



BANCA D'ITALIA
EUROSISTEMA

Questioni di Economia e Finanza

(Occasional Papers)

Can hybrid semantic-graph RAG bridge retrieval
and reasoning in Economic Bulletin drafting?

by Giuseppe Bruno

October 2026

Number

1071



BANCA D'ITALIA
EUROSISTEMA

Questioni di Economia e Finanza

(Occasional Papers)

Can hybrid semantic-graph RAG bridge retrieval
and reasoning in Economic Bulletin drafting?

by Giuseppe Bruno

Number 1071 – October 2026

The series Occasional Papers presents studies and documents on issues pertaining to the institutional tasks of Banca d'Italia and the Eurosystem. The Occasional Papers appear alongside the Working Papers series which are specifically aimed at providing original contributions to economic research.

The Occasional Papers include studies conducted within Banca d'Italia, sometimes in cooperation with the Eurosystem or other institutions. The views expressed in the studies are those of the authors and do not involve the responsibility of the institutions to which they belong.

The series is available online at www.bancaditalia.it .

CAN HYBRID SEMANTIC-GRAPH RAG BRIDGE RETRIEVAL AND REASONING IN ECONOMIC BULLETIN DRAFTING?

by Giuseppe Bruno*

Abstract

Drafting a comprehensive short-term macroeconomic report requires combining institutional sources, market information, statistical releases and expert knowledge of the macroeconomic landscape. For the purposes of Banca d'Italia's Economic Bulletins, fluency is not sufficient: the drafting process must also ensure factual accuracy, traceability, consistency and reproducibility. Although large language models can accelerate macroeconomic writing, they remain exposed to hallucinations, unsupported claims and weak grounding in verifiable evidence, limiting their use in institutional communication. This paper presents a reproducible retrieval-augmented generation pipeline for Economic Bulletin drafting. The framework combines a semantic vector store with a Neo4j knowledge graph and compares vector, graph and hybrid retrieval modes across chapter-level drafting tasks. The pipeline separates evidence collection, normalization, planning, deterministic drafting, verification, semantic support evaluation, optional open-source refinement and final LaTeX assembly. The results show that vector retrieval is the strongest baseline in most chapters, graph retrieval becomes reliable after schema-aware fallback, and hybrid retrieval is promising but does not automatically dominate under the current selection policy. The contribution is an auditable architecture for controlled, evidence-grounded economic drafting.

JEL Classification: O33, D83, C88.

Keywords: language models, retrieval-augmented generation, vector store, knowledge graph.

DOI: 10.32057/0.QEF.2026.1071

* Banca d'Italia.

The views expressed in this paper are those of the author only and do not necessarily reflect the official position of Banca d'Italia. giuseppe.bruno@bancaditalia.it

1 Introduction

Indeed it required the accident to be repeated before my attention was thoroughly aroused to the remarkable fact that seedlings of self fertilised parentage are inferior[...] in height and vigour to seedlings of cross fertilised parentage.

Charles Darwin, *The Effects of Cross and Self Fertilisation in the Vegetable Kingdom*. 1876

In their routine activities, central banks communicate with the public through different tools. Economic bulletins are flagship publications periodically released that provide comprehensive analyses of monetary, financial, and macroeconomic developments forming the basis for policy decisions. These bulletins express central banks’ perspectives on short-term economic developments.

Accountability and transparency entail the adoption of factual accuracy, traceability, consistency across chapters, and repeatability over time. Pursuing these targets calls for constant improvement, which today can be achieved through the savvy use of a range of Artificial Intelligence (AI) tools.

1.1 Why LLMs alone are insufficient.

Large language models (LLMs) have shown remarkable potential in enhancing drafting productivity across a wide range of different scenarios (see Yang and Mark Christopher, 2020 and Wu et al., 2023). Generative AI tools have shown how to significantly accelerate text production, reduce the cognitive burden of routine writing tasks, and improve the consistency of drafted outputs (Brynjolfsson et al., 2023 and Noy and Whitney, 2023). In particular, in Noy and Whitney, 2023, the authors offer experimental evidence that LLM-assisted writing raises the productivity of professional workers by reducing the average task completion time while keeping constant or, sometimes, improving the output quality. However, central bank communication and policy-oriented drafting operate under constraints that make unsupported text generation — where the model produces content not grounded in verifiable source documents — fundamentally unsuitable. Central bank publications must satisfy stringent requirements of factual accuracy, institutional consistency, and auditability (Blinder et al., 2008). Hallucination, a well-documented failure mode of LLMs whereby the model generates plausible but factually incorrect or unsupported statements, for example Ji et al., 2024 and Maynez et al., 2020, poses an unacceptable risk in this context. As highlighted in Araujo, Bruno, et al., 2026, the deployment of generative AI in high-stakes institutional settings requires robust grounding mechanisms to ensure that outputs remain traceable to authoritative sources.

1.2 RAG as a controlled generation framework.

Retrieval-Augmented Generation (RAG), originally put forward in Lewis et al., 2021, addresses this limitation by conditioning text generation on explicitly retrieved source passages, thereby anchoring model outputs to verifiable document corpora (see Yang and Mark Christopher, 2020). This architecture is particularly well-suited to central bank bulletin production, where each analytical claim must be traceable to underlying data, internal research, or official policy stances.

1.3 Paper’s contribution to the literature.

This paper makes two main methodological contributions and adds a production-oriented resilience feature.

First, to the best of our knowledge, it is the first to propose a hybrid retrieval architecture combining a semantic vector database and a knowledge graph for the automated drafting of institutional Economic Bulletins, extending the Graph RAG framework (see Edge et al., 2024) to a central bank communication context. In this benchmark, reasoning is assessed indirectly through the ability of graph and hybrid retrieval to recover relational evidence and support multi-hop evidence chains, rather than through a separate formal reasoning benchmark. Second, it introduces a chapter-level evaluation protocol inspired by RAGAS-style dimensions such as Faithfulness, Contextual Relevance, and Context Recall (Es et al., 2024), adapted to the requirements of Economic Bulletin drafting through deterministic and auditable proxy metrics. This disaggregated evaluation enables targeted quality diagnostics across the heterogeneous analytical chapters that characterize central bank publications. Finally, to ensure pipeline resilience in production, the data ingestion layer incorporates a URL fallback mechanism that automatically switches to a pre-specified mirror source upon receiving HTTP 4xx or 5xx error codes, following standard fault-tolerant design principles for data pipelines (for example Kleppmann, 2017) .

1.4 Structure of the paper.

The rest of the paper is organized as follows. Section 2 reviews the recent literature on LLMs, RAG and knowledge-based retrieval for economic and financial applications. Section 3 introduces the general RAG framework and discusses its relevance for institutional economic reporting. Section 4 describes vector stores and dense retrieval. Section 5 presents graph databases and Graph RAG, with particular attention to the Neo4j knowledge graph used in the project. Section 6 introduces the hybrid semantic-graph architecture. Section 7 describes the empirical Economic Bulletin pipeline, from source configuration to LaTeX assembly. Section 8 presents the evaluation framework and interprets the comparison across vector, graph and hybrid retrieval modes. Section 9 concludes with the main findings, limitations and directions for

future work.

