FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Graph-Augmented Conversational Memory for Context-Efficient LLM Agents

Eduardo Martins Oliveira

Mestrado em Inteligência Artificial

Supervisor: Prof. Davide Carneiro

Second Supervisor: Prof. Pedro Ferreira

June 2026

© Eduardo Martins Oliveira, 2026

Resumo

Os agentes baseados em modelos de linguagem de grande dimensão (Large Language Models, LLMs) que operam sobre conversas longas têm de decidir quais as partes do histórico de interação a manter dentro de uma janela de contexto limitada. As abordagens convencionais baseiam-se na recência, preservando as interações mais recentes, ou em recuperação semântica, selecionando interações passadas semanticamente semelhantes à consulta atual com base em embeddings. No entanto, a relevância conversacional nem sempre é capturada apenas pela proximidade temporal ou pela similaridade semântica. Este problema é particularmente relevante para modelos de linguagem de menor dimensão (Small Language Models, SLMs), que operam sob restrições rigorosas de contexto e capacidade de raciocínio limitada. Estes modelos são também importantes para organizações que pretendem executar sistemas conversacionais na sua própria infraestrutura, reduzindo a dependência de grandes fornecedores externos e mantendo informação sensível sob o seu próprio controlo.

Esta dissertação investiga uma arquitetura de memória aumentada por grafos (graph-augmented memory), na qual cada interação humano-modelo é representada como um nó num grafo conversacional persistente, sendo as interações estruturalmente relacionadas ligadas por relações discursivas tipadas (typed discourse relations), como especialização, generalização e continuidade temática entre tópicos do mesmo nível. Durante a inferência, o agente constrói o seu contexto dentro de um orçamento fixo de tokens através da combinação de interações recentes, recuperação semântica e expansão baseada em grafos a partir de pontos de entrada obtidos por recuperação semântica. Pretende-se, assim, recuperar interações passadas estruturalmente relacionadas que uma janela deslizante (sliding window) ou uma pesquisa semântica, por si só, dificilmente conseguiriam identificar. A arquitetura é avaliada utilizando o benchmark LoCoMo para memória conversacional de longo prazo, sendo comparada com uma baseline de janela deslizante e com uma baseline de recuperação semântica. Este estudo analisa se a estrutura discursiva baseada em grafos pode fornecer um sinal adicional de recuperação para memória conversacional eficiente em termos de contexto, estabelecendo uma base para desenvolvimentos futuros.

Palavras-chave: Memória Conversacional • Large Language Models • Retrieval-Augmented Generation • Grafos de Conhecimento

Abstract

Large Language Model (LLM) agents that operate over long conversations must decide which parts of the interaction history to keep within a limited context window. Standard approaches rely on recency, by keeping the most recent turns, or on semantic retrieval, by selecting past interactions that are embedding-similar to the current query. However, conversational relevance is not always captured by temporal proximity or semantic similarity alone. This problem is especially pressing for small language models (SLMs), which work under tight context budgets and limited reasoning capacity. Such models also matter for organisations that wish to run conversational systems on their own infrastructure, reducing their dependence on large external providers and keeping sensitive information under their own control.

This dissertation investigates a graph-augmented memory architecture in which each human-model interaction is represented as a node in a persistent conversational graph, and typed discourse relations, such as specialisation, generalisation, and same-level thematic continuity, link structurally related interactions. At inference time, the agent builds its context within a fixed token budget by combining recent turns, semantic retrieval, and graph-based expansion from semantically retrieved entry points. The aim is to surface structurally related past interactions that a sliding window or semantic search alone might miss. The architecture is evaluated on the LoCoMo benchmark for long-term conversational memory, against a sliding-window baseline and a semantic retrieval baseline. The study examines whether graph-based discourse structure can provide an additional retrieval signal for context-efficient conversational memory, and lays a foundation for future development.

Keywords: Conversational Memory • Large Language Models • Retrieval-Augmented Generation • Knowledge Graphs

The United Nations Sustainable Development Goals

The United Nations Sustainable Development Goalss (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals (https://sdgs.un.org/goals) to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

Beyond its core focus on conversational memory, this work carries broader sustainability implications. By assembling an agent's context within a fixed token budget, the proposed graph-augmented memory architecture bounds the computational resources consumed per inference and supports the use of Small Language Models (SLMs). Capable long-term conversational systems thus become viable on modest, self-hosted infrastructure, which encourages the responsible use of computational resources, reduces the energy footprint of conversational Artificial Intelligence (AI), and widens access to such systems for organisations that cannot rely on large external providers or that must keep sensitive information under their own control.

The specific Sustainable Development Goals addressed have the following names:

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 12 Ensure sustainable consumption and production patterns

SDG 13 Take urgent action to combat climate change and its impacts

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.5	Advances scientific research in artificial intelligence by proposing a graph-augmented conversational memory architecture and evaluating it against sliding-window and semantic-retrieval baselines on the LoCoMo benchmark.	Novel architecture proposed; number of baselines compared; reproducible benchmark results.
	9.c	Enables organisations to run conversational agents on their own infrastructure using SLMs, reducing dependence on large external providers and keeping sensitive data under their own control.	Task accuracy achieved with SLMs under fixed context budgets; performance across model sizes . . .
12	12.2	Operates under a fixed token budget, bounding the context, and thus the compute and memory, consumed per inference; selective graph-based retrieval avoids reprocessing entire conversation histories.	Tokens consumed per query relative to the context budget; accuracy retained at reduced budgets versus baselines . . .
13	13.2	Lowers the energy and carbon footprint of conversational AI by enabling capable long-term memory with bounded context and smaller models instead of ever-larger context windows.	Estimated reduction in compute and energy per query relative to full-history or larger-model baselines . . .

Acknowledgements

I would like to express my sincere gratitude to everyone who, directly or indirectly, contributed to the completion of this dissertation.

My deepest thanks go to my supervisors, Davide Carneiro and Pedro Ferreira, for their guidance, availability, and constant encouragement throughout this work. Their scientific insight and the knowledge they generously shared were essential at every stage of this project, and I am truly grateful for their patience and support.

I am also grateful to the Faculty of Engineering (FEUP) and the Faculty of Sciences (FCUP) of the University of Porto, as well as to all the professors whose teaching shaped my academic path and prepared me for this challenge.

To my colleagues and friends, thank you for the companionship, the shared moments, and the support that made this journey lighter, both in and out of the academic setting.

Finally, my most heartfelt thanks go to my family, for their unconditional support, their patience, and for always believing in me.

Eduardo Martins Oliveira

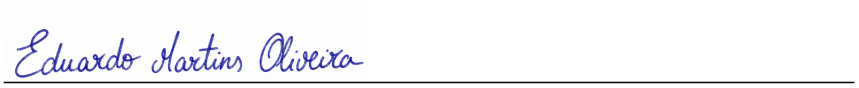
Declaration of honour

I declare that this dissertation is my own work and has not previously been submitted or assessed at this or any other institution. References to other authors (statements, ideas, thoughts), as well as the use of published works, strictly comply with the rules of attribution and are duly identified in the text and in the bibliographic references, in accordance with the applicable referencing standards. I am aware that the practice of plagiarism or self-plagiarism constitutes an academic offence, as established in the U.Porto Code of Ethics.

I further declare that I have included text(s) published in co-authorship, and that I have specified, with transparency and rigor, the contribution of each co-author, as well as stated whether there exists any other dissertation in which the same text(s) may also have been included. The authorization from the journal in which the work is published is also included, together with the express agreement signed by all co-authors authorizing the inclusion of the article.

I also declare that I have used Artificial Intelligence tools to complete part(s) of this dissertation, and that this use and its results are duly noted and have been carefully checked for errors, inaccuracies, unfounded attributions or other forms of bias in content and arguments, as well as determining authorship.

Eduardo Martins Oliveira



Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
SDG	Sustainable Development Goals
RAG	Retrieval-Augmented Generation
LLM	Large Language Model
SLM	Small Language Model

Chapter 1

Introduction

1.1 Background and Motivation

Large Language Models (LLMs) are increasingly used as the reasoning core of conversational agents, decision-support systems, and workflow automation tools. In specialised domains such as legal, medical, technical, or regulatory work, these agents do not answer isolated questions; they sustain long interactions in which a user introduces a topic, returns to it later at a different level of detail, and connects new questions to earlier exchanges. Maintaining a coherent conversation in this setting requires the agent to preserve relevant details across many turns and to recover previously stated information when it becomes relevant again.

This requirement is difficult to meet in practice. Performance degrades markedly as conversations grow: a large-scale study across leading open- and closed-weight models reports an average drop of 39% between single-turn and multi-turn settings, with models frequently failing to recover from early incorrect assumptions [15]. As a discussion extends over many turns, users are forced to restate information they have already provided, and the model regularly fails to detect that a current query is related to an earlier, non-adjacent part of the conversation. The consequence is repeated explanation and a gradual loss of coherence.

The difficulty is most visible when conversational agents support work over large and interconnected technical documents, such as process manuals, standards, and engineering reports. This dissertation is motivated by an applied setting at INESC TEC (Institute for Systems and Computer Engineering, Technology and Science), where the volume and cross-referenced nature of such documents give rise to long, non-linear conversations: users move repeatedly between sections, ask follow-up questions at varying levels of granularity, and return to earlier points after exploring others. This interaction pattern intensifies the context-retention challenge and frames the problem addressed here.

These challenges are sharpened by the kind of model an organisation can realistically deploy. Many practical settings favour Small Language Models (SLMs) over large proprietary services accessed through external Application Programming Interfaces (APIs) [30]. Beyond cost and latency, this preference reflects two further concerns. The first is resilience: dependence on a small

number of large, non-European providers has been argued to increase systemic fragility, so that improving the capabilities of locally deployable models broadens the range of viable alternatives [16]. The second is digital sovereignty: when sensitive information and operational knowledge must remain under organisational control, running models and their supporting memory on owned infrastructure becomes a requirement rather than an option [22].

Conversational memory mechanisms suited to small, locally deployable models are therefore an enabling component of this operating regime, not merely a technical convenience. The architecture studied in this dissertation does not address resilience or sovereignty directly; it targets the regime in which those concerns make small, local models the preferred choice.

1.2 Context Windows and Small Language Models

The technical root of the context-retention problem is the finite context window of a language model: the maximum number of tokens it can process at once. The most direct strategy, passing the entire interaction history to the model at every turn, is therefore bounded by this limit, and is in any case inefficient and costly, since it spends budget on text that is often irrelevant to the current query [26].

Recent models have expanded their stated context windows considerably, and open-source offerings have begun to converge with proprietary ones in advertised capacity, as illustrated in Figure 1.1 [4]. This convergence, however, applies mainly to large frontier models. In resource-constrained, locally hosted deployments, the small-to-medium open-source models that are actually used operate with far smaller windows [3, 29]. Even when such a window is large enough for a single document, it is quickly exhausted by an extended conversation, at which point the model either exceeds its limit or silently discards earlier content.

A further gap separates a model's stated context window from its *effective* one. A growing body of evidence shows that models exploit only a fraction of their advertised capacity before accuracy degrades, and that performance can fall sharply well below the nominal limit [23]. This effective length is particularly limited for smaller models, whose architectural trade-offs favour inference speed and memory efficiency over long-context capability, and whose effective range is further constrained by the position distributions seen during pretraining [2]. Enlarging the context window therefore does not, on its own, solve the memory problem: useful information may be present in the prompt yet remain underused. The decisive factor is the selection of context rather than its volume.

This degradation is not confined to the boundary of a model's capacity; it accumulates as the input grows, even when the underlying task is held fixed. Controlled experiments on a simple retrieval task, in which a single relevant passage must be recovered from a larger body of otherwise unrelated text, show that accuracy declines steadily as the surrounding context lengthens [10]. Figure 1.2 summarises this effect across models grouped by context capacity and performance level: in every group, accuracy falls as input length increases. The decline is sharpened by distractors, passages that resemble the target but do not answer the question, and it worsens as their number

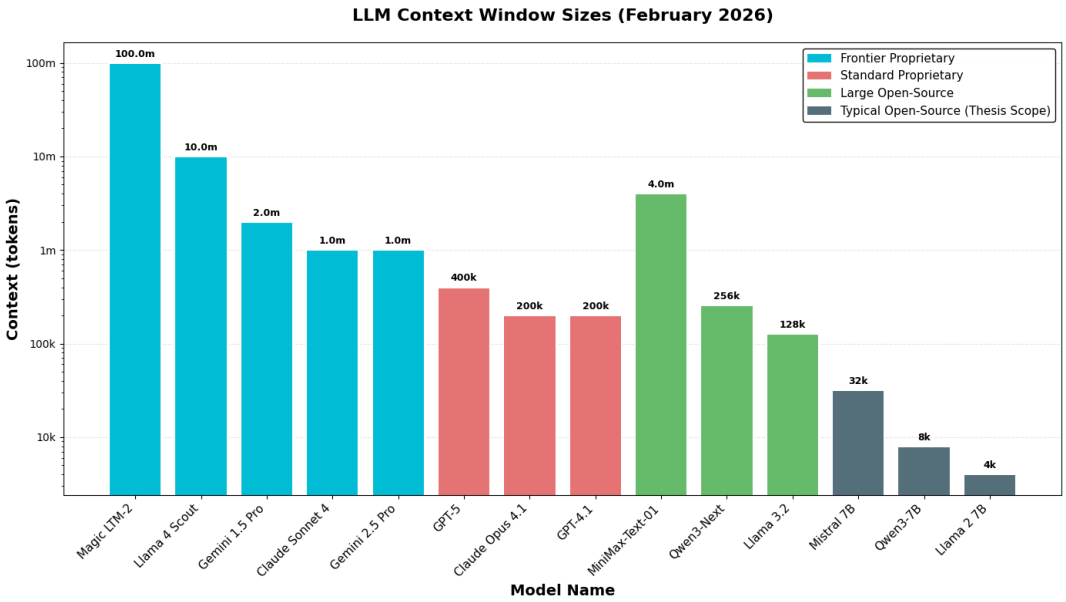


Figure 1.1: Comparison of stated LLM context-window sizes. Open-source models have converged with many proprietary offerings in advertised capacity, but the smaller open-source models typical of resource-constrained, locally hosted deployments, which are the operating regime targeted in this dissertation, remain comparatively constrained for long conversations.

rises, since longer inputs offer more room for such competing content. This pattern reinforces the central point of this dissertation: the value of a context window lies in what is placed within it, and filling it with more material, especially material that competes with the relevant content, tends to degrade performance rather than improve it.

1.3 Problem Statement

Two strategies dominate the selection of conversational context under a fixed budget. The first is a sliding window over the most recent turns. This preserves local continuity and supports immediate follow-up questions, but it fails whenever the relevant information lies further back in the conversation. The second is Retrieval-Augmented Generation (RAG), which extends the effective memory of a model by retrieving past interactions that are semantically similar to the current query [17]. Retrieval is more flexible than recency, but semantic similarity alone does not always capture conversational relevance: a past interaction may be decisive for the current question without being among the most embedding-similar, and the most similar fragments are not necessarily the most useful ones.

