question q_1), *Qaserdalu* and *Quyujoq* (gold for the comparison question q_0). The remaining seven passages serve as distractors, including three Polish "X is part of the name of two villages…" passages (Lubnowy, Radzice, Jawty) that the annotator explicitly marks as hard distractors for q_0. The 17 atoms a_0–a_{16} derived from this document are listed in Fig. 6 (Parts 2–3); all atoms in this example pass the substring check.

Annotation case study (Part 2 of 3) — 17 annotated atoms, first half (a_0–a_9)

(b) 17 annotated atoms — first half (a_0–a_9)

semantic types in this example: **entity_attribute** **fact_claim**

a$_0$ **entity_attribute**
conf=0.95 | span [49,67] ✓ | evidence | answer-bearing ✓
content: Ascension Island has country code +247.
source_span: *"Country Code:+ 247"*
retrieval: `Ascension Island country code`
relations: (none) **conflict_group:** (none)

a$_1$ **entity_attribute**
conf=0.95 | span [73,102] ✓ | evidence | answer-bearing ✓
content: International Call Prefix for Ascension Island is 00.
source_span: *"International Call Prefix: 00"*
retrieval: `Ascension Island international call prefix`
relations: (none) **conflict_group:** (none)

a$_2$ **fact_claim**
conf=0.95 | span [103,191] ✓ | evidence | answer-bearing ✓
content: Ascension Island does not share the same country code (+290) with the rest of St Helena.
source_span: *"Ascension Island does not share the same country code(+290) with the rest of St Helena."*
retrieval: `Ascension Island different country code from St Helena`
relations: (none) **conflict_group:** (none)

a$_3$ **fact_claim**
conf=0.90 | span [225,411] ✓ | **abstract** | answer-bearing ✓
content: A satellite tournament is a minor tournament on a competitive sporting tour or one of a group of tournaments in the same country or region.
source_span: *"A satellite tournament is either a minor tournament or event on a competitive sporting tour or one of a group of such tournaments that form a series played in the same country or region."*
retrieval: `satellite tournament definition`
relations: (none) **conflict_group:** (none)

a$_4$ **entity_attribute**
conf=0.95 | span [434,581] ✓ | evidence | answer-bearing ✓
content: Qaserdalu is a village in Tolbozan Rural District, Golgir District, Masjed Soleyman County, Khuzestan Province, Iran.
source_span: *"Qaserdalu(also Romanized as Qaṣerdālū) is a village in Tolbozan Rural District, Golgir District, Masjed Soleyman County, Khuzestan Province, Iran."*
retrieval: `Qaserdalu location`
relations: (none) **conflict_group:** (none)

a$_5$ **entity_attribute**
conf=0.95 | span [582,640] ✓ | evidence | answer-bearing ✓
content: Qaserdalu had population of 97 in 23 families in 2006.
source_span: *"At the 2006 census, its population was 97, in 23 families."*
retrieval: `Qaserdalu 2006 census population`
relations: (none) **conflict_group:** (none)

a$_6$ **entity_attribute**
conf=0.95 | span [661,793] ✓ | evidence | answer-bearing ✓
content: Lubnowy is part of the name of two villages in Gmina Susz, Iława County, Warmian-Masurian Voivodeship, Poland.
source_span: *"Lubnowy is part of the name of two villages, both located in Gmina Susz, within Iława County, Warmian- Masurian Voivodeship, Poland:"*
retrieval: `Lubnowy villages location`
relations: (none) **conflict_group:** (none)

a$_7$ **entity_attribute**
conf=0.95 | span [814,939] ✓ | evidence | answer-bearing ✓
content: Radzice is part of the name of two villages in Gmina Drzewica, Opoczno County, Łódź Voivodeship, Poland.
source_span: *"Radzice is part of the name of two villages, both located in Gmina Drzewica, within Opoczno County, Łódź Voivodeship, Poland:"*
retrieval: `Radzice villages location`
relations: (none) **conflict_group:** (none)

a$_8$ **entity_attribute**
conf=0.95 | span [968,1126] ✓ | evidence | answer-bearing ✓
content: The Limestone Coast is a South Australian government region in the south east of South Australia.
source_span: *"The Limestone Coast is a name used since the early twenty- first century for a South Australian government region located in the south east of South Australia"*
retrieval: `Limestone Coast location and name origin`
relations: (none) **conflict_group:** (none)

a$_9$ **entity_attribute**
conf=0.95 | span [1127,1204] ✓ | evidence | answer-bearing ✓
content: The Limestone Coast adjoins the continental coastline and the Victorian border.
source_span: *"which immediately adjoins the continental coastline and the Victorian border."*
retrieval: `Limestone Coast geographical boundaries`
relations: (none) **conflict_group:** (none)

Figure 6: **(continued, Part 2 of 3)** The first ten atoms a_0–a_9, each carrying its semantic type (chip colour), confidence, `source_span` with character offsets and a substring validity check (✓/✗), `abstraction_level`, `is_answer_bearing` flag, `retrieval_text`, `relations` and `conflict_group`. All atoms in this example pass the substring check (✓). a_3 (the satellite-tournament definition) is the only atom marked `abstraction_level`=*abstract*; the rest are *evidence*-anchored to verbatim spans.