2 Recent literature.

Economists and financial analysts increasingly rely on large volumes of textual information, including news articles, geopolitical reports, earnings releases, statistical publications, policy documents and macroeconomic reports. These sources contain information that is relevant for monitoring economic developments, interpreting market conditions and producing timely analytical assessments. However, because much of this material is unstructured, heterogeneous and institution-specific, classical data-analysis methods are often insufficient for extracting and organizing the information required for policy-oriented writing.

Large language models (LLMs) have substantially improved the ability to process, summarize and generate text. Following the transformer architecture introduced by Vaswani et al., 2017, LLMs have shown strong capabilities in language understanding, information extraction, sentiment analysis, automated reporting and financial communication tasks (Brown et al., 2020 and Yang and Mark Christopher, 2020 and Wu et al., 2023). In economic and financial applications, recent studies have explored their ability to extract information from reports, analyze macroeconomic documents and support decision-making workflows (Leland et al., 2025 and Yuqi, Kong, et al., n.d.). These developments suggest that LLMs can improve productivity in document-intensive analytical environments.

At the same time, important limitations remain. General-purpose LLMs may struggle with specialized economic documents that fall outside their training distribution, and they may generate plausible but unsupported statements. This risk is particularly relevant in institutional settings, where analytical outputs must be accurate, auditable and consistent with official sources. The literature on hallucination and factuality in natural language generation highlights that unsupported generation can be problematic even when the resulting text is fluent (Maynez et al., 2020 and Ji et al., 2023). In macroeconomic and policy-related applications, these limitations are amplified by the complexity of the source material: economic documents often combine quantitative indicators, forecasts, qualitative judgments, institutional terminology and evolving analytical narratives.

Retrieval-Augmented Generation (RAG) has emerged as a way to mitigate these weaknesses by conditioning generation on external evidence retrieved from a document repository (Lewis et al., 2021). Rather than relying only on the parametric knowledge of the language model, RAG systems retrieve relevant passages at inference time and insert them into the generation context. This can improve factual grounding, reduce unsupported statements and make the evidence behind generated text more inspectable. Traditional VectorRAG approaches use dense semantic embeddings to retrieve passages

that are close in meaning to the user query ¹, even when the wording differs across documents (Izacard and Edouard, 2021, Guu et al., 2020 and Huang and Huang, 2024). This makes vector retrieval attractive in economic applications, where the same concept may be expressed through different terminology across institutions, reports and time periods.

However, vector retrieval alone does not explicitly represent the relationships among economic concepts, indicators, institutions, sectors and time periods. This matters for Economic Bulletin drafting. A bulletin is not a generic summary of current information: it is an institutional document that must connect new data releases with historical patterns, previous assessments, official statistics, forecasts and established analytical language. Assessments of the Italian economy, for example, may draw on information from ISTAT, the European Central Bank, Eurostat, the OECD, the IMF, the World Bank and previous Economic Bulletins. These sources differ in frequency, scope, terminology and analytical emphasis. As a result, drafting a new bulletin requires both current textual evidence and a historically coherent representation of how concepts and sources are connected.

This is where hybrid semantic-graph RAG becomes relevant (see Singh et al., 2025 and Nagori et al., 2025 for references and examples). The semantic component retrieves text passages that are contextually close to the current analytical question, even when wording differs across sources. The graph component complements this by representing explicit relationships among documents, chunks, entities, indicators, institutions, sectors, policy themes and narrative claims. In this way, hybrid RAG combines two forms of retrieval: dense semantic similarity for recovering relevant textual evidence, and graph-based traversal for recovering relational context see, for example, Sarmah et al., 2024. This combination is particularly useful when the drafting task requires not only finding passages about a topic, but also understanding how that topic is connected to other economic variables, institutional sources or past interpretations.

Knowledge graphs have been proposed as a way to organize heterogeneous information into explicit data structures in which entities and relationships can be directly queried (Hogan et al., 2021). Recent Graph RAG approaches extend this idea to retrieval-augmented generation by using graph structure to support query-focused summarization and reasoning over connected information (Edge et al., 2024). In the context of Economic Bulletin drafting, a knowledge graph can help preserve part of the institutional memory contained in previous bulletins: which indicators were associated with which developments, how risks were framed, which institutions were cited, and how recurring themes such as inflation persistence, labour market resilience, credit conditions or external demand were linked across time.

The present paper contributes to this literature by applying hybrid semantic-graph retrieval to the production of institutional macroeconomic prose. Unlike applications focused mainly on question answering or enterprise document

¹For text we can use either a dense/semantic or a sparse/lexical representation

search, the task considered here is the construction of historically informed, evidence-grounded analytical text. The objective is not to replace expert judgment, but to support economists by reducing search frictions, improving access to institutional memory, and making the drafting process more traceable and reproducible. In this sense, hybrid RAG is treated not merely as a search enhancement, but as a knowledge-management infrastructure for controlled economic drafting.

3 Retrieval-Augmented Generation: general framework

3.1 RAG Definition

Retrieval-Augmented Generation (RAG) is a framework that combines information retrieval with large language models (LLMs) in order to improve the factual grounding and contextual relevance of generated responses. Instead of relying exclusively on the parametric knowledge stored in model weights during pretraining, RAG supplements generation with information retrieved at inference time from an external and reliable knowledge base (Lewis et al., 2021 and **Gao2023**). In this way, the language model conditions its answer on domain-specific and potentially more up-to-date evidence than would otherwise be available from its internal parameters alone, see for example Huang and Huang, 2024 .

The first step in building a RAG pipeline consists in producing a document repository containing relevant domain specific material, such as statistical releases, economic reports, historical bulletins, policy documents, and other institutional publications. These documents are then chunked into smaller units which are indexed so that they can be efficiently retrieved in response to a user query (Y. G. Gao et al., 2024). When a query is submitted, the system retrieves a user specified number of relevant passages through lexical, dense semantic, or hybrid retrieval strategies, and injects the selected evidence into the original prompt provided to the LLM.

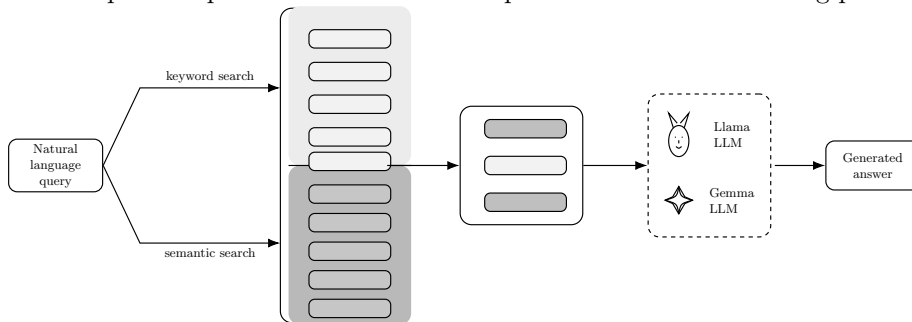
The effectiveness of RAG heavily depends on the quality of the retrieval phase. If the retrieved evidence is relevant, complete, and accurate, the generation model can produce responses that are better grounded in the source material and less dependent on unsupported inference. For this reason, RAG helps in mitigating hallucinations, improving factual consistency, and increasing the traceability of generated outputs by making the supporting evidence observable and inspectable (Huang and Huang, 2024). These properties are especially valuable in document-intensive institutional settings, such as economic bulletin drafting, where accuracy, reproducibility, and alignment with authoritative sources are essential requirements.