Neither signal accounts for the *structure* of the conversation. When a user specialises a previous topic, generalises from a specific case, or moves to a parallel thread, that relationship determines whether an earlier interaction is still relevant, yet it is invisible to both recency and semantic similarity. Adding more context is not a remedy, since models tend to underuse information depending on its position in the prompt [19]. The problem addressed in this dissertation is therefore

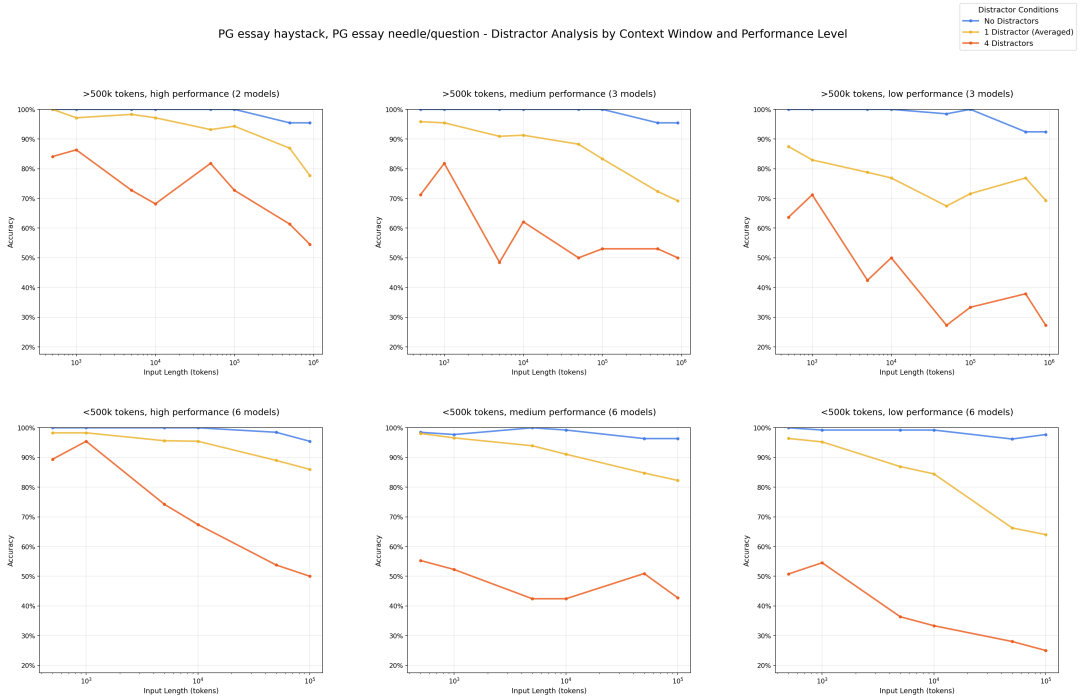


Figure 1.2: Retrieval accuracy as a function of input length, for models grouped by context capacity and performance level. A single relevant passage must be recovered from a longer body of text; accuracy declines as the input lengthens, even though the task is unchanged. The presence of distractors, passages similar to the target but not answering the question, amplifies the decline, and the effect grows with the number of distractors. Adapted from Hong, Troynikov, and Huber [10].

not to enlarge the context passed to the model, but to select *more useful* context within the *same* fixed token budget, in a way that is compatible with SLMs.

1.4 Proposed Approach

This dissertation investigates a graph-augmented conversational memory architecture that adds discourse structure as a third selection signal, alongside recency and semantic similarity. Rather than representing the conversation only as a chronological sequence, the system maintains a persistent graph in which each node corresponds to one complete human-model interaction, that is, a user query together with the response it received. Typed discourse edges connect structurally related interactions through three relations: specialisation (a node that refines or deepens an earlier topic), generalisation (a node that abstracts it), and same-level thematic continuity (parallel topics at a similar level of detail). The graph is built incrementally as the conversation unfolds, so that each new interaction is linked to the existing structure without recomputing it from scratch.

At each turn, the agent assembles its context within a fixed token budget from three complementary sources: the most recent interactions, the interactions most semantically similar to the current query, and a graph-based expansion that follows discourse edges outward from those semantic entry points. The expansion is what distinguishes the approach from standard semantic retrieval: it can recover interactions that are structurally related to a retrieved entry point but not themselves similar enough to be retrieved directly. The aim is qualitative, namely to improve which interactions populate a fixed budget. The architecture is described in full in Chapter 3.

1.5 Research Objectives and Questions

The principal objective of this dissertation is to design, implement, and empirically evaluate a graph-augmented conversational memory architecture for SLMs operating under fixed context budgets. The evaluation isolates the contribution of the graph from those of recency and semantic retrieval, so as to determine whether conversational discourse structure can be turned into a usable retrieval signal. The work is guided by the following principal research question:

> **RQ.** For small language models operating under a fixed token budget, can conversational discourse structure, represented as a graph of typed relations between past interactions, be turned into a usable signal for selecting context?

The investigation seeks to establish where a graph-based memory layer helps, where it is neutral, and why, rather than to claim a finished system. The work is intended as a foundation for the future development of richer conversational-memory architectures.

1.6 Contributions

This dissertation makes the following contributions:

- A graph-augmented conversational memory architecture in which interactions are represented as nodes of a persistent graph connected by typed discourse relations, and in which context is assembled under a fixed token budget by combining recency, semantic retrieval, and graph-based expansion. The graph is constructed incrementally under the same sequential conditions as a deployed agent.
- A fair and reproducible evaluation methodology on the LoCoMo benchmark for long-term conversational memory, in which the proposed approach and two baselines, the sliding window and semantic retrieval, are compared under identical conversations, questions, models, and budgets, so that any difference reflects the context-selection strategy alone and isolates the contribution of the graph.
- A controlled empirical investigation of when and why discourse structure acts as a useful retrieval signal under tight budgets, providing a baseline and a methodology on which more advanced, semantically richer conversational-memory layers can be built.

1.7 Dissertation Structure

The remainder of this dissertation is organised as follows. Chapter 2 reviews the state of the art on structural and relational dialogue modelling, dynamic graph construction, external knowledge integration, and graph-based context retrieval, and positions the proposed architecture with respect to existing work. Chapter 3 presents the methodology: the memory representation, the turn-processing pipeline, context construction under a fixed budget, graph expansion, the incremental memory-update rule, and the experimental design, including the LoCoMo benchmark, the methods compared, the models, the evaluation metrics, and the selection of the operating parameters. Chapter 4 reports and discusses the results, analysing overall and per-category performance and the mechanisms behind the observed behaviour. Chapter 5 summarises the findings, revisits the research objectives, discusses the limitations of the study, and outlines directions for future work.

Chapter 2

State of the Art

This chapter reviews the literature on graph-based representations of dialogue and its bearing on the problem of retrieving relevant context from a growing conversation history. Section 2.2 examines how a dialogue is turned into a graph, and Section 2.3 how that graph is updated as the conversation grows. Section 2.4 covers a related strand in which graphs hold external knowledge rather than the dialogue itself, and Section 2.5 examines how a graph is searched for relevant context. The limitations common to these methods recur throughout; Section 2.6 collects them and sets out where the architecture proposed in this thesis stands.

2.1 Search Strategy

The review was conducted through a targeted search on Web of Science. The query combines three groups of terms: graph structures (dynamic, hierarchical, knowledge, or context graphs), dialogue and conversational agents, and explicit memory or context-preservation mechanisms. Recurrent architectures predating the graph-based literature were excluded, along with unrelated tasks such as recommendation, single-turn question answering, and text generation without a memory component. Table 2.1 gives the exact query.

Table 2.1: Web of Science search query used for the literature review.

("knowledge graph" OR "personal knowledge graph" OR "dynamic knowledge graph" OR "temporal knowledge graph" OR "conversation graph" OR "dialogue graph" OR "context graph" OR "interaction graph" OR "hierarchical graph") AND (dialogue OR conversational OR chatbot OR "conversational agent" OR "dialogue system") AND (memory OR "long-term user modeling" OR "user model" OR "interaction history" OR "persistent context" OR "conversation history" OR "dialogue state" OR "context preservation" OR "relation prediction" OR "graph evolution" OR "context retrieval" OR "graph-based memory" OR "follow-up interaction" OR "hierarchical context") NOT (LSTM OR "long short-term memory" OR RNN OR "recurrent neural" OR recommendation OR recommender OR "question answering" OR QA OR generation OR generate)

The query returned 55 results. Restricting the publication window to 2020–2026 left 41 papers, which were screened by hand for works that use graph structures to model dialogue context, interaction history, or dynamic knowledge integration. Discarding papers concerned only with single-turn question answering or static generation left 22 core papers. A further five papers, found by manual tracing of references and recent venues, were added to cover architectures directly relevant to the research objectives, giving a final corpus of 27.

The corpus divides into four categories that follow the lifecycle of a conversational graph, from representation through evolution to use:

- **Structural and Relational Dialogue Modeling**: approaches that abandon the linear view of dialogue and encode its topology with trees, discourse parsers, or hierarchical dependencies.
- **Dynamic Graph Construction**: methods concerned with how the graph evolves, adding or updating nodes and edges as the conversation proceeds.
- **External Knowledge and Grounding**: works in which the graph carries external world knowledge rather than conversational episodic memory.
- **Graph-Based Context Retrieval and Reasoning**: techniques for the downstream problem of choosing which part of the graph to read under a context-window limit.

2.2 Structural and Relational Dialogue Modeling

Conventional dialogue systems read a conversation as a linear sequence of utterances and rely on recurrent or transformer encoders to relate adjacent turns. This works when meaning tracks turn order, but in extended discussions it frequently does not: as set out in Chapter 1, speakers reopen earlier topics, pursue sub-topics to different depths, and tie new questions to non-adjacent exchanges. To expose those connections, a growing body of work represents dialogue as a graph whose nodes are dialogue elements and whose edges record the relations between them.

The granularity of the node is the main axis of variation. At the finest level, some systems exploit structure inside an utterance. One derives dependency and constituent trees per utterance and merges them into a single graph of words and phrases linked by syntactic relations, where a graph attention network raises slot-value extraction on MultiWOZ by 5.32% to 6.08% [18]. Another parses meeting transcripts into 16 discourse relation types and promotes both utterances and relations to nodes, with graph convolution over the result improving summarization ROUGE by 1.7 to 4.4 points over sequential baselines [8]. Going lower still, entity-mention graphs take individual mentions as nodes joined by local co-occurrence [20], and token graphs learn their structure from key predicates while pruning edges during training [7]. The token-level designs scale quadratically with dialogue length, which limits their use on long conversations.

At a coarser level, other systems keep utterances whole and model the links between turns. A sparse graph attention network treats each utterance as a node and prunes connections layer by layer, keeping only the most relevant neighbours so that distant turns remain reachable without

the noise of the full history [33]. A different design clusters utterances by intention and uses inter-cluster transition probabilities as edges, predicting the next intention more accurately than text-only encoders [14]. Coarser again, dialogue-state graphs let a node stand for a complete belief state and dialogue act; merging identical states across training dialogues makes conversations that reach the same state share a node, with action-labelled transitions weighted by training frequency, an offline data-augmentation method that lifts policy success by up to 6.4% [9].

One result holds across this range of designs: explicit structure beats purely sequential processing, because graph connectivity reaches non-adjacent context that a linear encoder cannot. The spread of granularities is itself revealing, however. Existing nodes sit either below the utterance, on words and syntactic relations, or well above it, collapsing several exchanges into a state or an intention cluster. The query–response interaction, the natural unit of a question and the answer it received, is absent as a first-class element: a token graph splits it into words, a state graph may fold an entire booking into one node. This mismatch makes it hard to capture the topic and sub-topic hierarchy of an extended technical discussion.

Two further properties recur and matter for the present work. First, most of these graphs are not persistent: they are pre-computed from a finished dialogue or rebuilt each turn, and the few that update online adjust weights or remove edges rather than growing the structure as the conversation lengthens. Second, the edge vocabulary is syntactic, temporal, speaker-based, or entity-based, all useful for slot filling and relation extraction but silent on the rhetorical move a turn makes. When a user asks for more detail, generalises from a case, or opens a parallel thread, no edge type in these taxonomies records it, even though that move is what decides whether an earlier interaction is still in play.

2.3 Dynamic Graph Construction

If the previous section concerned what the graph contains, this one concerns when it is built. A conversation's context is temporal: entities enter, topics turn over, and arguments develop, so a graph that is fixed in advance or discarded each turn cannot follow it. A line of work therefore constructs the graph incrementally, turn by turn.

The clearest illustration comes from conversational semantic parsing, where context is assembled as knowledge-graph subgraphs extracted from Wikidata and merged across turns through entity linking, improving parsing accuracy by 22% over sequence baselines [12]. Notably, accuracy peaks when only the previous five turns are retained, an early sign that incremental construction must trade historical coverage against relevance. Other methods carry reasoning state forward rather than just structure. FlowDelta builds word-level graphs over context words and processes them recurrently across turns; in ablation the temporal links alone account for 6.1% in F1, so the conversation flow is doing real work [6]. SocAoG instead parses social structure probabilistically, modelling the network as an And-Or graph that Markov Chain Monte Carlo sampling edits one proposal at a time, accepting or rejecting edge additions and attribute updates against the dialogue, for a 2.4% F1 gain over static parsing [24]. GraphWOZ keeps an entity knowledge graph

that is transformed at every utterance, adding nodes and linking mentions through graph-distance features [28].

These methods agree that updating the graph as the dialogue proceeds outperforms rebuilding it, and the computational case agrees with them, since a from-scratch rebuild re-derives structure that an incremental update could have kept, provided the update itself stays cheap. What they share beyond dynamism, though, is their unit of change. They evolve entities, words, or dialogue states; the relation between two full exchanges is never the thing being updated, so a follow-up that specialises an earlier turn registers, if at all, only as a shift in entity co-occurrence. Most also lean on an external knowledge base to fix what a node is and how nodes connect, which ties them to domains where Wikidata or a hand-built schema exists. In an open-domain technical conversation the relevant knowledge may have no structured form, and coherence is governed by the shape of the discussion rather than by entity relations.

Scaling is the open question for this group. Re-embedding every node by message passing, running MCMC, or merging large knowledge-base subgraphs grows more costly as the dialogue extends, and the reported gains come from dialogues of roughly 10 to 20 turns. None of the systems prunes, compresses, or updates selectively, so their behaviour past 50 to 100 turns is untested.

2.4 External Knowledge and Grounding

The two preceding sections model the dialogue itself. A separate tradition uses the graph for something else entirely: to bring external world knowledge into the conversation. Here the graph is a knowledge base, such as ConceptNet, Wikidata, or a domain ontology, and dialogue understanding or generation is grounded in it. This work addresses a different problem from conversational memory, but it shows what graph reasoning buys a dialogue system and helps delimit the scope of this thesis.

Some systems read from such a base at inference time. To steer a conversation toward a target concept, one method retrieves multi-hop paths between mentioned concepts and the goal and predicts new relations to bridge gaps in ConceptNet, with the dynamic completion paying off exactly on the hard cases where no path exists, reaching 14.08 BLEU against 11.45 for baselines [27]. The graph here describes how concepts relate in the world, not how the conversation is organised. Other systems run in the opposite direction, writing structured knowledge extracted from dialogue into a base. MemoryGraph stores entity–relationship triples from user utterances in Neo4j as long-term chatbot memory; its user study surfaces a tension worth noting, that retrieval makes the agent seem more capable yet raises trust and privacy concerns once users see what has been retained [32]. Task-oriented variants fill a pre-defined ontology of entities such as employees, projects, and tasks, gaining validation and multi-threaded discussion at the price of a schema that must be designed by hand for each domain [11].