Annotation case study (Part 3 of 3) — atoms a_{10}–a_{16}, 3 annotated questions, annotation meta

(b) 17 annotated atoms — second half (a_{10}–a_{16})

types: **entity_attribute** **fact_claim**

a$_{10}$ **entity_attribute**
conf=0.95 | span [1205,1313] ✓ | evidence | answer-bearing ✓
content: The Limestone Coast name is also used for a tourist region and wine zone in the same part of South Australia.
source_span: *"The name is also used for a tourist region and a wine zone both located in the same part of South Australia."*
retrieval: `Limestone Coast tourist and wine zone usage`
relations: (none) **conflict_group:** (none)

a$_{11}$ **entity_attribute**
conf=0.95 | span [1345,1429] ✓ | evidence | answer-bearing ✓
content: Tunstall is an unincorporated community in New Kent County, Virginia, United States.
source_span: *"Tunstall is an unincorporated community in New Kent County, Virginia, United States."*
retrieval: `Tunstall Virginia location`
relations: (none) **conflict_group:** (none)

a$_{12}$ **entity_attribute**
conf=0.95 | span [1430,1542] ✓ | evidence | answer-bearing ✓
content: Foster's Castle and Hampstead are both located in Tunstall, Virginia and listed on the National Register of Historic Places.
source_span: *"Foster's Castle and Hampstead, both located in Tunstall, are listed on the National Register of Historic Places."*
retrieval: `Historic places in Tunstall Virginia`
relations: (none) **conflict_group:** (none)

a$_{13}$ **entity_attribute**
conf=0.95 | span [1563,1740] ✓ | evidence | answer-bearing ✓
content: Quyujoq is a village in Anzal-e Jonubi Rural District, Anzal District, Urmia County, West Azerbaijan Province, Iran.
source_span: *"Quyujoq(also Romanized as Qūyūjoq; also known as Kūjīkh and Qūjūq) is a village in Anzal -e Jonubi Rural District, Anzal District, Urmia County, West Azerbaijan Province, Iran."*
retrieval: `Quyujoq location`
relations: (none) **conflict_group:** (none)

a$_{14}$ **entity_attribute**
conf=0.95 | span [1741,1799] ✓ | evidence | answer-bearing ✓
content: Quyujoq had population of 93 in 14 families in 2006.
source_span: *"At the 2006 census, its population was 93, in 14 families."*
retrieval: `Quyujoq 2006 census population`
relations: (none) **conflict_group:** (none)

a$_{15}$ **entity_attribute**
conf=0.95 | span [1818,1948] ✓ | evidence | answer-bearing ✓
content: Jawty is part of the name of two villages in Gmina Susz, Iława County, Warmian-Masurian Voivodeship, Poland.
source_span: *"Jawty is part of the name of two villages, both located in Gmina Susz, within Iława County, Warmian- Masurian Voivodeship, Poland:"*
retrieval: `Jawty villages location`
relations: (none) **conflict_group:** (none)

a$_{16}$ **entity_attribute**
conf=0.95 | span [1983,2129] ✓ | evidence | answer-bearing ✓
content: Greenbury Point Light refers to two lighthouses at the mouth of the Severn River in Annapolis, Maryland.
source_span: *"Greenbury Point Light was the name of two lighthouses in the Chesapeake Bay, both located at the mouth of the Severn River in Annapolis, Maryland."*
retrieval: `Greenbury Point Light location`
relations: (none) **conflict_group:** (none)

(c) 3 annotated questions

legend: **gold** **support** **distractor**

q_0 **comparison** | **is_irrelevant:** ✗ | **has_conflict:** ✗ | **is_generated_irrelevant:** ✗
Q: Are Quyujoq and Qaserdalu both located in the same country?
A: "yes"
gold_atom_ids: a$_4$, a$_{13}$ **supporting_atom_ids**: — **distractor_atom_ids**: a$_6$, a$_7$, a$_{15}$
relevant_atom_ids: a$_4$, a$_{13}$ **annotation_warnings:** (none)

q_1 **field_lookup** | **is_irrelevant:** ✗ | **has_conflict:** ✗ | **is_generated_irrelevant:** ✓ (*generated as decoy, then rescued by Stage-2*)
Q: What is the local area code used for telephone numbers within Ascension Island?
A: —
gold_atom_ids: a$_0$ **supporting_atom_ids**: a$_1$ **distractor_atom_ids**: a$_2$
relevant_atom_ids: a$_0$, a$_1$ **annotation_warnings:** (none)

q_2 **irrelevant** | **is_irrelevant:** ✓ | **has_conflict:** ✗ | **is_generated_irrelevant:** ✓
Q: What are the names of specific wineries located in the Limestone Coast wine region?
A: —
gold_atom_ids: — **supporting_atom_ids**: — **distractor_atom_ids**: —
relevant_atom_ids: — **annotation_warnings:** (none)

annotation_meta: mode=batch2step, decomposition_parse_status=ok, question_parse_status=ok, num_atoms=17, num_questions=3, num_irrelevant_generated=2, warnings=**(none)** — all atom spans pass the substring check and no Stage-2 question is reclassified.