3.2 The three RAG foundational pillars.

There are three basic pillars which define a RAG pipeline:

Retrieval -> Augmentation -> Generation

The temporal sequence of the different steps is shown in the following picture:



A RAG system can be understood as resting on three closely related pillars: *retrieval*, *augmentation*, and *generation*. The first pillar, retrieval, identifies the external evidences most relevant to a user query by searching a knowledge base through lexical, dense semantic, or hybrid methods; its role is to supply the system with domain-specific and potentially up-to-date information that is not guaranteed to be fully represented in the model’s parameters (Lewis et al., 2021). The augmentation transforms the retrieved evidence into a form that can effectively guide the model by selecting, reranking, filtering, or structuring the most informative passages before they are inserted into the model context; this stage is crucial because the quality, ordering, and relevance of the supplied evidence strongly affect downstream answer quality see for example Batthula et al., 2025. The third pillar, generation, is the last step in which the large language model produces the final output conditioned on both the original query and the augmented evidence, thereby combining parametric language competence with non-parametric external knowledge. Taken together, these three components explain why RAG is not merely a retrieval layer attached to an LLM, but a coordinated architecture in which evidence selection, integration, and response generation jointly determine factual grounding, robustness, and traceability.

3.3 RAG in institutional economic reporting.

The adoption of Retrieval-Augmented Generation is especially relevant for institutions such as central banks, where analytical outputs must satisfy high standards of accuracy, transparency, and accountability. Economic Bulletins, forecasts, and policy assessments are expected to be supported by verifiable evidence and to remain consistent with established institutional positions. By grounding generated text in retrieved documents, RAG provides a mechanism

for linking narrative statements to identifiable sources, thereby improving the explainability of the drafting process see, for example NIST, 2024.

In addition, RAG naturally supports governance requirements. Retrieved documents, retrieval parameters, prompts, and generated outputs can all be logged and archived, enabling reproducibility and facilitating ex-post reviews. Such features are particularly important in environments where analytical products may be scrutinized by policymakers, researchers, market participants, or internal audit functions.

4 Vector stores and dense retrieval

In knowledge-intensive tasks such as domain specific texts or question answering, LLMs often struggle to generate factual answers, relying solely on their internal parametric knowledge². Classical approaches based on the frequency of different words such as tf-idf or BM25³ are heavily affected by the Vocabulary mismatch phenomenon: they yield good results if and only if there is exact overlap of words between the query and document set (for example see Furnas et al., 1987). In other words, the user asking a question needs to guess exactly what words the writer of the possible answer might have used. In our application, this dense-retrieval layer is implemented with the open-source vector database ChromaDB, which stores embeddings of document chunks and supports similarity search over chapter-level retrieval queries. Dense retrieval is based on the idea that documents, chunks, or other evidence items can be represented as continuous vectors, or *embeddings* in a high-dimensional space⁴. In this representation, semantic similarity is captured geometrically: items that are close in meaning tend to be mapped to nearby points, so that retrieval can be performed by searching for evidence whose embedding is close to that of the query or of a chapter-level thematic prompt (see for example A. K. Gao, 2023).

Compared with purely lexical retrieval, dense representations are designed to

²Based on weights estimated during pre-training.

³Best Match 25, introduced by Robertson and Spärk, 1976.

⁴Considering the following two statements:

1. *"In the first quarter of 2026, headline inflation in Italy declined while core inflation remained more persistent."*
2. *"During 2026Q1 in Italy, overall consumer price growth eased, whereas underlying inflation showed greater stickiness."*

In a dense retrieval system, each statement is mapped by an embedding model into a vector in $\mathbb{R}^D$ with, for example $D = 768$.

Although the two sentences do not use exactly the same wording, they express nearly the same economic content: both refer to Italy, the first quarter of 2026, headline inflation, and core inflation persistence. For this reason, their embeddings are expected to lie close to each other in the vector space. Using cosine similarity,

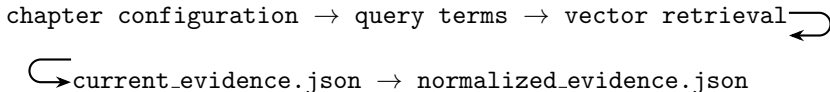
$$\cos(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1| \cdot |v_2|},$$

one would typically obtain a value close to 1 indicating strong semantic similarity. In retrieval terms, this means that if a query concerns headline and core inflation in Italy in early 2026, both statements are likely to be retrieved as relevant evidence even though they differ lexically.

capture semantic relatedness even when the wording of the query and that of the source text do not match exactly, which makes them particularly attractive in knowledge-intensive settings where relevant evidence may be expressed through paraphrase or domain-specific terminology (see for example Lewis et al., 2021 and Huang and Huang, 2024).

4.1 Vector RAG workflow

In the present project, vector retrieval operates through a relatively simple but effective workflow. The process starts from a chapter configuration that defines the scope of the section and the associated retrieval terms. These terms are then used to query the vector store, which returns the chunks that are scored most semantically relevant. The retrieved passages are written into a json file: `current_evidence.json` and subsequently consolidated in `normalized_evidence.json`, where they are transformed into a more structured set of support items for drafting and verification. In schematic form, the workflow can be summarized as:



Because the evidence returned by vector retrieval is typically textual, it integrates naturally with downstream drafting and semantic verification. In other words, vector mode tends to produce support material that is already close to the linguistic form required by the drafting layer, thereby reducing the transformation needed between retrieval and generation (Gao et al., 2023; Huang and Huang, 2024).

4.2 Strengths and weaknesses of vector retrieval

Vector retrieval is particularly effective when the relevant evidence is expressed in text that is semantically close to the target topic. Its main advantage is that it can recover passages that are topically relevant even when they do not share exact keywords with the query, thereby improving recall in domains where similar concepts are often expressed in slightly different ways (Lewis et al., 2020; Gao et al., 2023). In the benchmark developed for this project, vector mode emerged as the strongest overall baseline in most chapters. A plausible reason is that many sections of the Economic Bulletin depend heavily on rich textual evidence drawn from surveys, statistical releases, institutional notes, and bulletin-style narratives. In such settings, dense retrieval provides evidence that is both semantically informative and highly traceable, which in turn supports drafting and lexical verification with comparatively little additional processing.

Despite these strengths, vector retrieval also has important limitations. First, semantic proximity does not always imply structural relevance: a vector re-

triever may return passages that are broadly similar in topic but not sufficiently aligned with the precise institutional, causal, or comparative structure required by the drafting task (Gao et al., 2023; Huang and Huang, 2024). Second, dense retrieval does not explicitly encode relationships among entities unless those relationships are expressed in the retrieved text itself. As a result, it may miss evidence that depends on links among institutions, indicators, events, or concepts when those links are distributed across multiple passages rather than stated directly in one chunk. More generally, vector retrieval is not designed to represent causal, institutional, or conceptual structure in an explicit way; it can only recover such structure indirectly, to the extent that it is embedded in the surface text of the retrieved evidence. This is precisely the type of limitation that motivates comparison with graph-based and hybrid retrieval architectures.

4.3 Empirical role in the Economic Bulletin pipeline

From an empirical perspective, vector retrieval was usually the strongest baseline in the Economic Bulletin pipeline. Its performance was particularly strong in chapters where the target section could be grounded directly in textual evidence from surveys, statistical releases and institutional publications. In these cases, dense retrieval supplied coherent and content-rich support material that transferred naturally into the drafting layer and remained easy to inspect during verification.