Across both directions, graph-structured knowledge clearly helps tasks that need factual grounding, commonsense inference, or domain expertise, by supporting multi-hop reasoning and reusing

existing semantic resources. The architecture it produces, however, models the world and not the conversation. A system of this kind may store that a user mentioned learning languages at the library as connected concepts, yet it has no way to record that the library was raised as a refinement of an earlier question about educational resources, or that the user came back to it after a detour through online courses. The dependence on structured resources is the other constraint: ConceptNet methods inherit its coverage limits, ontology methods need their schema built first, and in specialised technical domains, where no such base exists, there is little for grounding to attach to. Even systems labelled "dynamic" tend to assemble a subgraph per query and drop it, and where they handle time they handle the validity of facts, not the order in which a discussion moved between topics.

These observations place the present work next to this tradition rather than against it. External knowledge graphs supply facts about the world; the conversation graph studied here is about discourse structure. A complete system could use both, grounding factual content with one and deciding which history is still relevant with the other. This thesis takes up the second, less-explored problem.

2.5 Graph-Based Context Retrieval and Reasoning

Building and maintaining a graph still leaves the operative question open: when the history outruns the context window, which prior interactions should be read? This section reviews the retrieval and reasoning mechanisms that answer it, and the question grows sharper with length, since pushing every node through message passing or attention is soon both too expensive and too redundant to sustain.

One family filters the graph. Progressive sparsification keeps, at each layer of a graph attention network, only the neighbours above a learned threshold, ending with 30 to 40 percent of the original edges and lower cost at higher accuracy [33]. A leaner variant scores every historical utterance against the current response with BERT embeddings and keeps the top twelve "pivot turns", for a 0.4 to 1.0 point gain that plateaus once more than twelve are kept, evidence of a real coverage-versus-noise trade-off [34]. A second family traverses instead of filters. Biased random walks treat the dialogue as a directed tree and bias sampling toward parent nodes, accumulating ancestor context while skipping siblings and distant descendants; with tuned parameters this adds 3 to 6 F1 points, and the error analysis confirms that upstream context exposes patterns invisible in an isolated utterance [1]. A third family propagates over dense, typed graphs: stacked graph-attention layers with speaker- and entity-aware edges reason one hop further per layer, optimal at three layers before over-smoothing, and most valuable for turns ten or more apart, where sequential models lose the thread [31]. The recurring cost is that even sparse adjacency grows quickly, so exhaustive propagation scales poorly.

More recent pipelines combine signals and rerank. A heterogeneous graph of sessions, discourse units, and event arguments first proposes candidates by dense retrieval, then applies a recall-biased language-model relevance filter, and finally runs Personalized PageRank from the

surviving seeds through synonym edges between arguments; it reaches 78 percent accuracy while compressing context from over 23,000 tokens to under 1,000, with argument-level edges alone worth 13 points on multi-hop reasoning [35]. A fragment-then-compose method decomposes history into propositions with normalised entities and predicates and retrieves those whose argument overlap with the query clears an embedding-similarity threshold, improving accuracy by 7 to 12 points over summary baselines at 24 times less context; the threshold is delicate, since below 0.7 weak matches flood the context and above 0.9 valid ones are dropped [13]. A hybrid system fuses semantic similarity, BM25, and breadth-first graph search by reciprocal rank fusion, then reranks with maximal marginal relevance and a cross-encoder, for 94.8 percent on memory retrieval; the per-method breakdown is the instructive part, with semantic search handling abstract queries, lexical search catching specific terminology, and graph traversal recovering connected but semantically distant items [25].

Despite their variety, these mechanisms share limitations that bear on extended technical conversations under tight budgets. Their retrieval granularity tends to miss the natural unit of dialogue: word- and mention-level retrieval recovers detail but discards the exchange that gives it meaning, while session- and segment-level retrieval drags in irrelevant material and wastes the budget. None selects or omits a complete user–system exchange as a unit. Their relevance signal is also shallow. Cosine similarity cannot tell a past interaction that merely provides background from one that addresses a subcase of the current query, and BM25 sees term overlap but not whether a match generalises, specialises, or contradicts. Even the graph-native methods walk edges that encode co-occurrence, temporal proximity, or coreference, so when a discussion returns to a topic after several subcases they cannot separate what is still open from what has been resolved. The biased-walk result is suggestive here, showing that a parent makes better context than a sibling [1], but its bias runs over reply links, not over the abstraction relations between topics; a question on authentication that follows exchanges on database security and network protocols relates to both, yet no reply chain records that authentication is a subcase of security.

Two final mismatches concern cost and objective. Pipelines that follow dense retrieval with language-model reranking spend several expensive inference calls per turn [35], which is acceptable for benchmarks run on frontier models but prohibitive for the 7B to 13B open models where context limits bite hardest, and which presume a reasoning quality those models often lack. The objective differs too. Most of these systems optimise question-answering accuracy, locating the fact that answers a discrete query; the 94.8 percent figure above measures whether the right fact is present, not whether the retrieved context sustains coherent continuation [25]. When a discussion resumes a partially resolved topic, the target is the interaction history that establishes what is settled and what remains open, which propositional retrieval undercuts by shedding precisely the rhetorical structure it fragments away [13].

2.6 Positioning

The four bodies of work above converge on a common shortfall, which this thesis takes as its starting point. Existing graphs operate either below the level of a complete utterance, focusing on words, entities, or syntactic structures, or above it, grouping multiple exchanges into states, intentions, or dialogue segments. The query–response interaction never appears as a unit of representation or retrieval. Their edges record syntax, time, speakers, entities, or knowledge-base relations, none of which captures the rhetorical move that decides whether an earlier exchange is still relevant. Their structures are largely transient or rebuilt in full, and their retrieval either propagates over the whole graph or selects by surface similarity, with the strongest pipelines leaning on reranking too costly for small models.

The architecture proposed in this dissertation, and detailed in Chapter 3, is positioned against each of these points. It represents memory at interaction granularity, one node per query–response pair. It connects nodes with typed discourse edges, specialisation, generalisation, and same-level continuity, inferred from the conversation rather than an external base. The graph is persistent and grows incrementally, each new interaction attached to existing structure without reprocessing the whole. Retrieval follows these edges outward from semantic entry points, recovering structurally relevant exchanges that surface similarity would miss, and it does so without exhaustive scoring or multi-stage filtering. The goal, consistent with Chapter 1, is to choose more useful context within a fixed budget for SLMs, complementing the external-knowledge line rather than replacing it.

Chapter 3

Methodology

This chapter describes the graph-augmented conversational memory architecture proposed in this thesis and the experimental procedure used to evaluate it. The central problem is unchanged from the one motivated in Chapter 1 and situated in the literature in Chapter 2: when a conversation grows past what a language model can hold in its context window, the system must decide which prior interactions to keep. Recency and semantic similarity are the two signals in common use; the architecture studied here adds a third, derived from the discourse structure of the conversation itself.

The chapter is organised in two parts. The first part presents the architecture: its design goals, the graph used to represent conversational memory, the per-turn processing pipeline, the procedure that assembles context under a fixed budget, the graph-expansion mechanism that distinguishes the proposed method from semantic retrieval, and the rule that updates memory after each interaction. The second part presents the experimental setup: the research design, the LoCoMo benchmark, the three context-selection strategies compared, the models and metrics, and the procedure used to select the operating point at which the system is evaluated. The chapter closes with the implementation details required for reproducibility.

Two concrete constraints shape every design decision that follows, and it is worth stating them at the outset. The first is a *scalability* constraint: the per-turn cost of maintaining memory must not grow linearly with the number of past interactions, so that conversations extending over several hundred turns remain tractable. The second is a *small-model* constraint: the architecture targets small open-source language models, in the 7–13B parameter range, and must operate within their limited context windows and latency budgets. These two constraints, rather than any appeal to large-model capacity, explain why careful context selection matters and why a structured memory layer is worth exploring.

3.1 Design Goals and Rationale

The architecture is designed for a specific operating regime, and its goals follow directly from that regime rather than from a general pursuit of accuracy. Three requirements define the design space.

Fixed, small context budget. The model receives a bounded amount of conversational context, far smaller than the full interaction history. This is the defining condition of the problem: if the entire conversation fit in the context window, no selection would be needed. Passing the full history is in any case inefficient, increases latency, and can degrade answer quality by surrounding the relevant evidence with distracting material [19]. The system must therefore choose a small subset of past interactions and use the budget well.

Bounded per-turn cost. The mechanism that maintains memory is invoked at every turn, so its cost must not scale with conversation length. A method whose per-turn work grows linearly with the number of stored interactions becomes progressively slower as the conversation extends, which is precisely the setting where long-term memory is supposed to help. The architecture therefore bounds the work performed at each turn by comparing a new interaction only against a fixed-size set of candidates, never against the whole history, and by deferring memory maintenance so that it does not lie on the path that produces the user-facing answer.

Compatibility with small models. The target deployment uses small open-source language models in the 7–13B range. Models of this class are increasingly favoured for their low inference latency and cost-effective operation, and for being well suited to resource-constrained settings where data is handled locally to preserve privacy and where domain knowledge can be acquired through lightweight fine-tuning [29]. These same properties, however, come with shorter context windows and more limited reasoning capacity than large proprietary systems: the models most attractive for on-device or self-hosted use, under strict latency constraints, are also the ones least able to recover from a long or poorly organised prompt. The 8B instruction-tuned model used throughout the evaluation is a deliberate instantiation of this requirement, not an incidental limitation. Designing for this class of model means that the memory layer cannot assume the generator will compensate for a poorly selected context; the selection itself has to be good.

Within this regime, recency and semantic similarity each capture part of what makes a past interaction relevant, but neither captures the structure of the discussion. Recency keeps the most recent turns and supports local follow-up questions, but it loses information stated earlier in a long conversation [19]. Semantic retrieval recovers interactions whose wording resembles the current query, but conversational relevance is not always a matter of surface similarity [5]: a useful earlier interaction may generalise the current topic, specialise it, or run parallel to it without sharing many terms. The hypothesis investigated in this thesis is that the *discourse structure* of a conversation, encoded as typed relations between interactions, provides an additional retrieval signal that complements recency and semantic similarity under a fixed budget.

The contribution is framed accordingly. The goal is not to increase the amount of context passed to the model, but to select more useful context within the same budget, and the architecture is presented as a first, controlled investigation of whether discourse structure can be turned into a usable retrieval signal, not as a finished system claimed to be superior to existing methods. Chapter 2 identified a consistent gap in prior work: existing conversational-memory methods

operate either below the level of the interaction, on tokens and entities, or above it, on sessions and dialogue states, and they connect memory items by co-occurrence, temporal order, or entity coreference rather than by rhetorical relation. The architecture described next responds to that gap directly, operating at interaction granularity over typed discourse edges.

3.2 Memory Representation

The agent maintains a persistent conversational memory in the form of a directed graph $G = (V, E)$. The defining choice is the granularity of the nodes: each node represents one complete *interaction*, that is, a user query together with the response the model produced for it. This is deliberately coarser than a token or entity and finer than a session or dialogue state. The interaction is the natural unit of a conversation, the level at which a topic is raised, refined, or revisited, and Chapter 2 argued that no reviewed method treats it as a first-class graph element.

Each node $v_i \in V$ stores three items: the question, the generated answer, and a vector embedding computed from the concatenated text of the question and the answer. The embedding is what makes the node retrievable by similarity; the stored question and answer together are what is later injected into context when the interaction is surfaced, as Section 3.4 explains.

Edges encode typed discourse relations between interactions. The architecture uses three relation types, each carrying a strength score that reflects how informative that relation is expected to be when expanding from a retrieved node:

- **subcase** — the connected interaction is *more specific* than the current one; it narrows the focus, adds detail, or treats a particular case of a broader topic.
- **supercase** — the connected interaction is *more general* than the current one; it broadens the scope or abstracts away from a particular case.
- **same_level** — the two interactions address parallel topics at a similar level of specificity.

A fourth outcome, `no_relation`, records the absence of any meaningful connection; such pairs are simply left unconnected and contribute no edge. Table 3.1 lists the three retained relation types and their strength scores. The scores are ordered to reflect a working assumption about discourse: a more specific elaboration of the current topic (subcase) is treated as the strongest signal, a generalisation (supercase) as slightly weaker, and a parallel topic (same_level) as weaker still. These scores enter the expansion ranking described in Section 3.5.

Representing interactions at this granularity, with rhetorically typed edges, lets the memory remain coherent across the kind of non-linear movement that characterises extended discussion. A user may introduce a general topic early in a conversation, return much later to a specific subcase, and then move to a parallel case in another session. In a linear transcript these three interactions may be far apart and easily separated by a budget that keeps only recent or only semantically similar turns. In the graph they remain directly connected, and the connection can be followed when any one of them is retrieved.

Table 3.1: Typed discourse relations between interactions and their strength scores. The `no_relation` outcome adds no edge.

Edge type	Meaning	Strength
`subcase`	Connected interaction is more specific	1.0
`supercase`	Connected interaction is more general	0.8
`same_level`	Parallel topics at similar specificity	0.6
`no_relation`	No meaningful connection (no edge)	—

3.3 Processing Pipeline

At each turn the system performs two phases in sequence: *context construction*, which runs before generation, and *memory update*, which runs after it. Figure 3.1 gives an overview of the complete pipeline.

In the first phase, the current query is used to assemble a prompt from a selected subset of the conversation. The query is embedded once; the embedding drives both semantic retrieval over past dialogue turns and the selection of entry points into the memory graph. The selected items are formatted, ordered, and combined with the query into the prompt that is sent to the generator, which produces the answer.

In the second phase, after the answer has been produced and returned, the new interaction is added to the memory graph and connected to existing nodes by typed edges. Placing the update after generation is a deliberate consequence of the bounded-cost requirement of Section 3.1: edge classification involves an additional language-model call, and performing it after the answer is returned keeps it off the path that determines user-facing latency. The update is also best-effort. If any step of it fails, the failure is caught and the conversation continues; a problem in memory maintenance never prevents the system from answering.

This separation has a second benefit. Because the graph is built incrementally as the conversation proceeds, and is never recomputed from the full history, the system operates under the same sequential conditions as a deployed agent. The memory available at turn t is exactly the memory that a live system would have accumulated by turn t, which makes the evaluation faithful to the intended deployment.

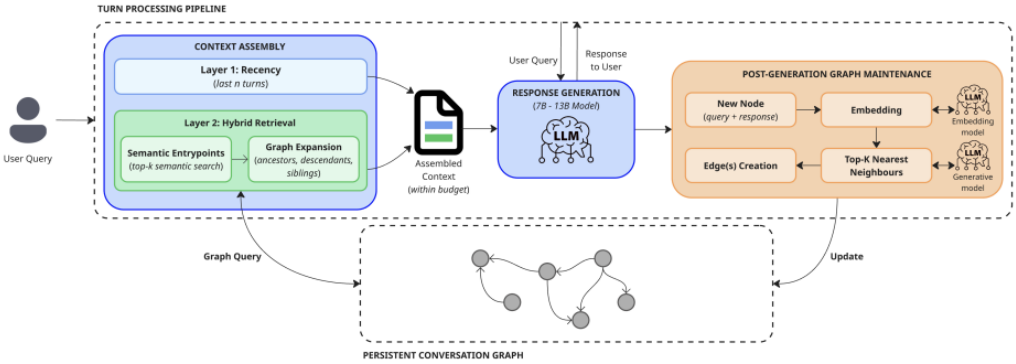


Figure 3.1: Pipeline overview of the system architecture.