Figure 6: **(continued, Part 3 of 3)** Atoms a_{10}–a_{16} and the three Stage-2 questions. q_0 is a comparison question with two gold atoms (a_4, a_{13} — the location of the two villages) and three explicit Polish "X is part of the name of two villages" distractors (a_6, a_7, a_{15}) that the router must *not* activate. q_1 is a "rescued" field-lookup query: it was generated as an intended-irrelevant decoy (`is_generated_irrelevant`=✓), but Stage-2 cross-validation recognised that a_0 (country code +247) does answer it and flipped `is_irrelevant` back to ✗, while marking the related-but-non-answering a_2 (+290 vs. St Helena) as a distractor. q_2 is a genuine irrelevant query about wineries. The `annotation_meta` is clean — no Stage-1 substring failures and no Stage-2 type cross-validation warnings.

Table 5: Hyperparameter configuration of **Doc2Atom** on both base LLMs. Values are read row by row: the middle column is the gemma-2-2b-it run, the right column is the Qwen3-4B-Instruct-2507 run.

Component	**Gemma-2-2B-It**	**Qwen3-4B-Instruct-2507**
Base model (frozen)		
Identifier	`google/gemma-2-2b-it`	`Qwen/Qwen3-4B-Instruct-2507`
Hidden size d	2304	2560
Decoder layers	26	36
Attention / KV heads	8 / 4	32 / 8
Head dim	256	128
Position cap	8192	32768
Weight dtype	`bfloat16`	`bfloat16`
Attention impl.	FlashAttention-2	FlashAttention-2
Atom encoder (shared between atom and query)		
Backbone (frozen)	first 4 layers of base LLM	first 4 layers of base LLM
Encoder dtype	`bfloat16`	`bfloat16`
Pooling	mean over attended tokens	mean over attended tokens
Projection head	LN $\rightarrow$ Lin(2304$\rightarrow$256) $\rightarrow$ GELU $\rightarrow$ Lin(256$\rightarrow$256)	LN $\rightarrow$ Lin(2560$\rightarrow$256) $\rightarrow$ GELU $\rightarrow$ Lin(256$\rightarrow$256)
Output dim	256	256
Memory compiler		
Trunk	LN $\rightarrow$ Lin(256$\rightarrow$256) $\rightarrow$ GELU $\rightarrow$ Lin(256$\rightarrow$256)	LN $\rightarrow$ Lin(256$\rightarrow$256) $\rightarrow$ GELU $\rightarrow$ Lin(256$\rightarrow$256)
Latent / key dim	256	256
LoRA rank r / scale α	8 / 16.0	8 / 16.0
Memory layers	last 4 blocks (indices 22–25)	last 4 blocks (indices 32–35)
Target modules	`q_proj`, `v_proj`, `o_proj`, `down_proj`	`q_proj`, `v_proj`, `o_proj`, `down_proj`
LoRA init	$A \sim \mathcal{N}(0, 0.02^2)$, B=0	$A \sim \mathcal{N}(0, 0.02^2)$, B=0
Two-stage router		
Stage-1 retrieval	cosine over L_2-normalised keys	cosine over L_2-normalised keys
Stage-1 top-K_1	32	32
Stage-2 reranker	`BAAI/bge-reranker-base`	`BAAI/bge-reranker-base`
Stage-2 max length	192 tokens	192 tokens
Rerank fusion weight α_r	1.0	1.0
Metadata bias	4 learned scalars	4 learned scalars
Final selection	softmax-weighted top-k (curriculum)	softmax-weighted top-k (curriculum)
Routing temperature	1.0	1.0
Composer softmax	on	on
Optimization		
Optimiser	AdamW (β_1=0.9, β_2=0.999)	AdamW (β_1=0.9, β_2=0.999)
Weight decay	0.01	0.01
Learning rate	1.0×10^{-4}	8.0×10^{-5}
LR schedule	cosine	cosine
Warm-up	500 steps	500 steps
Hardware & data loading		
GPUs	8$\times$ NVIDIA A100 80 GB	8$\times$ NVIDIA A100 80 GB
Distributed	DDP, `find_unused_parameters=true`	DDP, `find_unused_parameters=true`
Dataloader workers / prefetch / persistent	8 per rank / 4 / on	8 per rank / 4 / on