This result suggests that vector retrieval is especially effective when the evidentiary structure of a chapter is already reflected in the wording of the source documents. At the same time, the observed limits of vector mode in more relational or structurally complex cases provide part of the motivation for testing graph-based and hybrid semantic-graph configurations alongside the vector baseline. Vector evidence is well suited to drafting and semantic verification because it is text-rich, but it does not explicitly represent the institutional, causal or conceptual relationships that may connect evidence across documents, entities and time.

5 Graph databases and Graph RAG

5.1 Knowledge graphs and multi-hop reasoning

A knowledge graph represents information as a set of nodes connected by relationships. Figure 1 provides a stylized example of causal relationships among macroeconomic variables. More generally, a graph database stores informa-

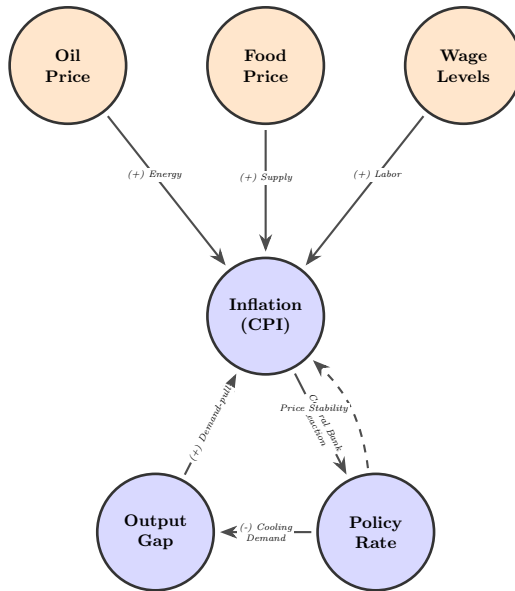


Figure 1: Macroeconomics Causal relationships

tion in a form where entities, documents, concepts, events, indicators or text fragments are represented as nodes, while the links among them are represented as relationships. This structure makes it possible to query not only individual pieces of information, but also the paths that connect them. In this sense, a graph database is particularly well suited to settings where the analytical value lies not only in isolated documents, but also in the connections among sources, concepts and evidence. One of the main advantages of graph-based RAG over pure vector search is its ability to support multi-hop reasoning. As a concrete example in the macroeconomic domain let us assume we need to answer the following question: *Did the monetary policy stance of the ECB in 2022 influence sovereign spread volatility in high debt/GDP ratio countries?*

No single chunk/document is sufficient to answer this question because the answer requires chaining multiple inference steps across different pieces of knowledge. The Reasoning Chain for this question can be broken down in four hops.

Hop 1 Policy identification The model retrieves that the ECB began its hiking cycle in July 2022, raising the deposit facility rate by 50bps — the first hike in 11 years.

Hop 2 Mechanism linkage

It then connects this to the transmission mechanism: rising rates increase the discount rate applied to sovereign debt, disproportionately affecting high-debt periphery countries (Italy, Greece, Spain).

Hop 3 Instrument activation

It further retrieves that the ECB announced the TPI (Transmission Protection Instrument) in July 2022 in response to concerns about fragmentation and widening sovereign spreads, thereby linking the policy environment to the spread reaction.

Hop 4 Quantitative grounding

Finally it retrieves that the Italy–Germany 10Y spread peaked at 250bps in June 2022 before TPI announcement compressed it, providing quantitative support for the directional claim.

Knowledge graphs have been widely discussed as a way to organize heterogeneous information into explicit graph-based data models, making relationships among entities and statements directly queryable see Hogan et al., 2021 for example.

This example also clarifies the limitation of vector search alone. A pure semantic search may retrieve documents about ECB policy and documents about BTP spreads, but it does not explicitly traverse the causal chain connecting them. Each hop requires resolving an entity or relationship that becomes relevant only after the previous step has been identified.

Figure 2 illustrates this logic as a simplified multi-hop graph traversal. Each hop corresponds to a link between entities or concepts, which is precisely where a knowledge graph adds value over a flat semantic vector store such as ChromaDB.

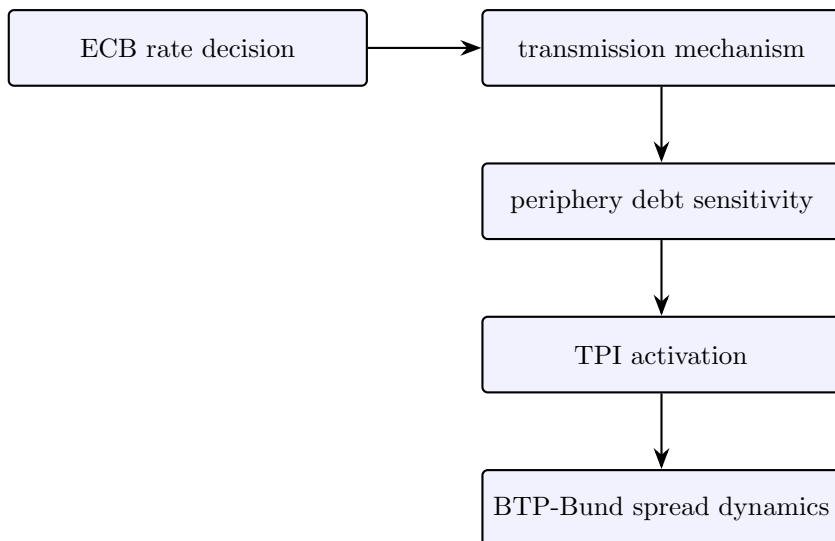


Figure 2: Multi-hop graph path linking ECB policy to sovereign spread dynamics

5.2 Neo4j representation in the Economic Bulletin project

In this project, the knowledge graph was implemented as a Neo4j graph database and populated from the corpus of past Economic Bulletin material. The graph contained nodes such as **Chunk**, **Document** and **Entity**. A **Document** node represents a source document or publication; a **Chunk** node represents a smaller textual unit extracted from a document; an **Entity** node represents an economic concept, institution, indicator, geographic area or other named object mentioned in the corpus. This representation differs from a purely text-based corpus. In a standard document collection, the basic retrieval unit is usually a passage or a document. In the graph, the retrieval unit can also be a relationship. This means that evidence can be recovered not only because a passage contains certain words, but also because it is connected to an entity, because it participates in a relation, or because it belongs to a document that reports a relevant economic development. For an Economic Bulletin corpus, this is an important distinction. The same themes recur across time — inflation persistence, labour market resilience, external demand, credit conditions, financial fragmentation, wage pressures, or fiscal support — and a graph representation can preserve part of this analytical structure by connecting recurring concepts and evidence chunks across publications. In schematic form, the graph-retrieval workflow can be summarized as:

chapter configuration → graph keywords and entities → graph-derived evidence
→ `current_evidence.json` → `normalized_evidence.json`

During the first implementation, however, graph retrieval exposed an important practical problem. For most chapters, graph mode did not return substantive evidence. Instead, it returned an empty placeholder item: `kg_empty_0001` — Knowledge Graph retrieval returned no edges.