3.4 Context Construction under a Fixed Budget

Context construction is governed by a strict budget that all three methods compared in this thesis share. The budget is expressed not in tokens but in a fixed number of context *items*, and this choice is central to the experimental design, so it is stated precisely here and revisited in Section 3.7.

3.4.1 The fixed-item budget

Every method presents the model exactly

$$T \;=\; N_{\text{recent}} + K_{\text{historical}} \;=\; 5 + 40 \;=\; 45 \text{ items}, \tag{3.1}$$

where an item is a single dialogue turn or, in the graph-augmented method, a single graph-derived entry. The methods differ only in *which* items they select, never in *how many*. Fixing the item count, rather than a token count, mirrors the fixed top-k retrieval used in the official LoCoMo evaluation and ensures that any difference in performance between methods is attributable to the selection strategy alone, which is the single variable under study. Token usage is therefore a non-binding safety cap rather than the active constraint; as Section 3.9 reports, the assembled prompts use only about a quarter of the nominal token budget in practice.

The budget is divided into two layers. The *recency layer* (Layer 1) always contains the last $N_{\text{recent}} = 5$ dialogue turns, preserving local continuity and supporting follow-up questions that depend on the immediately preceding exchanges. The *historical layer* (Layer 2) contains $K_{\text{historical}} = 40$ items selected from the older history; how these 40 slots are filled is exactly what distinguishes the methods.

3.4.2 Filling the historical layer

In a purely semantic strategy, the 40 historical slots are filled with the 40 past dialogue turns whose embeddings are most similar to the query, excluding any turn already present in the recency layer. The proposed graph-augmented strategy reserves part of the historical layer for graph-derived entries. Of the 40 slots, up to

$$G \;=\; \text{round}(K_{\text{historical}} \times f) \;=\; \text{round}(40 \times 0.25) \;=\; 10 \tag{3.2}$$

may be filled by interactions surfaced through graph expansion, with the remaining $40 - G_{\text{used}}$ slots filled by semantic retrieval, where $G_{\text{used}} \leq G$ is the number of graph entries actually available. The fraction f is the single parameter that controls the graph's presence in the context; its value of 0.25 is justified in Section 3.12.

This construction makes the relationship between the methods explicit. The graph entries *displace* semantic turns: every slot given to a graph-derived interaction is one fewer slot available for a semantically retrieved dialogue turn. When the graph has no usable neighbours to offer, which is necessarily the case early in a conversation before any edges have been formed, semantic retrieval

fills all 40 historical slots and the graph-augmented method reduces exactly to the semantic baseline. The graph can only ever change the contents of the historical layer by substitution, never by addition, which is what makes the comparison a controlled ablation of the graph signal.

The complete context combines the three sources,

$$C = C_{\text{recent}} \cup C_{\text{semantic}} \cup C_{\text{graph}}, \tag{3.3}$$

where C_{recent} is the recency layer, C_{semantic} the semantically retrieved historical turns, and C_{graph} the graph-expanded entries, subject to $|C_{\text{recent}}| = 5$ and $|C_{\text{semantic}}| + |C_{\text{graph}}| = 40$. For the two baselines, $C_{\text{graph}} = \varnothing$ and, in the sliding-window case, C_{semantic} is also empty, the historical layer being filled by recency instead.

3.4.3 Formatting and ordering

Dialogue turns are rendered one per line in a uniform format that exposes the session date,

`[<session date>] Speaker: text`,

so that the temporal structure of the conversation is visible to the generator; the same dates are later made available to the evaluation judge. Semantically retrieved turns are re-sorted into chronological order before they are placed in the prompt, so that the historical layer reads as a coherent sequence rather than a similarity-ranked list. The retrieved historical block and the recent block are presented under separate headings within the prompt, with the historical material first and the recent turns last.

Graph-derived entries are formatted differently and deliberately so. Each is injected as

`[related earlier topic] Q: <question> A: <answer>`,

that is, as the full earlier interaction, including both the question and the answer the model produced for it. The answer is included because the substance of a past interaction lies in its resolution, not in the question alone: surfacing only the question would tell the generator what was once asked but not what was established about it. Injecting the complete exchange gives the generator the actual content of the structurally related earlier *topic*, which is what makes a surfaced interaction useful as context rather than a bare pointer. The graph entries are placed after the semantic block in the assembled historical layer.

3.5 Graph Expansion

Graph expansion is the mechanism that distinguishes the proposed method from semantic retrieval, and it is invoked during context construction whenever the graph already contains nodes and at least one graph slot is available. Semantic retrieval finds interactions that are directly similar to the

query; expansion finds interactions that are *structurally related* to those, and which may be useful even though they are not similar enough to the query to be retrieved on their own.

The procedure has four steps. First, the query embedding is compared against the embeddings of the graph nodes, and the top $K_{\text{semantic}} = 5$ nodes by similarity are selected as *entry points*: these are the points at which the query touches the conversational graph. Second, for each entry point, the system collects its graph neighbours along the three typed edge classes. Third, each candidate neighbour is scored by a weighted combination of how similar it is to the query and how strong the discourse relation connecting it to its entry point is,

$$\text{score} \;=\; 0.75 \times \text{semantic\_sim} \;+\; 0.25 \times \text{edge\_strength}, \tag{3.4}$$

where semantic_sim is the cosine similarity between the neighbour and the query and edge_strength is the strength of the connecting edge from Table 3.1. Fourth, the neighbours are ranked by this combined score and the top G are taken, where G is the number of graph slots from Equation 3.2. Entry points themselves are excluded from the neighbour set so that expansion contributes genuinely new interactions rather than echoing the semantic entry points.

The weighting in Equation 3.4 keeps semantic relevance dominant while letting the discourse relation act as a structured tie-breaker: among neighbours of comparable similarity to the query, those connected by a stronger relation are preferred. The mechanism is intentionally simple, a single round of expansion from the entry points rather than a deep traversal, in keeping with the bounded-cost requirement of Section 3.1.

One consequence of expansion concerns retrieval coverage and must be made explicit, because it underlies the analysis in later chapters. Graph nodes are QA interactions, not dialogue turns; they carry no dialogue identifier linking them to the annotated evidence of the benchmark. They therefore contribute nothing to the dialogue-evidence recall measured in Section 3.11: only the recency turns and the semantically retrieved turns count towards that measure. Because each graph slot displaces a semantic turn that would have carried a dialogue identifier, expansion can only reduce, never increase, the dialogue evidence present in the context. This displacement is the precise mechanism by which the graph affects retrieval coverage, and it is examined directly in the parameter study of Section 3.12.

3.6 Memory Update Rule

After the generator has produced and returned an answer, the new interaction is added to the memory graph. Its text, the concatenation of the question and the answer, is embedded, and a node storing the question, the answer, and this embedding is appended to the graph. The node is then connected to existing nodes by typed edges, through a classification step that respects the bounded-cost requirement.

Rather than comparing the new interaction against every previous node, which would make the per-turn cost grow with conversation length, the system selects a fixed-size candidate set. When

the graph holds at most $K_{\text{edge}} = 8$ prior nodes, all of them are used as candidates; otherwise the candidates are the $K_{\text{edge}} = 8$ prior nodes most similar to the new interaction by embedding cosine similarity. This caps the number of relations considered at each turn regardless of how large the graph has grown, keeping the update cost bounded.

An instruction-tuned language model then classifies the relation from the new interaction to each candidate. The classifier receives the new question, the new answer, and the candidate questions, and returns a structured mapping from each candidate identifier to one of the four labels `subcase`, `supercase`, `same_level`, or `no_relation`. The model is run with a low temperature and a fixed seed for stability, and its output is parsed leniently: any candidate the model does not classify, or classifies with an unrecognised label, defaults to `no_relation`. For each candidate not labelled `no_relation`, a typed edge is added from the new node to that candidate; `no_relation` outcomes add nothing. The exact model and decoding settings are listed in Section 3.10.

This incremental update lets the graph grow without any global recomputation. Each new node attaches to a small, bounded number of relevant neighbours, so the graph remains sparse as the conversation extends, while the accumulation of these local connections produces a global structure that later retrieval steps can traverse. As noted in Section 3.3, the entire update is best-effort: should embedding, classification, or parsing fail, the error is absorbed and the node is added without edges if necessary, so that memory maintenance can never interrupt the conversation.

3.7 Research Design

The architecture is evaluated through a controlled comparison of three context-selection strategies that share everything except the way they fill the context. All three use the same conversations, the same questions, the same generation model, the same embedding model, and the same fixed-item budget of Equation 3.1. The only variable that changes between them is the selection strategy, so any difference in their results is attributable to that strategy and to nothing else.

The three strategies form a graded sequence. *Sliding Window* uses recency alone. *Semantic RAG* adds embedding-based retrieval over the history. *Graph-Augmented RAG* adds graph expansion on top of semantic retrieval, reusing the identical semantic step and differing from Semantic RAG only in that up to ten historical slots may be filled by graph-derived interactions. This grading is what gives the comparison its analytical power: the step from Sliding Window to Semantic RAG isolates the value of semantic retrieval, and the step from Semantic RAG to Graph-Augmented RAG isolates the value of the graph signal, since the graph method reduces exactly to the semantic method when the graph contributes nothing.

The evaluation is reported in the order that this design supports, and the same order is followed in the results and discussion chapters. First, the robust effect: retrieval-based methods are expected to improve substantially on the recency-only baseline, because most of the relevant evidence in a long conversation lies outside the last few turns. Second, the honest comparison of the two retrieval methods: the question of whether the graph signal improves on semantic retrieval is left open here

and answered empirically, and the answer obtained is a statistical tie rather than a win. Third, the mechanism that explains that tie, namely the displacement effect introduced in Section 3.5. Fourth, the implications for where a structurally motivated method might be evaluated more favourably in future work. The design is built to support this account rather than to demonstrate the superiority of any one method, and a clean, fair, reproducible comparison that isolates why the graph signal is neutral on this benchmark is the intended contribution.

3.8 The LoCoMo Benchmark

The methods are evaluated on LoCoMo, a benchmark designed to assess very long-term conversational memory in language-model agents [21]. LoCoMo is well matched to the problem studied here because its conversations are long enough that the full history cannot fit in a small context window, and its questions are deliberately constructed to require information distributed across the conversation rather than concentrated in the most recent turns.

The benchmark comprises ten multi-session conversations. Each conversation contains hundreds of dialogue turns distributed over several sessions, and each session carries a calendar date, which is what makes the temporal formatting of Section 3.4.3 meaningful. Accompanying each conversation is a bank of independent question–answer pairs, and each pair is annotated with the gold evidence required to answer it: the specific dialogue turns, identified by dialogue identifier, that support the correct answer. These annotations are what permit the retrieval-coverage measure of Section 3.11, since they make it possible to check whether the assembled context actually contains the turns needed to answer.

The evaluation runs every question–answer pair of all ten conversations, a total of 1,986 pairs. The pairs are distributed across five reasoning categories, summarised with their counts in Table 3.2. The categories range from single-hop questions, answerable from a single turn, through multi-hop and temporal questions that require combining or ordering information from several turns, to open-domain questions and adversarial questions designed to elicit confidently wrong answers. Reporting results per category, as the next chapter does, allows the analysis to distinguish the kinds of reasoning each memory strategy helps with.

Table 3.2: LoCoMo question categories and their counts across the ten conversations.

Category	Description	Count
1	Multi-hop	282
2	Temporal reasoning	321
3	Open-domain	96
4	Single-hop	841
5	Adversarial	446
Total		1,986

One characteristic of the benchmark is important enough to state here, because it bears directly on the interpretation of the results. The question–answer pairs are independent: each is a

self-contained probe of the conversation, and the bank as a whole does not form a natural chain of discourse in which one question specialises or generalises another. The interaction graph of the proposed method is built over exactly this bank, since each processed question–answer pair becomes a node. The typed edges therefore connect probes that were never intended to relate to one another as a conversation would. This is a limitation of construct validity, not of the implementation: the benchmark provides an honest and demanding test of retrieval, but it does not provide the kind of structured discourse that the graph is designed to exploit. The consequences of this mismatch are taken up in the parameter study and again in the discussion.

3.9 Methods Compared

The three strategies introduced in Section 3.7 are specified in full below. All three are implemented as separate context builders over a shared evaluation harness, in the modules `sliding_window.py`, `semantic_rag.py`, and `graph_augmented.py`.

Method A — Sliding Window (baseline). This method uses no retrieval, no embeddings, and no graph. The context is simply the last $T = 45$ dialogue turns, presented as a single contiguous block in chronological order. There is no separation into recency and historical layers; the window is one block. Fewer than 45 turns are used only when the conversation is shorter than the window. This is the recency-only reference against which the value of retrieval is measured.

Method B — Semantic RAG (baseline). This method uses the two-layer structure of Section 3.4.1. Layer 1 is the last five turns. Layer 2 is the forty past turns most similar to the query embedding, excluding any already in Layer 1, re-sorted into chronological order before presentation. The prompt presents the retrieved historical block first and the recent block last, under separate headings. This is a true fixed-size top-k retrieval, not a budget fill: exactly forty historical turns are selected whenever the history is large enough to provide them.

Method C — Graph-Augmented RAG (proposed). This method shares Layer 1 and the semantic retrieval of Method B exactly. It differs only in Layer 2, where up to $G = 10$ of the forty historical slots may be filled by graph-expanded interactions, as described in Sections 3.4 and 3.5, with the remaining slots filled by semantic retrieval. When the graph offers no usable neighbours, all forty slots are semantic and the method is identical to Method B. The interaction graph is built incrementally over the course of the evaluation, exactly as it would be in deployment.

Because the budget is enforced by item count rather than by token count, the methods end up using only a fraction of the nominal token budget, and they use comparable amounts of it. Table 3.3 reports the mean and maximum prompt sizes observed across the evaluation. All three methods sit well below the budget, between roughly a quarter and, at the maximum, about two-fifths of it. This confirms that the comparison is controlled by the number of items presented, not

by any difference in how much text each method happens to pack into the prompt: the fairness of the comparison rests on Equation 3.1, not on token accounting.

Table 3.3: Prompt size in tokens for each method across the full evaluation. The fixed-item budget, not token fill, is the binding constraint; the maximum observed prompt reaches about 40% of the nominal token budget.

Method	Mean prompt tokens	Max prompt tokens
Sliding Window	1,744	2,077
Semantic RAG	1,495	2,477
Graph-Augmented RAG	1,449	2,401

3.10 Models

A single small instruction-tuned language model serves every generative role in the pipeline, in keeping with the small-model requirement of Section 3.1. The same model generates the answers, classifies the discourse edges, and acts as the evaluation judge; it is run with different decoding settings appropriate to each role. A separate sentence-embedding model provides all the vector representations. Table 3.4 lists the components and their settings.

Table 3.4: Models and decoding settings used in the evaluation pipeline.

Component	Model	Settings
Answer generation	`llama-3.1-8b-instruct`	temp 0.2, max 512 tokens
Embedding	`multi-qa-mpnet-base-dot-v1`	768-dim, dot-product
Edge classifier	`llama-3.1-8b-instruct`	temp 0.1, seed 42, max 256 tokens
LLM-as-Judge	`llama-3.1-8b-instruct`	temp 0, max 200 tokens

The generation model, `meta-llama/llama-3.1-8b-instruct`, is an 8B-class open model chosen as a concrete and representative instance of the deployment target. Answer generation uses a low temperature to keep responses stable; edge classification uses a still lower temperature with a fixed seed for reproducibility; the judge uses greedy decoding so that its verdicts are deterministic.

The embedding model, `sentence-transformers/multi-qa-mpnet-base-dot-v1`, is purpose-trained for question-to-passage retrieval over a large corpus of question–answer pairs, produces 768-dimensional vectors compared by dot product, and requires no special query or passage prefixes. This choice was made after an earlier vision-language embedding model produced near-zero retrieval recall on the multi-hop and open-domain questions, where the relevant evidence shares little surface wording with the query. The purpose-trained retrieval embedder resolved that failure, which is why it is used for all similarity computations in both the semantic retrieval and the graph.

3.11 Evaluation Metrics

Answer quality and retrieval behaviour are assessed with three metrics, which serve different purposes and are not interchangeable. One is the primary metric used to select hyperparameters; one is a secondary check on answer quality; one is a diagnostic of retrieval that is explicitly excluded from selection.

Token-level F1 (primary). The primary metric is the official LoCoMo token-level F_1 between the generated answer and the reference answer, using the benchmark's own category-specific scoring code unchanged. It balances precision and recall over the answer tokens and is reported per category and as a weighted mean. Because it is native to the benchmark and directly measures answer quality, it is the metric on which the operating point is selected in Section 3.12.

LLM-as-Judge (secondary). Lexical overlap can underrate answers that are correct but phrased differently from the reference, so a complementary judgement is obtained from the same 8B model acting as a judge. The judge issues a binary correct-or-incorrect verdict given the question, the reference answer, the gold evidence turns together with one surrounding turn on each side, and the session dates. Accuracy is reported over the responses that parse; roughly one to two percent of judge outputs cannot be parsed and are excluded. The judge runs with greedy decoding so that its verdicts are deterministic.

recall@k (diagnostic). To separate retrieval failure from generation failure, the evaluation measures how much of the annotated evidence the assembled context actually contains,

$$\text{recall@}k \;=\; \frac{|\text{evidence} \cap \text{retrieved}|}{|\text{evidence}|}, \tag{3.5}$$

where "evidence" is the set of gold evidence dialogue identifiers for the question and "retrieved" is the set of dialogue identifiers present in the assembled context. As established in Section 3.5, only Layer 1 turns and semantically retrieved turns carry dialogue identifiers; graph entries are excluded from the retrieved set because they carry none. This metric is reported as a diagnostic only and is deliberately *not* used to select hyperparameters, because it increases monotonically with k and so trivially favours the largest possible context, which would defeat the purpose of selecting a budget at all.

3.12 Parameter Selection

Two parameters fix the operating point at which the system is evaluated: $K_{\text{historical}}$, the size of the historical layer, and f, the fraction of that layer that graph expansion may occupy. Both are selected before the main evaluation, following one principle: selection is made on token-level F_1, the primary metric, with judge accuracy as a secondary check, and recall@k is consulted only as a

diagnostic. recall@k is excluded from selection for the reason given in Section 3.11, namely that it rewards larger k unconditionally and so cannot distinguish a genuine improvement from a trivial one.

To keep selection honest, both sweeps are run on an independent three-conversation subset chosen *before* any sweep result was seen. The three conversations were selected a priori to span the range of dialogue length, which is the primary determinant of retrieval difficulty: the shortest conversation, the one closest to the dataset's median length, and the longest. Table 3.5 lists them. Fixing the subset this way means the chosen parameters cannot have been tuned to the conversations on which they were measured, and reserving the remaining conversations for the main evaluation keeps that evaluation independent of selection.

Table 3.5: The three-conversation subset used for parameter selection, chosen a priori to span the range of dialogue length.

Conversation	Dialogue turns	QA pairs	Role
conv-30	369	105	shortest
conv-42	629	260	median
conv-47	689	190	longest

3.12.1 Selecting the historical-layer size

The size of the historical layer was selected by sweeping $K_{\text{historical}}$ over $\{10, 20, 30, 40, 50\}$, using Semantic RAG as a representative retrieval method. Table 3.6 reports the result. Answer quality rises with k at first and then saturates: F_1 improves steadily up to $k = 40$ but is essentially flat from 40 to 50, gaining only 0.005, and judge accuracy peaks at $k = 40$ before plateauing, with the small difference at $k = 50$ falling within run-to-run noise. recall@k, by contrast, keeps rising monotonically and does not saturate, which is exactly the behaviour that disqualifies it as a selection criterion. There is no interior optimum in $[10, 50]$ for the answer-quality metrics; both simply level off by $k = 40$.

Table 3.6: k-sweep for $K_{\text{historical}}$ on the selection subset, using Semantic RAG. Answer-quality metrics saturate by $k = 40$; recall@k rises monotonically and is not used for selection.

k	Judge accuracy	F_1	recall@k
10	0.227	0.315	0.284
20	0.277	0.339	0.411
30	0.319	0.341	0.503
40	0.350	0.366	0.590
50	0.347	0.371	0.639

The per-conversation breakdown in Table 3.7 explains why a single conversation would have been insufficient to fix k. The short conversation peaks near $k = 20$ and then plateaus, whereas the median and long conversations keep improving up to $k = 40$. The optimal k therefore grows with conversation length, which is precisely why the selection subset was constructed to span that

length. Taking the subset as a whole, $k = 40$ is the smallest value at which both answer-quality metrics reach their plateau, and it is selected as $K_{\text{historical}}$.

Table 3.7: Per-conversation F_1 across the k-sweep. The optimal k grows with conversation length: the short conversation peaks near $k = 20$, the longer ones near $k = 40$.

Conversation	Turns	k=10	k=20	k=30	k=40	k=50
conv-30 (short)	369	0.295	0.419	0.404	0.400	0.423
conv-42 (median)	629	0.216	0.220	0.285	0.328	0.312
conv-47 (long)	689	0.205	0.274	0.317	0.353	0.353

3.12.2 Selecting the graph-slot fraction

With $K_{\text{historical}}$ fixed at 40, the graph-slot fraction f was selected by sweeping it over $\{0.00, 0.10, 0.25, 0.50\}$, using the Graph-Augmented method; $f = 0$ corresponds to no graph and is therefore identical to Semantic RAG. Table 3.8 reports the result, and it is the central empirical observation behind the honest positioning of this thesis: no value of f improves on $f = 0$. The graph is neutral on this benchmark. Differences in F_1 and judge accuracy among the configurations with $f > 0$ fall within run-to-run noise, and every $f > 0$ shows a small decrease in recall@k relative to $f = 0$.

Table 3.8: f-sweep for `GRAPH_SLOTS_FRACTION` on the selection subset, with $K_{\text{historical}} = 40$. No f improves on $f = 0$ (no graph); every $f > 0$ slightly lowers recall@k through displacement.

f	Graph slots	Judge accuracy	F_1	recall@k
0.00	0	0.358	0.376	0.590
0.10	4	0.331	0.374	0.564
0.25	10	0.342	0.367	0.561
0.50	20	0.353	0.363	0.564

The decrease in recall@k is exactly the displacement effect of Section 3.5 made quantitative. Each graph slot is filled by a QA-interaction node, which carries no dialogue identifier and so contributes nothing to recall, while displacing one semantic turn that would have contributed. On a benchmark whose question bank does not form natural discourse chains, the graph neighbours offer little additional signal to offset the dialogue evidence they displace, and the net effect on retrieval coverage is neutral-to-slightly-negative. The mechanism is therefore not a defect of the implementation but a predictable consequence of applying a discourse-structured method to a benchmark that lacks the structure it is designed to use, as anticipated in Section 3.8.

Because no value of f wins on the selection metric, the choice of operating point is made on a different basis and is stated explicitly as a design decision rather than an optimisation. The value $f = 0.25$ is selected, giving the graph ten of the forty historical slots. This is a substantial and clearly visible graph presence, large enough that any effect of the graph, positive or negative, would be apparent in the main evaluation, yet not so large as to dominate the historical layer. Selecting it maximises the graph's opportunity to demonstrate an effect on the full benchmark,

which is the appropriate choice when the goal is to characterise the graph signal honestly rather than to tune a metric.

3.12.3 Final operating point

The two sweeps fix the operating point at which all three methods are evaluated on the full benchmark. Table 3.9 collects the resulting parameter values. The total number of context items, $T = 45$, is identical across the three methods by construction, which is what makes the main comparison a controlled test of the selection strategy alone.

Table 3.9: Final operating point used for the main evaluation. The total item count T is identical across all three methods.

Parameter	Value	Meaning
`LAYER1_RECENT_TURNS`	5	Recency layer (Layer 1)
`K_HISTORICAL`	40	Historical slots (Layer 2)
`GRAPH_SLOTS_FRACTION`	0.25	Graph entries: up to 10 of 40 slots
`K_SEMANTIC`	5	Graph entry points
`K_EDGE_CLASSIFICATION`	8	Candidate nodes per edge classification
Total items T	45	Identical across all methods

3.13 Implementation and Reproducibility

The evaluation is implemented as a shared harness with one context builder per method, in the modules `sliding_window.py`, `semantic_rag.py`, and `graph_augmented.py`. Each builder receives a conversation and a question, assembles the context under the fixed-item budget, and returns both the prompt and the set of retrieved dialogue identifiers used to compute recall.

Several aspects of the procedure are fixed to make the results reproducible. The interaction graph is constructed incrementally during evaluation, never precomputed from the full conversation: each processed question–answer pair is added as a node before the next pair is seen, so the memory available at any point is exactly the memory a deployed agent would have accumulated by then. Decoding is controlled for stability where determinism matters: the edge classifier uses a fixed seed and a low temperature, the judge uses greedy decoding, and answer generation uses a low temperature. The embedding model is fixed, and all similarity computations, both for semantic retrieval over dialogue turns and for entry-point selection and neighbour scoring in the graph, use the same 768-dimensional embeddings under dot-product comparison.

The main evaluation runs all 1,986 question–answer pairs of all ten conversations at the single operating point of Table 3.9, for each of the three methods. Because the three methods share conversations, questions, budget, generation model, embedding model, and metric code, and differ only in the context-selection strategy, the comparison reported in the next chapter isolates the contribution of that strategy, and in particular the contribution of the graph signal, as cleanly as the benchmark allows.

3.14 Summary

This chapter has presented a graph-augmented conversational memory architecture and the procedure used to evaluate it. The architecture stores each interaction as a node in a persistent directed graph, connects interactions by typed discourse relations, and, at each turn, assembles context from three complementary signals, recency, semantic similarity, and graph expansion, under a fixed budget of forty-five items. Its design follows from two concrete constraints: bounded per-turn cost, satisfied by deferring memory maintenance and capping the edge-classification candidate set, and compatibility with small open-source models, instantiated by the 8B model used throughout.

The evaluation compares this method against a recency-only baseline and a semantic-retrieval baseline on the LoCoMo benchmark, holding everything constant except the selection strategy so that the graph signal is isolated as the single variable under study. The operating point was selected on token-level F_1 over an independent, length-spanning subset, and the parameter study already reveals the central finding that the next chapters develop: the graph is neutral on this benchmark, with its graph slots displacing dialogue evidence that the benchmark's independent question bank gives the graph little structure to replace. The contribution is thus a fair, reproducible, controlled account of where and why a structurally motivated memory method is neutral, which the following chapter reports in full.

Chapter 4

Results and Discussion

This chapter reports and interprets the empirical evaluation of the three context-selection strategies introduced in Chapter 3. All results come from a full run of the three methods at the operating point fixed in Section 3.12, namely $K_{\text{historical}} = 40$ and a graph-slot fraction of 0.25, evaluated on the complete LoCoMo benchmark: every question–answer pair of all ten conversations, 1,986 pairs in total. The generation, edge-classification, and judging roles are all served by the same `llama-3.1-8b-instruct` model, and all similarity computations use the `multi-qa-mpnet-base-dot-v1` embedder, exactly as specified in Section 3.10. Because the three methods share the conversations, the questions, the generation and embedding models, the fixed item budget, and the metric code, every difference reported below is attributable to the context-selection strategy alone.

The chapter follows the order anticipated in the research design of Section 3.7 and proceeds in four parts. First, the robust effect: both retrieval-based methods improve on the recency-only baseline by a large margin, on every metric and in nearly every category, and this is the decisive finding of the thesis. Second, the comparison of the two retrieval methods: Semantic RAG and Graph-Augmented RAG perform at statistically equivalent levels on the aggregate, trade category results symmetrically, and agree on the large majority of questions. Third, the category-level breakdown, which shows where the aggregate comes from, including the one category in which the sliding window appears to win and why that appearance is an artefact. Fourth, the mechanism behind this equivalence: a measurable displacement of dialogue evidence by graph entries, set against a construct-validity limitation of the benchmark that explains why the graph signal, though active in nine of every ten questions, carries little answer-relevant information here. The chapter closes with the broader implications of these findings and a summary.

4.1 Overall Results

Table 4.1 reports the aggregate results across all 1,986 question–answer pairs. It is the reference point for the entire chapter, and the three parts of the argument each return to it from a different angle. Three quantities describe each method: the mean token-level F_1 that is the primary answer-quality metric, the LLM-as-Judge accuracy that serves as a secondary check on answer

quality, and the recall@40 diagnostic that measures how much of the annotated gold evidence the assembled context actually contained. The number of answers judged correct, the number of judge responses that parsed, and the mean prompt size in tokens are included to make the comparison fully transparent.

Table 4.1: Overall results across all 1,986 question–answer pairs at the operating point $K_{\text{historical}} = 40$, $f = 0.25$. Judge accuracy is computed over parsed responses; the unparsed rate (about 1.3%) is uniform across methods. recall@40 is averaged over the 1,982 pairs that carry at least one gold-evidence identifier.

Metric	Sliding Window	Semantic RAG	Graph-Augmented
Mean F_1	0.2627	**0.3544**	0.3490
Judge accuracy	0.1413	**0.3172**	0.3126
Correct answers	277	**622**	613
Judge parsed	1,960	1,961	1,961
Judge failed (unparsed)	26	25	25
recall@40	0.0745	**0.5720**	0.5564
Mean prompt tokens	1,744	1,495	1,449

Two features of the table frame the analysis that follows. The gap between the sliding window and the two retrieval methods is large on every row: F_1 rises by roughly nine points, judge accuracy roughly doubles, and recall@40 increases almost eightfold. The gap between the two retrieval methods, by contrast, is small on every row: their F_1 scores differ by 0.0054, judge accuracy by 0.0046, and correct answers by nine out of nearly two thousand. The first gap is the headline result; the second is statistical equivalence. The mean prompt sizes confirm once more that the comparison is controlled by item count and not by token fill: all three methods use between roughly 1,450 and 1,750 tokens on average, a small fraction of the nominal budget, and the two retrieval methods in fact use *fewer* tokens than the sliding window while answering far better. The advantage is therefore in the selection of items, not in their volume.

4.2 Retrieval versus Recency: The Decisive Effect

The clearest result of the evaluation is that retrieval is decisive and the recency-only baseline is not competitive. On answer quality, mean F_1 rises from 0.2627 for the sliding window to 0.3544 for Semantic RAG and 0.3490 for Graph-Augmented RAG, an improvement of roughly nine F_1 points. The judge accuracy shows the same pattern even more clearly: the proportion of answers judged correct rises from 0.1413 to 0.3172 and 0.3126 respectively, so retrieval roughly *doubles* the share of correct answers. Both answer-quality metrics, computed independently and measuring different aspects of the output, agree, and by a wide margin.