This occurred for 9 of the 11 chapters. As a result, graph-based generation for those chapters was artificially weak. Only one placeholder-like evidence item was selected, and the downstream verification process could not assess most of the generated content against real evidence. In particular, four paragraph blocks became “not checkable”, because they had no meaningful graph-derived evidence attached to them. This produced artificially low faithfulness scores, not because graph retrieval was intrinsically unsuitable for the task, but because the retrieval query was failing to match the actual graph schema. The diagnosis showed that the problem was a schema mismatch. The original graph retrieval query assumed that the Neo4j graph database contained properties such as `chapter_id`, `quarter` or `geography`. These fields were natural from the point of view of the Economic Bulletin generation pipeline, because the vector and manifest-based layers were organized by chapter, time period and target geography. However, the existing Neo4j graph had been created from the underlying knowledge-graph extraction pipeline and did not contain a `chapter_id` property. Its actual schema used a different set of properties, including fields such as `name`, `title`, `text`, `evidence`, `predicate`, `source_file`, `file_name` and `doc_id`. Consequently, queries that filtered directly on `chapter_id`, `quarter` or `geography` could return no rows even when the graph contained relevant information. The issue was therefore not the absence of graph evidence, but the use of retrieval predicates that did not correspond to the stored graph structure.

The fix was to make graph retrieval schema-aware while preserving backward compatibility. The original graph query was kept unchanged as the first retrieval attempt. This was important because two chapters were already working correctly under the initial graph retrieval logic, and the objective was not to disrupt successful cases. A fallback retrieval query was then introduced: if the original query returned zero rows, the system searched across the actual node and relationship properties available in the graph. Instead of relying only on missing fields such as `chapter_id`, the fallback searched over properties such as `name`, `title`, `text`, `evidence`, `predicate`, `source_file`, `file_name` and `doc_id`. In practical terms, the fallback allowed the retrieval layer to match chapter-specific themes, keywords and entities against the schema that was truly present in Neo4j. This made the graph layer more robust without requiring a reload or redesign of the underlying knowledge graph.

The empirical effect of this change was substantial. After the schema-aware fallback was introduced, graph retrieval became healthy across all chapters.

Graph mode no longer selected only the empty placeholder for the previously failing chapters; instead, it returned real graph-derived evidence items connected to the underlying Economic Bulletin corpus. This improved the downstream generation and verification process. Faithfulness scores increased to high values because generated statements could now be checked against actual graph evidence. Paragraph blocks that had previously been “not checkable” became verifiable, and all graph runs passed verification. Once retrieval was aligned with the actual graph structure, the graph-based pipeline provided a meaningful relational evidence layer and became a viable component of the hybrid RAG system.

6 Hybrid RAG

6.1 Hybrid retrieval

In this project, hybrid RAG refers to a retrieval-augmented generation system blending two retrieval frameworks: vector-based semantic retrieval and graph-based relational retrieval. The vector component, based on ChromaDB, retrieves text passages that are semantically close to the user query or chapter configuration. It is designed to capture textual similarity even when the wording differs across documents. The graph component retrieves evidence from a knowledge graph hosted on the open source Neo4j graph database, using explicit relationships among documents, chunks, entities and economic concepts. In this sense, hybrid RAG is a system which combines two different retrieval logics. Figure 3 provides a schematic representation of this hybrid retrieval scheme.

The first retrieval logic is semantic. It treats the corpus as a collection of textual passages and uses embeddings to identify passages that are close in meaning to the query. This is useful when the relevant evidence is expressed in different language across sources or time. For example, a query about inflation persistence may retrieve passages referring to underlying inflation, domestic price pressures, wage dynamics or services inflation, even when the exact phrase “inflation persistence” is not used.

The second retrieval logic is relational. It treats the corpus as a graph of connected objects. Documents, chunks and entities are represented as nodes; economic, institutional or textual connections are represented as relationships. Graph retrieval can therefore identify evidence not only because it is textually similar to the query, but also because it is connected to relevant concepts, documents, entities or predicates. For example, a passage may be retrieved because it is linked to an entity such as “credit conditions”, because it is part of a document that reports financial-market developments, or because it participates in a relationship such as ‘FORECASTS’, ‘CAUSES’ or ‘MENTIONS’. Hybrid retrieval combines these two layers. In the implementation used for this benchmark, the hybrid mode collected evidence from both the vector pipeline and the graph pipeline, then passed the combined evidence set to the

Pipeline Setup

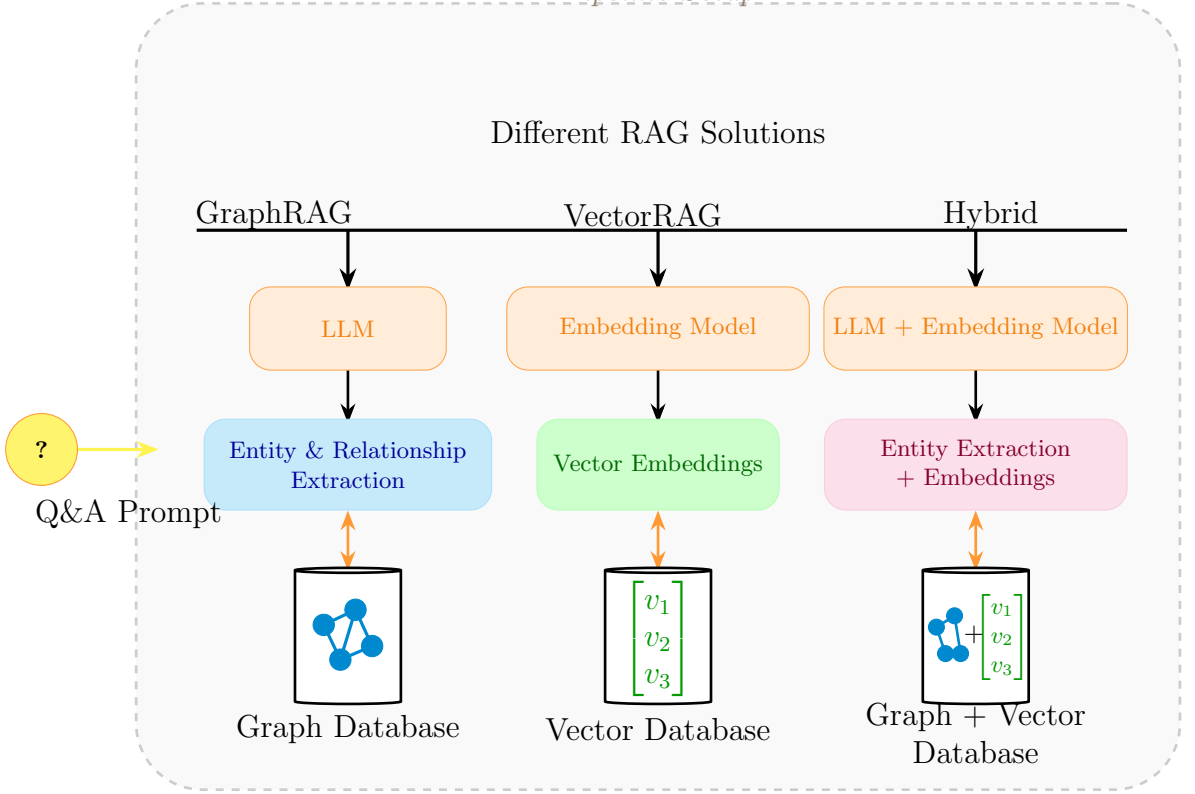


Figure 3: Overview of GraphRAG, vector RAG and hybrid RAG, showing the main components used to transform a query into graph-based, vector-based or combined retrieval structures.

normalization, planning, drafting and verification stages. The objective was to test whether the combination of semantic similarity and graph structure could provide a richer evidence base for generating Economic Bulletin chapters.