The retrieval diagnostic explains why, and the causal chain is direct. recall@40 measures the fraction of gold-evidence dialogue turns that the assembled context actually contained. For the sliding window it is 0.0745: the context held, on average, well under a tenth of the evidence needed to answer. For the two retrieval methods it is 0.5720 and 0.5564, an almost eightfold

increase. The reason is structural and was anticipated in Section 3.8. The sliding window sees only the last 45 dialogue turns, but LoCoMo's conversations span hundreds of turns across several dated sessions, and its questions are deliberately constructed so that the supporting evidence lies distributed through the conversation rather than concentrated at its end. In such conversations the evidence almost always falls outside the recency window, and a method cannot answer from what it never sees. The near-zero recall of the sliding window is not a shortcoming of the generator; it is a direct consequence of presenting the model with the wrong 45 turns.

This result carries the main practical message of the evaluation, and that message is worth stating directly. All three methods present the generator with the same number of items, 45, and even the same approximate token volume; they differ only in *which* items those are. The eightfold swing in evidence coverage, and the doubling of answer accuracy that follows it, are therefore produced entirely by the selection strategy. For the deployment regime that motivates this thesis, namely small, open, locally hosted models operating under a fixed and modest context budget, as set out in Section 3.1, this is the central practical payoff. An 8B-class model benefits not from a larger context but from a better-chosen one, and choosing well is precisely the function of a retrieval layer.

The finding also confirms, on this benchmark, the general observation that a small model is poorly served by a long and undirected context and well served by a short and relevant one. Surrounding the evidence with recent-but-irrelevant turns, as the sliding window does, leaves the answer unsupported [19].

It is against this robust effect that the second, much finer comparison must be read. Retrieval clearly matters; the open question, left deliberately unanswered by the design and answered empirically here, is whether adding a graph signal on top of semantic retrieval matters as well.

4.3 Semantic versus Graph-Augmented Retrieval

The comparison between the two retrieval methods gives a clear result: the two perform at statistically equivalent levels. On the aggregate their scores differ only marginally: F_1 of 0.3544 and 0.3490, judge accuracy of 0.3172 and 0.3126, and 622 versus 613 correct answers, a difference of nine answers out of 1,961 that parsed. These gaps are far smaller than the variation expected between runs of a stochastic generator, and, as the category breakdown in Section 4.4 shows, the two methods do not even agree on which of them is better from one category to the next. The difference has no consistent direction, and its magnitude is small.

The agreement analysis in Table 4.2 makes this equivalence concrete. Of the 1,943 questions on which both methods produced a parsable judge verdict, the two agree on 1,719, or 88.5%: they are jointly correct on 500 and jointly incorrect on 1,219. They disagree on only 11.5% of questions, and on that remainder the split is nearly even: 116 questions answered correctly by Semantic RAG alone against 108 answered correctly by Graph-Augmented RAG alone, a net advantage of eight questions to the semantic baseline. A method that genuinely improved on

another would not divide the disagreements so evenly; this even split is the signature of statistical equivalence at the level of individual questions.

Table 4.2: Agreement between Semantic RAG and Graph-Augmented RAG over the 1,943 questions on which both produced a parsed judge verdict. The two methods agree on 88.5% of questions and split the remainder almost evenly.

Outcome	Count
Both correct	500
Both incorrect	1,219
Agree (either way)	1,719 (88.5%)
Semantic correct, Graph incorrect	116
Graph correct, Semantic incorrect	108

The conclusion is therefore as follows. At this operating point, adding the graph layer neither improves nor degrades answer quality on LoCoMo to any degree that the data can distinguish from noise. This is reported as statistical equivalence, neither presented as a graph improvement nor dismissed as a graph failure. The remainder of the chapter is devoted to understanding it: first by decomposing the aggregate into its categories, then by identifying the mechanism that produces this equivalence.

4.4 Category-Level Analysis

The aggregate numbers are weighted means over five reasoning categories of very different sizes and difficulties, and the breakdown by category both explains where the aggregate comes from and exposes one result that the aggregate alone would misrepresent. Table 4.3 reports mean F_1 per category and Table 4.4 reports judge accuracy per category, for all three methods. The category labels follow the verified LoCoMo mapping established in Section 3.8: category 1 is multi-hop, category 2 temporal, category 3 open-domain, category 4 single-hop, and category 5 adversarial.

Table 4.3: Mean F_1 by category. Semantic RAG and Graph-Augmented RAG are indistinguishable on the four genuine-retrieval categories; the sliding window's high score on adversarial is examined in Section 4.4.1.

Category	n	Sliding	Semantic	Graph
Single-hop	841	0.1061	**0.4373**	0.4346
Multi-hop	282	0.0742	**0.2669**	0.2412
Temporal	321	0.0444	0.1764	**0.1820**
Open-domain	96	**0.1084**	0.1054	0.1051
Adversarial	446	**0.8677**	0.4350	0.4283

The main point of the two tables, for the retrieval methods, is that neither is consistently higher than the other. Under F_1, Semantic RAG scores higher on single-hop, multi-hop, and adversarial, and Graph-Augmented RAG on temporal; under the judge, Graph-Augmented RAG scores higher on single-hop and multi-hop, and Semantic RAG on temporal, open-domain, and adversarial.

Table 4.4: Judge accuracy by category. Which of the two retrieval methods scores higher in a category is not the same under F_1 and under the judge, which is itself a symptom of noise-level differences.

Category	Sliding	Semantic	Graph
Single-hop	0.125	0.538	**0.543**
Multi-hop	0.032	0.114	**0.121**
Temporal	0.051	**0.179**	0.152
Open-domain	0.095	**0.073**	0.064
Adversarial	**0.318**	0.183	0.167

The higher-scoring method changes from one category to the next, and even between the two metrics within a single category, with no consistent pattern. Moreover, every gap between the two retrieval methods is small relative to the per-category standard deviation of F_1, which ranges from about 0.19 on the lowest-variance category to about 0.50 on adversarial. A difference of two or three thousandths of an F_1 point, against a standard deviation two orders of magnitude larger, is exactly the pattern expected from run-to-run noise. The category view, in other words, confirms the equivalence of Section 4.3 from a second angle.

Single-hop: the strongest category for retrieval. Single-hop questions are the largest category at 841 of the 1,986 pairs, and they are where retrieval performs best: F_1 of about 0.44 and judge accuracy of about 0.54 for both retrieval methods, against 0.1061 and 0.125 for the sliding window. These questions are answerable from a single supporting turn, which is precisely the case that semantic retrieval handles well, and because the category is the largest it dominates the aggregate. Semantic and Graph-Augmented RAG are statistically identical here (0.4373 versus 0.4346 on F_1; 0.538 versus 0.543 by the judge), and given the category's size this near-identity is the single largest reason the aggregates are so close.

Multi-hop and temporal: the hard categories. Multi-hop ($n = 282$) and temporal ($n = 321$) are the difficult categories for every method. Multi-hop questions require combining evidence from several turns; retrieval recovers some of that evidence but rarely all of it, and the 8B model struggles to compose what it does recover. The retrieval methods reach only about 0.25 in F_1 and about 0.12 by the judge, far below their single-hop performance, though still several times the sliding window's 0.0742 and 0.032. Temporal questions require ordering events and performing date arithmetic, which the small model does poorly even when the relevant turns are present: retrieval coverage on this category is reasonable (recall@40 above 0.51, as Section 4.5 shows), yet F_1 stays near 0.18 and judge accuracy near 0.15. These two categories are also where the retrieval methods trade small differences: Semantic RAG is higher on multi-hop F_1 while Graph-Augmented RAG is higher by the judge, and temporal is the one category where Graph-Augmented RAG is higher on F_1 (0.1820 versus 0.1764) while lower by the judge. The net effect on both categories is inconclusive, consistent with the overall equivalence.

Open-domain: small and low-signal. Open-domain is the smallest category at 96 questions, and it carries little signal. All three methods sit near 0.105 in F_1, and it is the only genuine-retrieval category in which the sliding window's F_1 (0.1084) marginally exceeds the retrieval methods' (about 0.105). On a category this small the difference amounts to a handful of questions and lies well within noise; it should not be over-read as a sliding-window strength. The category is also where retrieval coverage is weakest (these questions draw on commonsense and world knowledge that is only loosely anchored to specific dialogue turns), and four of the 96 pairs carry no gold evidence at all and are excluded from the recall average, which is therefore computed over 92.

4.4.1 The Adversarial Anomaly: An Abstention Artefact

The adversarial category is the one place where the sliding window scores higher than the retrieval methods, and by a wide margin: F_1 of 0.8677 against about 0.43 for both retrieval methods, and judge accuracy of 0.318 against about 0.18. This is the only category in which the sliding window scores highest, and it is the reason its aggregate F_1 is not even lower than it is. It must be read carefully, because it is an artefact of how the category is scored and not a genuine capability of the recency baseline.

The key fact is that all 446 adversarial questions share the same gold answer: the exact string "Not mentioned in the conversation". These are questions deliberately constructed to be unanswerable from the conversation, and the correct response is to abstain. The sliding window, which retrieves almost no relevant context on these questions (its recall@40 on adversarial is 0.0919, no higher than elsewhere), is frequently left without usable context and defaults to the abstention response, which is an exact lexical match to the gold answer and therefore scores $F_1 = 1.0$. The retrieval methods, by contrast, succeed in surfacing topically-related material, and that material is exactly what misleads them: prompted with plausible but non-supporting context, the generator produces a specific, confident, and wrong claim, scoring $F_1 = 0$. The two outcomes are visible in matched examples. Asked "Why did Melanie choose the adoption agency?", whose gold answer is the abstention string, the sliding window answers "Not mentioned in the conversation" and scores 1.0; asked "What is Melanie excited about in her adoption process?", also unanswerable, a retrieval method answers "creating a family for kids who need one" and scores 0.

The correct interpretation is that on adversarial questions retrieval is penalised precisely for working. By bringing back related context, the retrieval methods induce the model to answer where abstention was the right move. This is a known failure mode of retrieval-augmented generation on unanswerable questions, namely over-answering in the presence of plausible-but-irrelevant evidence, rather than any sign that the sliding window understands adversarial cases. The category therefore should not be read as a recency strength, and the aggregate of Table 4.1 is, if anything, slightly generous to the sliding window, since this one artefactual category inflates its score. The finding is also a genuine and practical pointer for deployment, taken up again in Section 4.7: a retrieval-augmented assistant needs an explicit means of declining to answer when the retrieved context does not actually support a response.

4.5 The Displacement Mechanism

Having established that the two retrieval methods are equivalent on answer quality, the analysis turns to the mechanism. The most direct evidence of what the graph does to the context is in the retrieval diagnostic, broken down by category in Table 4.5. The pattern is regular: Graph-Augmented RAG has lower recall@40 than Semantic RAG in every single category, by a small and consistent margin, and never higher in any.

Table 4.5: recall@40 by category. Graph recall is *uniformly* below semantic recall in every category, by a small, consistent margin, the signature of the displacement effect. (Open-domain recall uses $n = 92$; the other categories use the full counts.)

Category	Sliding	Semantic	Graph
Single-hop	0.0809	**0.6482**	0.6292
Multi-hop	0.0402	**0.3379**	0.3234
Temporal	0.0654	**0.5210**	0.5179
Open-domain	0.0688	**0.2890**	0.2745
Adversarial	0.0919	**0.6715**	0.6525

This uniform dip is the displacement effect of Section 3.5 expressed quantitatively, and it is a controlled demonstration rather than a stochastic coincidence. Graph nodes are QA interactions, not dialogue turns, and they carry no dialogue identifier; by construction they contribute nothing to recall@40, which counts only the gold-evidence dialogue identifiers present in the context. At the same time, every slot given to a graph entry is one slot taken from semantic retrieval, so each injected graph node displaces exactly one dialogue turn that might have carried gold evidence. The graph can therefore only reduce, never increase, the dialogue evidence in the context. The aggregate recall falls from 0.5720 for Semantic RAG to 0.5564 for Graph-Augmented RAG, a drop of about 0.016, and the per-category table shows that this drop is distributed evenly across all five categories, exactly as a uniform displacement of a few evidence-bearing turns per question would predict. The recall cost of the graph is thus real, measurable, and precisely the size the design anticipated in the parameter study of Section 3.12.2.

The key observation is that this measurable loss of evidence coverage does *not* translate into a corresponding loss of answer quality. The recall gap of 0.016 between the methods coincides with an F_1 gap of 0.0054 and a judge gap of 0.0046, both within noise.

Two facts reconcile the real recall cost with the neutral answer-quality outcome. First, the displaced turns are the marginal ones: semantic retrieval fills the historical layer in rank order, so the turns a graph node pushes out are the lowest-ranked, least-similar semantic hits, which were the least likely to carry decisive evidence in the first place. Second, the graph nodes occasionally contribute a relevant earlier interaction that semantic similarity alone would have missed, compensating for some of what they displace. The net of a small cost in marginal evidence and an occasional structural gain is the neutral result already seen on aggregate. The displacement effect, then, explains the direction of the small recall gap completely while leaving the answer-quality equivalence intact.

4.6 Graph Activity and the Construct-Validity Limitation

A neutral result for the graph would be uninformative, and easy to dismiss, if it arose simply because the graph did nothing. That is not the case here, and establishing it is necessary before the neutrality can be attributed correctly. Table 4.6 reports how often and how heavily the graph layer was used across the evaluation.

Table 4.6: Distribution of the number of graph-derived nodes injected per question (out of a maximum of ten slots) in the Graph-Augmented run. The graph fired in 1,791 of 1,986 questions (90.2%), injecting a mean of 3.02 nodes overall and 3.35 when it fired.

Graph nodes	0	1	2	3	4	5	6	7	8	9	10
Questions	195	328	388	384	263	167	107	74	43	21	16

The graph was active and substantially used. It injected at least one graph node in 1,791 of the 1,986 questions, or 90.2%, and on average displaced about three of the forty historical slots, with a mean of 3.02 nodes per question overall and 3.35 on the questions where it fired. Only 195 questions received no graph contribution: those early in each conversation, before any edges had formed, or those for which the entry points had no usable neighbours. On exactly those questions the Graph-Augmented method reduces identically to Semantic RAG, as the design requires.

The neutral aggregate is therefore not the product of an idle component. The graph fired in nine of every ten questions and changed the contents of the context substantially, and still it neither helped nor hurt. This is what makes the neutrality informative rather than vacuous: the structure is built, queried, and injected, yet carries no net answer-relevant signal on this benchmark.

The structure of the graph itself, reported in Table 4.7, points to why. Built incrementally over the QA bank, the ten per-conversation graphs together hold 1,986 nodes (one per interaction) and 3,954 typed edges, about two edges per node, so the graph is genuinely connected rather than sparse to the point of inertness. The distribution of edge types, however, is skewed. The classifier labels 60.3% of edges `subcase` and 39.6% `same_level`, and almost never emits `supercase`: just four `supercase` edges out of 3,954. The directional generalisation relation that the discourse model provides is, in practice, essentially absent.

Table 4.7: Structure of the interaction graph aggregated over the ten conversations. The classifier almost never emits `supercase`: directional generalisation barely forms, consistent with a bank of independent questions rather than a hierarchical line of inquiry.

Edge type	Count	Share
`subcase` (strength 1.0)	2,384	60.3%
`same_level` (strength 0.6)	1,566	39.6%
`supercase` (strength 0.8)	4	0.1%
Total edges	3,954	—
Total nodes	1,986	—

The explanation for both the near-absence of `supercase` and the neutral answer-quality effect is the same, and it is a limitation of the benchmark rather than of the method, as anticipated in

Section 3.8. The typed discourse edges were designed to capture how a line of questioning develops: how one interaction specialises, generalises, or runs parallel to another within an unfolding conversation. LoCoMo's question bank, however, is not an unfolding conversation. It is a set of *independent* evaluation probes, authored to interrogate a fixed conversation from many angles, with no intended relation of one question to the next.

Building the interaction graph over this bank connects probes that were never meant to relate as discourse, and over independent probes there is little developmental structure for the edges to encode. The graph forms `subcase` and `same_level` links by surface specificity, but a generalisation relation between two interactions presupposes that one was actually raised as a broadening of the other, which independent probes are not; hence `supercase` almost never applies. The edges, though plentiful, carry limited answer-relevant signal, and the expansion they drive surfaces interactions that are structurally adjacent but not decisively more useful than the marginal semantic turns they displace. The neutral result is the direct consequence.

This is the core of the thesis's negative result. The experiment does not show that discourse structure is useless for conversational memory; it shows, under a clean and controlled comparison, precisely *why* a structurally-motivated method is neutral on this particular testbed, namely because the testbed lacks the developing discourse the method is built to exploit, while the method itself is active, well-formed, and faithfully implemented. Isolating that reason is a contribution, and a more useful outcome than an unexplained win or an unexplained loss would have been. It also identifies, with some precision, the kind of evaluation on which the method's premise could actually be tested: settings in which a sequence of questions forms a real line of inquiry, such as agentic task decomposition, tutoring dialogues, or multi-turn technical support, where one interaction genuinely specialises or generalises another.

4.7 Discussion and Implications

Three implications follow from the results as a whole, and they are drawn together here before the chapter closes.

The first concerns the deployment regime that motivated the architecture. The decisive finding of Section 4.2 is that, for a small open model under a fixed and modest context budget, the quality of the answer is governed by which turns the model is shown, not by how many. The retrieval methods present the same number of items as the sliding window, and slightly fewer tokens, yet roughly double the share of correct answers. For resource-constrained, locally deployable settings, where the appeal of an 8B-class model is its low cost, low latency, and suitability for private on-device use, but where its short context window and limited reasoning leave it least able to recover from a poorly chosen prompt, this is the practical message. Investing in selection is the most effective way to make such a model useful on long conversational histories.

The second concerns the graph signal specifically. At this operating point the graph neither helped nor hurt, and Section 4.6 located the reason in the benchmark rather than the implementation. The proper conclusion is not that discourse-structured memory is a failed direction, but

that LoCoMo cannot adjudicate it: its independent question bank does not contain the developing discourse the graph is built to exploit, as the near-total absence of generalisation edges directly shows. The result therefore reframes the contribution exactly as Chapter 3 proposed, as a fair, reproducible, controlled account of where and why the graph signal is neutral, and it motivates, rather than forecloses, evaluation on testbeds whose questions form genuine lines of inquiry. Detailed proposals along these lines are deferred to the conclusions.

The third is a concrete deployment caution drawn from the adversarial category of Section 4.4.1. There, retrieval was penalised for working: by surfacing plausible-but-non-supporting context, it induced the model to answer confidently where abstention was correct, while the sliding window's lack of retrieved context was scored as correct. The lesson is not that recency is better but that a retrieval-augmented system needs an explicit mechanism for declining to answer when the retrieved evidence does not in fact support a response. Over-answering on unanswerable questions is a real risk for any deployed retrieval-augmented assistant, and the adversarial results quantify its cost on this benchmark.

Finally, the limits of these conclusions should be stated plainly. The evaluation uses a single 8B-class generator, a single embedding model, and a single benchmark, and the answer-quality differences between the two retrieval methods are small enough that a different generator or judge could move them slightly without changing the qualitative picture. The headline retrieval-versus-recency effect is large enough to be robust to such variation; the graph-versus-semantic equivalence is, by its nature, a statement that no effect was detectable here, which is weaker than a demonstrated absence of any effect anywhere. Both claims are made at the strength the evidence supports and no further.

4.8 Summary

This chapter has reported and interpreted the full-benchmark evaluation of the three context-selection strategies at the selected operating point, over all 1,986 LoCoMo question–answer pairs. Three findings structure the result. First, retrieval is decisive: both retrieval methods improve on the recency-only sliding window by roughly nine F_1 points, double the share of answers judged correct, and lift evidence coverage almost eightfold, an effect produced entirely by the selection of items under an identical budget and the central practical payoff for small, locally deployed models.

Second, the two retrieval methods are statistically equivalent: their scores differ by margins indistinguishable from noise, the two agree on 88.5% of questions and split the rest almost evenly, and many of those splits are equivalent answers separated only by judge stochasticity. Third, this equivalence has a clear and dual explanation: a measurable displacement effect, by which graph entries uniformly lower evidence coverage by a small margin without a matching loss in answer quality, set against a construct-validity limitation, by which LoCoMo's bank of independent questions offers little of the developing discourse the graph is designed to exploit, a limitation made visible by the near-total absence of generalisation edges, and not attributable to an idle graph, since the graph fired in over 90% of questions. The contribution of the evaluation is thus a clean, honest,

and controlled characterisation of where a discourse-structured memory method helps, where it is neutral, and why, which the concluding chapter takes up to draw the thesis to a close.

Chapter 5

Conclusions

This dissertation addressed a focused problem: how a conversational agent should choose, within a fixed token budget, which parts of a long interaction history to place before the model. Small, open, locally deployable language models are attractive to organisations that must keep sensitive information and operational knowledge on their own infrastructure. These are also the models whose short context windows and limited reasoning capacity make the selection of context most important.

The question was therefore not how to pass *more* context to such a model, but how to select *more useful* context within the same budget. A second question followed from it: whether the structure of a conversation, beyond recency and semantic similarity, could be turned into a usable selection signal.

To answer these questions, a graph-augmented conversational memory architecture was designed, implemented, and evaluated. Each human-model interaction is a node in a persistent graph, connected to earlier interactions through typed discourse relations, and context is assembled at each turn from three sources: recent turns, semantic retrieval, and a graph-based expansion from the semantically retrieved entry points. The architecture was compared against a sliding-window baseline and a semantic-retrieval baseline on the LoCoMo benchmark for long-term conversational memory. The three methods shared the same conversations, questions, models, and budget, so that any difference reflects the context-selection strategy alone.

This chapter draws the work together. It revisits the research objectives against the evidence obtained, presents a prototype built around the memory layer, states the limitations that bound the conclusions, sets out the directions the work opens, and closes with the broader context that motivated it.

5.1 Revisiting the Research Objectives

The principal objective, stated in Section 1.5, was to design, implement, and empirically evaluate a graph-augmented conversational memory architecture for SLMs operating under fixed context

budgets, framed as a first and controlled investigation rather than as a finished system. The architecture was built and run end to end under the same sequential conditions as a deployed agent, the graph was constructed incrementally as each conversation unfolded, and the evaluation isolated the contribution of the graph from those of recency and semantic retrieval.

The principal research question asked whether conversational discourse structure, represented as a graph of typed relations between past interactions, can be turned into a usable signal for selecting context, and under what conditions it contributes to the quality of an agent's responses. The evidence gathered in Chapter 4 supports a two-part answer.

The first part is clear, and is the main practical finding of the thesis. Under a fixed and modest budget, the quality of a small model's answers depends on *which* past turns it is shown, not on how many. Both retrieval-based methods improved on the recency-only baseline by a wide margin on every answer-quality metric, while presenting the model with the same number of items and slightly fewer tokens (Section 4.2, Table 4.1). The improvement is therefore attributable to selection alone.

For the deployment regime that motivated the work, small open models running on local infrastructure, this is the practical conclusion: such a model benefits more from a better-chosen context than from a longer one, and a retrieval layer is the most effective way to provide it. The result is consistent with earlier evidence that models underuse information placed in long prompts [19].

The second part concerns the graph signal. On LoCoMo, adding the graph layer on top of semantic retrieval neither improved nor degraded answer quality by a margin the data could separate from noise: the two retrieval methods were statistically equivalent (Section 4.3). This is neither a success nor a failure of the underlying idea, and the reason for the equivalence can be identified. Two factors account for it.

The first is a small, measurable displacement cost. Each graph-derived entry occupies a slot that would otherwise hold a retrieved dialogue turn, so the graph lowers the amount of gold evidence in the context, though without a matching loss in answer quality (Section 4.5, Table 4.5).

The second factor is more fundamental: a construct-validity limitation of the benchmark. The typed discourse relations were designed to capture how a line of questioning develops over a conversation, but LoCoMo's questions form an independent evaluation bank rather than an unfolding line of inquiry. Over independent probes there is little developmental structure for the edges to encode, as the near-total absence of generalisation edges confirms (Section 4.6, Table 4.7). The graph was not inert, however: it was active in nine of every ten questions and changed the contents of the context substantially, yet it carried no net answer-relevant signal on this testbed (Table 4.6).

The correct interpretation is therefore that LoCoMo cannot exercise discourse-structured memory, rather than that the direction has failed. The contribution, as Chapter 3 anticipated, is a controlled and reproducible account of where a discourse-structured memory layer helps, where it is neutral, and why. The same analysis identifies the conditions under which the method's premise could be tested directly: settings in which a sequence of questions forms a genuine line of inquiry. This turns a neutral empirical result into a concrete research direction, taken up in Section 5.4.

The evaluation conducted here addresses a specific question: whether discourse structure provides an additional retrieval signal beyond semantic similarity. The graph, however, serves purposes beyond retrieval performance alone. As discussed in Section 5.2, making the structure of a conversation explicit creates opportunities for navigation, inspection, and resumption that are not available in a linear transcript. The graph is also a persistent representation of the conversation itself. Rather than existing only as an intermediate retrieval structure, it records how topics, questions, and their relations develop over time, making it potentially useful as a foundation for longer-term memory and for retrieval across conversations as well as within them, directions outlined in Section 5.4.3. These possibilities were not evaluated in the present work and no claim is made about their effectiveness. They are noted here only to distinguish the absence of a measurable answer-quality benefit on LoCoMo from a broader judgement about the usefulness of the underlying structure.

A further lesson concerns unanswerable questions. When the conversation does not contain an answer, retrieval can still surface plausible but non-supporting context, which induces the model to respond where abstention was the correct behaviour (Section 4.4.1). Any deployed retrieval-augmented assistant therefore needs an explicit means of declining to answer when the retrieved evidence does not support a response.

5.2 A Prototype Conversational-Memory Interface

The empirical study evaluated the memory architecture as a context-selection mechanism, in isolation and under a deliberately strict protocol. The conversational graph is more than an internal retrieval index, however. Because it makes the structure of a dialogue explicit, it can also be shown directly to a user. To explore this, a full-stack prototype application was built around the memory layer. It is described here as evidence that the architecture can be embodied in a usable system, and as motivation for the directions that follow. The prototype was not subjected to any empirical or user evaluation, and no claim of effectiveness is made for it; the discussion is confined to what was built and why it is of interest.

The prototype targets the scenario that motivated the work in Chapter 1: an organisation whose members interact, through a conversational agent, with large and densely cross-referenced technical documents, and whose conversations consequently grow long, non-linear, and difficult to revisit.

It exposes the same conversation through two complementary views. The first is a conventional chat interface (Figure 5.1), through which the user converses with the agent in the familiar linear form. The second is a graph interface (Figure 5.2), which renders the conversational memory as an explicit graph of interactions and their typed relations, so that the thematic and semantic structure of a long dialogue becomes visible rather than buried in a transcript.

The second view addresses a dimension of long conversations that retrieval performance does not capture. A linear transcript forces a user returning to a long conversation to re-read it in order to recover where a topic was raised, refined, or decided. An explicit graph of topics and their

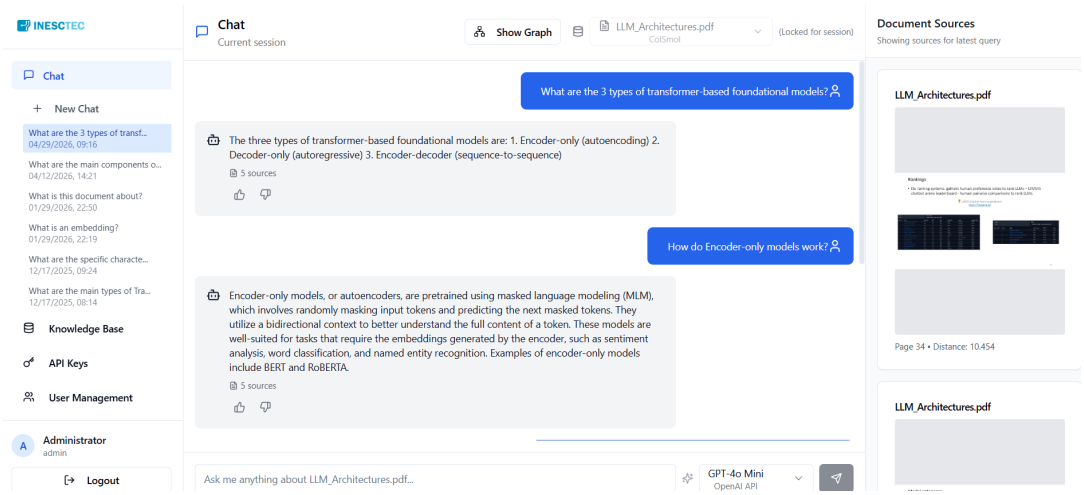


Figure 5.1: Chat view of the prototype application. The user converses with the agent in the usual linear form, while the underlying memory layer records each interaction as a node in the conversational graph.

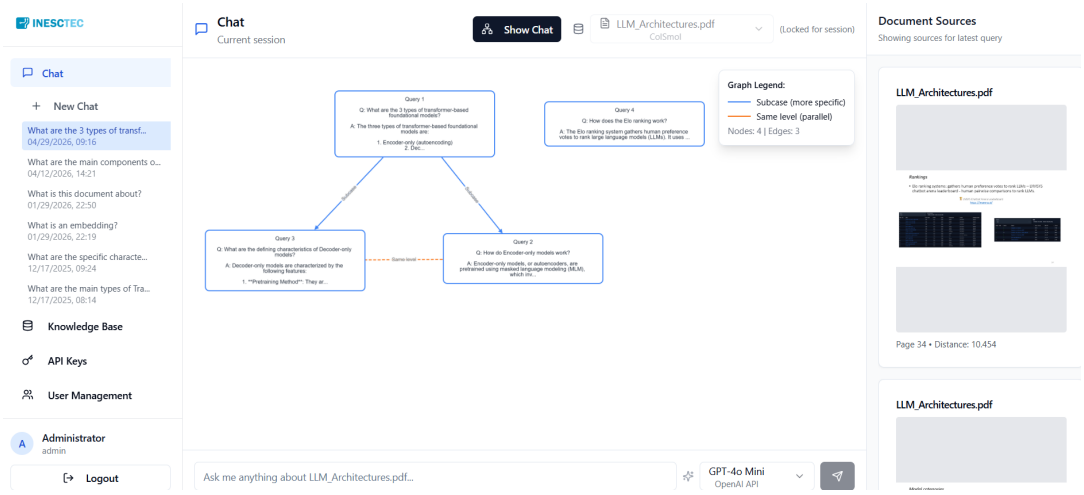


Figure 5.2: Graph view of the same conversation. Interactions are shown as nodes connected by typed discourse relations, exposing the thematic and semantic structure of a dialogue and allowing the user to navigate it and resume it at the relevant point rather than re-reading a linear transcript.

relations instead lets the user see how the conversation evolved, locate the relevant region, and re-enter it directly.