6.2 Expected benefits

Hybrid retrieval combines the strengths of vector and graph retrieval. Vector retrieval contributes textual richness. It is effective at finding detailed passages, statistical explanations, institutional commentary and source excerpts that are semantically close to the chapter topic. This is particularly useful for drafting, because the language model needs enough textual material to formulate precise and fluent economic narrative. Vector evidence often contains complete sentences, contextual descriptions and source-specific terminology that can be directly used to support the generated text.

Graph retrieval contributes relational structure. It can surface explicit links among concepts, entities, sources and chunks. This is useful when the relevant information is not only a matter of semantic similarity, but also of connection. In an Economic Bulletin corpus, many analytical ideas recur across publications: inflation persistence, external demand weakness, labour market resilience, credit conditions, wage growth, financial-market tensions and policy transmission. A graph can help recover how these concepts are connected to each other and to specific documents or evidence chunks. This can support historical continuity and help the system identify relationships that a purely vector-based approach may not emphasize.

The theoretical appeal of hybrid RAG is therefore clear. Vector retrieval answers the question: “Which passages are semantically close to the topic?”

Graph retrieval adds: “Which concepts, sources and evidence chunks are explicitly connected to the topic?” In principle, a hybrid system should retrieve both the rich textual context needed for drafting and the structured relationships needed for analytical coherence. For a central-bank publication, this combination is attractive because the final text must be both grounded in official evidence and consistent with an established analytical tradition.

In the benchmark, hybrid retrieval generally behaved well in terms of faithfulness and traceability. Once both vector and graph collection were working correctly, the hybrid mode usually provided a broad evidence base and supported generated chapters with identifiable evidence markers. The verification pipeline was able to trace generated statements back to retrieved evidence, and hybrid runs often achieved high faithfulness. This indicates that combining semantic and graph retrieval can produce drafts that remain well grounded in the available evidence.

However, hybrid retrieval did not systematically dominate vector retrieval in the overall retrieval-quality score. This result is important because it shows that adding graph evidence does not automatically improve measured retrieval quality. A hybrid evidence pool may be more diverse and more structurally informative, but the final benchmark score also depends on how evidence is selected, planned, cited and evaluated downstream. The next subsection discusses why this occurred in the present implementation.

6.3 Why hybrid did not dominate

In the current benchmark, the planner imposes a fixed evidence-selection cap, often equal to 24 selected evidence items. This means that, regardless of how many candidate items are retrieved, only a limited number can be used for drafting. When hybrid retrieval adds graph-derived evidence to the candidate pool, those graph items compete with vector items for the same selection budget.

This creates a displacement effect. Vector evidence often contains richer textual excerpts: complete source passages, official descriptions, statistical context and narrative language. Graph evidence, by contrast, may be shorter, more relational or more fragmentary. It can be valuable for connecting con-

cepts, entities and sources, but it may be less directly useful for drafting a complete paragraph. Under a fixed evidence cap, graph evidence can therefore displace text-rich vector evidence that would have scored well under lexical relevance, citation coverage or answer-support metrics.

This explains why hybrid retrieval may be more diverse without necessarily achieving a higher score under the current evaluation function. The hybrid pool contains both semantic and relational signals, but the planner treats all candidate evidence items as undifferentiated competitors. If the selection policy does not explicitly balance retrieval types, the final selected set may either contain too much graph evidence at the expense of textual richness, or too much vector evidence to exploit the graph layer effectively. In both cases, the theoretical advantage of hybrid retrieval can be weakened by the evidence-selection mechanism.

A second issue is that the scoring function may reward textually aligned evidence more strongly than relationally useful evidence. Metrics based on lexical overlap, paragraph-level support and citation coverage naturally favour evidence items that contain rich, directly quotable passages. Graph evidence may help organize the argument by identifying relationships among entities, themes and documents, but this contribution is harder to capture if the evaluation mainly compares the final cited text with textual excerpts.

The benchmark result therefore does not imply that hybrid RAG is inferior to vector RAG. Rather, it suggests that the current hybrid implementation is still simple. It combines vector and graph evidence in a common pool, but it does not yet optimize their interaction. A more mature hybrid policy should treat vector and graph evidence as complementary rather than as competitors for the same evidence slots.

Future versions should therefore introduce retrieval-type-aware selection. One option is to assign separate evidence budgets, for example reserving a minimum share for vector evidence and a minimum share for graph evidence, with the remaining slots allocated by relevance score. A second option is to assign different roles to the two retrieval types: vector evidence can serve primarily as drafting evidence, while graph evidence can support planning, entity coverage and relational continuity. A third improvement would be graph-conditioned vector retrieval, where graph results identify relevant entities, predicates or neighbouring chunks that are then used to refine the vector query.

Overall, the benchmark suggests that hybrid RAG is promising, but its value lies in retrieving different kinds of evidence and not simply in retrieving more evidence. Vector retrieval supplies semantic and textual richness; graph retrieval supplies relational and conceptual structure. A mature hybrid architecture should preserve both advantages through separate budgets, role-specific evidence selection, graph-conditioned vector expansion and scoring metrics that recognize relational support.

7 Empirical Application: the Economic Bulletin

This section describes the operational pipeline used to produce, verify and compare Economic Bulletin chapters under three retrieval settings: Vector RAG, Graph RAG and Hybrid RAG. The objective of the pipeline is not only to generate text, but to make the entire process reproducible and auditable. Each stage produces explicit intermediate artifacts, so that the final chapter can be traced back to its sources, evidence items, planning decisions, verification results and retrieval-quality metrics.

7.1 Chapter configuration

The pipeline starts from chapter-level configuration. Each one of the 11 chapters is associated with two chapter-specific scripts:

```
build_source_manifest.py
```

```
build_retrieval_config.py
```

The first script, `build_source_manifest.py`, defines the source universe for the chapter. It specifies which institutional sources, statistical releases, forecasting pages, reports or database entry points should be used to collect evidence. Each source item contains metadata such as the source identifier, source name, source type, URL, topic, geography, measure, category, paragraph identifier, reliability, claim and parser hints. In the later versions of the pipeline, the source manifest also supports `fallback_urls`, allowing the collector to try alternative official sources when the primary URL is unavailable, blocked or stale.

The second script, `build_retrieval_config.py`, defines the retrieval logic for the chapter. It specifies the analytical categories used by the chapter, the intended paragraph blocks, the keywords and themes to be used during retrieval, and the guidance needed to select and organize evidence. In other words, the source manifest answers the question *which sources are relevant for this chapter?*, while the retrieval config answers the question *which concepts, categories and paragraph-level structures should guide retrieval and evidence selection?*

This separation is important. It keeps the generic pipeline scripts independent of chapter-specific details. The chapter scripts encode domain knowledge, while the generic scripts perform collection, normalization, planning, drafting and verification in the same way across all chapters. This design also makes the benchmark more transparent: if a chapter changes, the relevant source and retrieval assumptions are visible in two explicit configuration scripts rather than hidden inside the retrieval code.