The structure that the agent traverses internally to assemble context can thus also serve the user: as a means of navigating and resuming a conversation, and of seeing what the agent has retained. These benefits are qualitative and remain to be tested. They are noted here as a feasibility result and as a bridge to the research programme set out below.

5.3 Limitations

The conclusions above hold within several bounds, stated here so that the strength of each claim is clear.

The evaluation rests on a single configuration of components: one 8B-class generation model, one embedding model, and one judge, all evaluated on a single benchmark. The headline retrieval-versus-recency effect is large enough to be robust to reasonable variation in these choices, but the fine comparison between the two retrieval methods is, by its nature, a statement that no effect was detectable under these conditions, which is weaker than a demonstration that no effect exists. A different generator, embedder, or judge could shift the small margins without altering the qualitative picture, but this has not been established.

The most consequential limitation is the one already identified as the explanation for the neutral graph result. LoCoMo is a sound benchmark for long-term conversational memory, but its question bank is a set of independent probes rather than a developing line of inquiry, and so it does not contain the discourse structure the graph is built to exploit. The graph result is therefore conditioned on a testbed that cannot, even in principle, exercise the mechanism's premise. This is a limitation of the evaluation, not of the method, but it bounds what the evaluation can conclude about the method.

Several limitations are internal to the architecture as implemented. The graph-expansion and ranking strategy is deliberately simple, and the quality of the graph depends on a relation classifier that, on this data, almost never produced the directional generalisation relation the discourse model provides. The computational cost of building and maintaining the graph incrementally, while bounded by design, was not analysed systematically, and would matter in a latency-sensitive deployment. Finally, as noted in Section 5.2, the interaction and visualization capabilities of the prototype were not evaluated at all; the navigation, visualization, and organisation benefits they suggest are described, not demonstrated.

5.4 Future Work

Each of the limitations above points to a distinct line of further work. Three are set out here, ordered from the most immediate to the most ambitious.

5.4.1 Evaluation on Genuine Lines of Inquiry

The clearest implication of the results is methodological. Because the neutral graph result was traced to the absence, in LoCoMo, of the developing discourse the method requires, the natural next step is to evaluate the architecture on settings whose questions genuinely form a line of inquiry, in which one interaction specialises, generalises, or runs parallel to another. Agentic task decomposition, tutoring dialogues, and multi-turn technical support are concrete examples, and the last is especially aligned with the technical-document scenario that motivated this work in Chapter 1, where a user moves repeatedly between sections, asks follow-up questions at varying levels of granularity, and returns to earlier points after exploring others.

On such material the typed edges would encode real developmental relations, and the central question, whether discourse structure adds a usable retrieval signal beyond semantic similarity, could be answered on a testbed capable of exercising it. Constructing or adapting a benchmark with these properties is itself a useful contribution, and a prerequisite for that test.

5.4.2 Richer Graph Construction and Retrieval

A second line concerns the mechanism itself. The expansion strategy used here was kept deliberately simple to isolate the contribution of the graph; with that contribution now characterised, more capable variants can be explored.

Promising directions include ranking expanded nodes by a combination of semantic similarity, graph distance, relation type, and node centrality, rather than by proximity alone; improving the relation classifier so that directional relations such as generalisation are reliably recovered where they exist; and adding an explicit abstention mechanism that allows the agent to decline to answer when the assembled context, retrieved or expanded, does not in fact support a response. The last of these follows directly from the adversarial findings of Section 4.4.1 and would benefit any retrieval-augmented assistant, not only the graph-based one.

5.4.3 Toward a Knowledge-Graph Conversational-Memory Layer

The architecture studied here is best understood as an initial baseline for a graph-based conversational-memory layer, and the most ambitious direction is to develop that layer into a richer structure. The interaction graph records relations between whole interactions. A more expressive memory layer would instead represent the conversation as a knowledge graph whose elements are the contents of the dialogue: topics, entities, decisions, goals, constraints, open questions, and the relations among them. In multi-agent LLM-based systems, such a layer would support memory, organisation, and visualization of long conversations at a finer granularity, and with a closer correspondence to what a conversation is about.

This memory layer can be pursued along two largely independent fronts. The first is the direct continuation of the present work: using the structured memory as a context-retrieval mechanism for agents operating under limited context windows, where the unit of retrieval becomes a relevant subgraph rather than a set of isolated past turns.

The second is the foundation such a layer provides for new forms of interaction and visualization, of which the prototype of Section 5.2 is an early instance. A structured, navigable representation lets users revisit and resume long conversations more effectively than a linear transcript allows, and helps organise the decisions, goals, constraints, and open questions that accumulate across sessions.

This direction raises several open problems: the design of memory architectures for conversational agents, the incremental construction of knowledge graphs from dialogue, the retrieval of the subgraphs most relevant to a query, and the interactive visualization of the thematic structure of conversations. Together they define a programme of work for which this dissertation provides a baseline and a methodology.

5.5 Concluding Remarks

This dissertation proposed and examined a graph-augmented conversational memory architecture for small language models operating under fixed context budgets, and evaluated it in a first controlled study. Its two main conclusions differ in strength. The first is well supported: for such models, the selection of context is decisive. A retrieval layer that chooses the right turns roughly doubled answer quality over a recency baseline that saw the same number of turns (Section 4.2), and investing in selection is the most effective way to make a small model useful over a long conversational history.

The second is more tentative. Discourse structure, added as a third selection signal alongside recency and semantic similarity, is a plausible but unproven signal whose value the standard benchmark could not adjudicate. The study's contribution is to establish why, and to identify the conditions under which it could be.

These findings sit within the broader context that motivated the work. Conversational-memory mechanisms suited to small, locally deployable models are not merely a technical convenience. By improving the usefulness of models that run on owned infrastructure, they contribute, modestly, to a more diverse and resilient technological landscape, one less dependent on a few large external providers, and they help organisations keep sensitive information and operational knowledge under their own control.

This dissertation does not claim to solve the broad challenges of resilience or sovereignty; it explores one architectural component that may serve as an enabling part of conversational systems designed to be more autonomous, locally deployable, and less reliant on external services. As such a component, a graph-based conversational-memory layer is offered here as an initial baseline, and as a foundation on which richer, navigable, and discourse-aware conversational memory can be built.

References

[1] Vibhor Agarwal et al. "A Graph-Based Context-Aware Model to Understand Online Conversations". In: *ACM Transactions on the Web* 18.1 (Nov. 2023), 1–27. ISSN: 1559-114X. DOI: 10.1145/3624579. URL: http://dx.doi.org/10.1145/3624579.

[2] Chenxin An et al. *Why Does the Effective Context Length of LLMs Fall Short?* 2024. arXiv: 2410.18745 [cs.CL]. URL: https://arxiv.org/abs/2410.18745.

[3] Artificial Analysis. *Comparison of Open Source Models*. https://artificialanalysis.ai/models/open-source. Accessed: 3 Feb 2026. 2025.

[4] D. Bristot. *Open source vs proprietary LLMs: Complete 2025 benchmark analysis*. https://whatllm.org/blog/open-source-vs-proprietary-llms-2025. What LLM, accessed February 3, 2026. Oct. 2025.

[5] Maitreyi Chatterjee and Devansh Agarwal. *Semantic Anchoring in Agentic Memory: Leveraging Linguistic Structures for Persistent Conversational Context*. 2025. arXiv: 2508.12630 [cs.CL]. URL: https://arxiv.org/abs/2508.12630.

[6] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. *GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension*. 2020. arXiv: 1908.00059 [cs.CL]. URL: https://arxiv.org/abs/1908.00059.

[7] Hao Fei et al. *Conversational Semantic Role Labeling with Predicate-Oriented Latent Graph*. 2022. arXiv: 2210.03037 [cs.CL]. URL: https://arxiv.org/abs/2210.03037.

[8] Xiachong Feng et al. *Dialogue Discourse-Aware Graph Model and Data Augmentation for Meeting Summarization*. 2021. arXiv: 2012.03502 [cs.CL]. URL: https://arxiv.org/abs/2012.03502.

[9] Milan Gritta, Gerasimos Lampouras, and Ignacio Iacobacci. *Conversation Graph: Data Augmentation, Training and Evaluation for Non-Deterministic Dialogue Management*. 2020. arXiv: 2010.15411 [cs.CL]. URL: https://arxiv.org/abs/2010.15411.

[10] Kelly Hong, Anton Troynikov, and Jeff Huber. *Context Rot: How Increasing Input Tokens Impacts LLM Performance*. Tech. rep. Chroma, 2025. URL: https://research.trychroma.com/context-rot.

[11] Vasile Ionut Remus Iga and Gheorghe Cosmin Silaghi. "LLMs for Knowledge-Graphs Enhanced Task-Oriented Dialogue Systems: Challenges and Opportunities". In: *Advanced Information Systems Engineering Workshops*. Ed. by João Paulo A. Almeida, Claudio Di Ciccio, and Christos Kalloniatis. Cham: Springer Nature Switzerland, 2024, pp. 168–179. ISBN: 978-3-031-61003-5.

[12] Parag Jain and Mirella Lapata. *Conversational Semantic Parsing using Dynamic Context Graphs*. 2023. arXiv: 2305.06164 [cs.CL]. URL: https://arxiv.org/abs/2305.06164.

[13] Cai Ke et al. “Flexibly Utilize Memory for Long-Term Conversation via a Fragment-then-Compose Framework”. In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Ed. by Christos Christodoulopoulos et al. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 21119–21136. ISBN: 979-8-89176-332-6. DOI: 10.18653/v1/2025.emnlp-main.1069. URL: https://aclanthology.org/2025.emnlp-main.1069/.

[14] D. Kuznetsov and Daria Ledneva. “Graph Models for Contextual Intention Prediction in Dialog Systems”. In: *Doklady Mathematics* 108 (Mar. 2024). DOI: 10.1134/S106456242370117X.

[15] Philippe Laban et al. *LLMs Get Lost In Multi-Turn Conversation*. 2025. arXiv: 2505.06120 [cs.CL]. URL: https://arxiv.org/abs/2505.06120.

[16] Paul Langer and Stefan Haag. “AI Sovereignty: Redundant Use of Large Language Models for Public Sector Resilience”. In: *Public Organization Review* (May 2026). DOI: 10.1007/s11115-026-01021-4.

[17] Patrick Lewis et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. arXiv: 2005.11401 [cs.CL]. URL: https://arxiv.org/abs/2005.11401.

[18] Hongmiao Liao et al. “Hierarchical fine-grained state-aware graph attention network for dialogue state tracking”. In: *The Journal of Supercomputing* 81 (Mar. 2025). DOI: 10.1007/s11227-025-07101-4.

[19] Nelson F. Liu et al. *Lost in the Middle: How Language Models Use Long Contexts*. 2023. arXiv: 2307.03172 [cs.CL]. URL: https://arxiv.org/abs/2307.03172.

[20] Xinwei Long, Shuzi Niu, and Yucheng Li. “Position Enhanced Mention Graph Attention Network for Dialogue Relation Extraction”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021). URL: https://api.semanticscholar.org/CorpusID:235792517.

[21] Adyasha Maharana et al. *Evaluating Very Long-Term Conversational Memory of LLM Agents*. 2024. arXiv: 2402.17753 [cs.CL]. URL: https://arxiv.org/abs/2402.17753.

[22] Guanzhong Pan et al. *A Cost-Benefit Analysis of On-Premise Large Language Model Deployment: Breaking Even with Commercial LLM Services*. 2025. arXiv: 2509.18101 [cs.AI]. URL: https://arxiv.org/abs/2509.18101.

[23] Norman Paulsen. *Context Is What You Need: The Maximum Effective Context Window for Real World Limits of LLMs*. 2025. arXiv: 2509.21361 [cs.CL]. URL: https://arxiv.org/abs/2509.21361.

[24] Liang Qiu et al. “SocAoG: Incremental Graph Parsing for Social Relation Inference in Dialogues”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2021, 658–670. DOI: 10.18653/v1/2021.acl-long.54. URL: http://dx.doi.org/10.18653/v1/2021.acl-long.54.

[25] Preston Rasmussen et al. *Zep: A Temporal Knowledge Graph Architecture for Agent Memory*. 2025. arXiv: 2501.13956 [cs.CL]. URL: https://arxiv.org/abs/2501.13956.

[26] Freda Shi et al. *Large Language Models Can Be Easily Distracted by Irrelevant Context*. 2023. arXiv: 2302.00093 [cs.CL]. URL: https://arxiv.org/abs/2302.00093.

[27] Yue Tan et al. “Guiding Dialogue Agents to Complex Semantic Targets by Dynamically Completing Knowledge Graph”. In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 6506–6518. DOI: 10.18653/v1/2023.findings-acl.407. URL: https://aclanthology.org/2023.findings-acl.407/.

[28] Nicholas Thomas Walker, Stefan Ultes, and Pierre Lison. *GraphWOZ: Dialogue Management with Conversational Knowledge Graphs*. 2022. arXiv: 2211.12852 [cs.CL]. URL: https://arxiv.org/abs/2211.12852.

[29] Fali Wang et al. *A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness*. 2024. arXiv: 2411.03350 [cs.CL]. URL: https://arxiv.org/abs/2411.03350.

[30] Xinlin Wang and Mats Brorsson. *Rethinking Scale: Deployment Trade-offs of Small Language Models under Agent Paradigms*. 2026. arXiv: 2604.19299 [cs.CL]. URL: https://arxiv.org/abs/2604.19299.

[31] Han Wu, Kun Xu, and Linqi Song. *CSAGN: Conversational Structure Aware Graph Network for Conversational Semantic Role Labeling*. 2021. arXiv: 2109.11541 [cs.CL]. URL: https://arxiv.org/abs/2109.11541.

[32] Zhicheng Yan, Joel E Fischer, and Jeremie Clos. “Remembering Things Makes Chatbots Sound Smarter, but Less Trustworthy - A Pilot Study”. In: *Proceedings of the 7th ACM Conference on Conversational User Interfaces*. CUI ’25. New York, NY, USA: Association for Computing Machinery, 2025. ISBN: 9798400715273. DOI: 10.1145/3719160.3737617. URL: https://doi.org/10.1145/3719160.3737617.

[33] Haoyu Zhang et al. “Multimodal Dialog System: Relational Graph-based Context-aware Question Understanding”. In: *Proceedings of the 29th ACM International Conference on Multimedia*. MM ’21. Virtual Event, China: Association for Computing Machinery, 2021, 695–703. ISBN: 9781450386517. DOI: 10.1145/3474085.3475234. URL: https://doi.org/10.1145/3474085.3475234.

[34] Zhuosheng Zhang, Junlong Li, and Hai Zhao. *Multi-turn Dialogue Reading Comprehension with Pivot Turns and Knowledge*. 2021. arXiv: 2102.05474 [cs.CL]. URL: https://arxiv.org/abs/2102.05474.

[35] Sizhe Zhou and Jiawei Han. *A Simple Yet Strong Baseline for Long-Term Conversational Memory of LLM Agents*. 2025. arXiv: 2511.17208 [cs.CL]. URL: https://arxiv.org/abs/2511.17208.