7.2 Evidence collection

This section shows the evidence collection phase which is implemented by the routine `collect_evidence.py`. This script reads the chapter’s source manifest and retrieval config, builds a runtime retrieval profile, and collects raw evidence for the selected retrieval mode. The benchmark evaluates three modes:

1. `vector`
2. `graph`
3. `hybrid`

In vector mode, evidence is collected from the chapter source manifest and associated textual sources. In graph mode, evidence is retrieved from the Neo4j knowledge graph. In hybrid mode, the two retrieval channels are combined: the pipeline collects both vector-style textual evidence and graph-derived relational evidence.

Each run writes a mode-specific collection artifact:

```
<chapter>/<mode>/current_evidence.json
```

The collection stage is deliberately separated from the later stages. At this point, the objective is to gather candidate evidence, not yet to select the best evidence, draft the final narrative or evaluate the output. The raw evidence may therefore contain heterogeneous structures, including web-derived excerpts, graph-derived records, parser metadata, collection statuses and diagnostic fields.

This stage revealed several important practical issues. Some web sources produced metadata-only evidence: the source was reachable and technically collected, but the extracted text did not contain substantive content useful for drafting. Some graph runs produced placeholder evidence, especially before the graph retrieval query was aligned with the actual Neo4j schema. The most important placeholder was: `kg_empty_0001` — Knowledge Graph retrieval returned no edges‘

The collection stage therefore provided both the input data for the benchmark and the first indication of where retrieval or extraction needed to be made more robust.

7.3 Normalization

The third stage is normalization, implemented with the script `normalize_evidence.py`. The purpose of this script is to convert heterogeneous raw evidence into a common schema that downstream scripts can consume reliably. Raw evidence may come from different retrieval modes, different source types and different backends. Without normalization, later scripts would need to handle many special cases. Normalization removes this complexity by producing a canonical evidence representation.

The normalized schema includes fields such as:

- evidence_id
- text
- raw_excerpt
- source_name
- category
- paragraph_id
- evidence_backend
- hybrid_component
- quality_flags

The most important transformation is the assignment of stable evidence identifiers. These identifiers are later used as citation markers in drafts and as anchors for verification. Normalization also standardizes the main evidence text, preserves the raw excerpt where available, records the original source and backend, and marks quality issues through `quality_flags`.

Two robustness fixes were especially important. The first was the metadata-only text fallback. Some collected items did not contain a useful main text field, but did contain a usable raw excerpt, claim, source description or other textual metadata. Rather than discarding such items immediately, the normalizer uses fallback fields to populate ‘text’ when appropriate. This increases the number of usable evidence items while still allowing weak items to be flagged.

The second fix was empty graph placeholder skipping. Items such as empty placeholder should not be passed to the planner as if they were real evidence. If they are selected, they artificially reduce faithfulness because the generated paragraph cannot be checked against substantive supporting material. The normalization stage therefore identifies and skips empty graph placeholders, preventing them from contaminating the evidence pool used by the planner.

The output of this stage is:

```
data/<chapter>/<mode>/normalized_evidence.json
```

This artifact is the canonical input for planning, drafting and evaluation.

7.4 Section planning

The fourth stage is section planning, implemented by `plan_section.py`. This script reads the normalized evidence and assigns selected evidence items to the intended paragraph blocks of the chapter. Its role is to transform a pool of normalized evidence into a structured plan for drafting.

The planner uses the chapter configuration, evidence categories, paragraph identifiers and retrieval guidance to decide which evidence items should support each paragraph block. It creates a formal section plan containing planned evidence IDs, paragraph-level assignments, coverage diagnostics and drafting guidance. The plan therefore acts as an intermediate layer between evidence retrieval and text generation.

This step is important because retrieved evidence is not yet a chapter structure. A raw evidence pool may contain many relevant items, but they must be organized into an analytical sequence. For example, an Economic Bulletin chapter may need to move from headline activity to sectoral developments, then to labour market conditions, risks and projections. The planner makes this structure explicit before drafting begins.

The section plan also provides diagnostics. It can show whether all paragraph blocks have evidence, whether some categories are underrepresented, whether the selected evidence is too concentrated in a small number of sources, and whether the chapter has enough support for a faithful draft. These diagnostics are essential for interpreting later scores. A weak draft may reflect poor generation, but it may also reflect insufficient or poorly distributed evidence at the planning stage.

7.5 Deterministic drafting and verification

The fifth stage is deterministic drafting, implemented by `draft_section.py`. This script creates a first-pass chapter draft using the evidence selected by the planner. The goal is not to produce a final editorially polished Economic Bulletin chapter. Instead, the objective is to create a reproducible, evidence-grounded baseline that can be compared across retrieval modes.

The deterministic draft uses the same basic procedure for vector, graph and hybrid modes. It follows the section plan, uses the planned evidence items, and produces paragraph-level text with explicit evidence markers. These markers link narrative statements back to evidence identifiers. This makes the draft suitable for automatic verification.

The deterministic nature of this stage is important for the benchmark. If the drafting process were highly variable, differences across modes could be driven by generating randomness rather than retrieval quality. By using a controlled drafting procedure, the benchmark tries to ensure that the comparison among vector, graph and hybrid modes reflects differences in retrieved and selected evidence rather than differences in stylistic variation.

The output draft should therefore be understood as a controlled experimental artifact. It is designed to be faithful, traceable and comparable. Editorial fluency can be improved later, but only after the evidence-grounded structure has been produced and verified.

The sixth stage is verification which is carried out with the following two scripts:

`verify_section.py` and `verify_semantic_support.py`

The first script, `verify_section.py`, performs structural and citation-marker verification. It checks whether the draft uses valid evidence markers, whether cited evidence IDs actually exist in the normalized evidence file, whether paragraph blocks are covered, and whether the draft follows the expected section structure. This stage is mainly concerned with formal traceability. If the draft cites a particular evidence, the ensuing item must exist; if a paragraph is supposed to rely on selected evidence, the markers should reflect that.

The second script, `verify_semantic_support.py`, estimates whether each paragraph is substantively supported by the evidence it cites. This is a stricter requirement. A citation marker can be formally valid but semantically weak. For example, a paragraph may cite a real evidence item, but the evidence may not actually support the claim being made. The semantic verifier therefore evaluates paragraph-level support by comparing the generated text with the cited evidence.

Together, these scripts separate two forms of verification. Structural verification asks whether the evidence markers are well formed and traceable. Semantic verification asks whether the cited evidence actually supports the paragraph. Both are necessary. A draft with perfect citation syntax can still be unfaithful, while a substantively plausible draft can still be unusable if it fails to preserve traceable evidence markers.

7.6 Retrieval-quality evaluation

The script `evaluate_retrieval_quality.py` combines the artifacts produced by the previous stages into a scorecard for each chapter and retrieval mode. It uses the normalized evidence, section plan, draft, structural verification results and semantic support results to compute metrics that summarize retrieval and drafting quality.

The evaluation is designed to capture several dimensions of performance. These include faithfulness, traceability, evidence coverage, relevance, paragraph support and alignment between the selected evidence and the generated text. The resulting scorecard allows each mode to be compared within a chapter.

This stage is important because no single metric is sufficient. A system may retrieve many relevant documents but fail to cite them. Another system may produce valid citations but weak semantic support. A third may achieve strong faithfulness but cover only part of the planned chapter structure. The retrieval-quality scorecard therefore combines multiple diagnostics to provide a more balanced assessment of each run.

The output of this stage is a mode-specific evaluation artifact, typically stored alongside the draft and verification outputs. It is later used by the benchmark comparison script.

7.7 Benchmark comparison

The eighth stage is the benchmark comparison which is implemented by the script `build_benchmark_comparison.py`. This script aggregates the retrieval-quality outputs across modes and compares vector, graph and hybrid retrieval for each chapter. Its purpose is to move from individual run diagnostics to a chapter-level benchmark result.

For each chapter, the script compares the three retrieval modes using the overall retrieval-quality score and its underlying components. It identifies the winning mode by overall retrieval quality, while also preserving the sub-metrics needed to interpret the result. This is important because a mode may win overall for different reasons. Vector may win because of strong textual relevance; graph may win because of strong traceability and relational coverage; hybrid may win because it combines enough textual richness with graph-derived structure.

The comparison also helps identify systematic patterns. If vector dominates in most chapters, this suggests that textual richness is especially important under the current scoring function. If graph performs well after schema-aware fallback retrieval, this suggests that relational evidence can support faithful drafting. If hybrid performs well but does not always dominate, this may reveal interactions between evidence-selection caps, vector richness and graph diversity. The benchmark comparison therefore provides both a ranking and a diagnostic explanation of retrieval behavior.

7.8 Open-source language refinement

The ninth stage is an optional language refinement using local open-source language models. This stage is implemented by `refine_chapter_language_ollama.py`. This script uses locally served Ollama models ⁵, such as:

```
gemma4:31b-it-q8_0 or llama3.1:latest
```

The purpose of this stage is not to add new economic content. It is only to improve language quality, fluency and readability. This distinction is essential. The deterministic draft is already evidence-grounded and citation-marked. The refinement stage may make the prose smoother, more coherent and closer to a publishable institutional style, but it is not allowed to introduce unsupported claims, remove evidence markers or change the underlying analytical content.

If the local model drops or changes markers, the script retries with a repair prompt. If the repair attempt still fails, the system falls back to the original block. This ensures that **language refinement** does not break traceability. In other words, the open-source model acts as a **stylistic editor**, not as an independent analyst.

⁵Ollama is an open-source framework designed to download and run LLMs directly on your own computer. It bundles model weights, configurations, and data into a single package, making it easy to run AI models offline without cloud subscriptions or privacy concerns.

This stage is particularly useful because it separates factual grounding from editorial polish. The benchmark can first evaluate retrieval and deterministic drafting under controlled conditions. Only after that can a local language model refine the output while preserving evidence links. This makes the workflow more suitable for institutional use, where fluency matters but unsupported rewriting is unacceptable.

7.9 LaTeX assembly

The final stage is LaTeX assembly, implemented by `build_bulletin_latex.py`. This script assembles selected chapter drafts or refined sections into a complete LaTeX document. The assembled document includes a title page, a table of contents and one section per chapter.

This stage turns the individual chapter artifacts into a coherent bulletin-style output. It also preserves the modularity of the pipeline: each chapter can be generated, verified and refined separately, then combined into a single document. The LaTeX assembly step is therefore not a retrieval or drafting step, but a publication step.

By the time a chapter reaches this stage, it should already have passed through evidence collection, normalization, planning, drafting, verification, evaluation and optional refinement. The LaTeX document is thus the final presentation layer of a much broader reproducible workflow. The value of the pipeline lies precisely in this separation: the final text is not an opaque output of a language model, but the product of a sequence of documented, auditable and reproducible steps.

8 Evaluation framework and comparison metrics

A RAG pipeline cannot be evaluated only by the fluency of the final text. Fluency is necessary, but it is not sufficient. In an institutional setting such as the production of an Economic Bulletin, the generated chapter must also be grounded in retrieved evidence, use valid citations, cover the intended analytical structure, and remain faithful to the sources on which it relies. A fluent paragraph that is unsupported by the retrieved evidence is not a successful output. Similarly, a text with valid evidence markers but weak semantic support is not fully reliable. The evaluation framework must therefore distinguish between retrieval quality, traceability, semantic faithfulness, evidence quality and structural coverage.

This approach follows the broader RAG evaluation literature. RAG systems put together a retrieval component along with a generation component, and the two must be assessed separately as well as jointly. The original RAG formulation emphasized the use of external non-parametric memory to support knowledge-intensive generation and improve factual specificity (see Lewis et

al., 2021). Later RAG evaluation framework literature made explicit the multidimensional nature of the problem. RAGAS, for example, evaluates RAG systems along dimensions such as the relevance of retrieved context, the faithful use of that context, and the quality of the generated answer (see Es et al., 2024 for example). In Saad-Falcon et al., 2024 ARES similarly evaluates RAG systems through context relevance, answer faithfulness and answer relevance. The benchmark developed in this project follows the same principle. The final objective is not merely to determine whether a chapter “sounds good”, but whether the pipeline retrieves useful evidence, selects it appropriately, cites it correctly, uses it faithfully, and covers the planned chapter structure. For this reason, the evaluation is built from several scripts and artifacts. The script `verify_section.py` checks structural and citation-marker validity, `verify_semantic_support.py` estimates whether generated paragraphs are supported by the cited evidence. The script `evaluate_retrieval_quality.py` combines these diagnostics with evidence-quality and coverage measures to produce a retrieval-quality scorecard. The resulting comparison is therefore not an evaluation of style alone, but an assessment of the evidence-grounded drafting process.

8.1 Traceability and Faithfulness

Traceability measures the extent to which generated text can be linked back to valid, planned evidence. In this project, traceability is computed by `verify_section.py`. This script checks whether evidence markers used in the draft correspond to evidence items that exist in the normalized evidence file, whether cited evidence was part of the planned evidence set, and whether citations are distributed across the paragraph blocks in a way that supports the intended chapter structure.

In operational terms, traceability answers three questions. First, are the cited evidence markers valid? A marker is valid only if an evidence item with that identifier exists in `normalized_evidence.json`. Second, are the cited markers planned? A marker may be valid in the evidence file but not selected by the planner for the relevant draft. Third, are citations distributed across the chapter rather than concentrated in only one paragraph or block? A draft that cites evidence only in the first paragraph but leaves the rest of the chapter unsupported would have weak paragraph-level traceability even if all markers are formally valid.

Traceability is therefore a formal property of the draft-evidence connection. It does not, by itself, prove that the cited evidence semantically supports the claim being made. Instead, it checks that the draft preserves the evidence-marker discipline required for later verification. This is essential for establishing an auditable workflow. If generated text cannot be linked to stable evidence IDs, then subsequent faithfulness checks, post-generation audits and benchmark comparisons become unreliable.

A concise definition used in this benchmark is the following: traceability measures whether cited evidence markers are valid, planned and correctly dis-

tributed across paragraph blocks.

Faithfulness measures the extent to which drafted paragraphs are actually supported by the evidence they cite and is computed by `verify_semantic_support.py` which evaluates paragraph-level support by comparing the generated text with the cited evidence items. The purpose is to go beyond formal citation validity. A citation marker may point to a real evidence item, but the paragraph may still make a claim that the evidence does not support. Faithfulness addresses this substantive question.

The semantic verification stage classifies support using four classes:

- supported = 1.0
- partially supported = 0.5
- unsupported = 0.0
- not checkable = 0.0

The main metric is the `strict_rag_faithfulness_score`. It is strict because unsupported and not-checkable paragraphs receive zero credit. This is appropriate for the benchmark because the purpose is to evaluate whether generated text is grounded in retrievable evidence. In an Economic Bulletin setting, not-checkable text is not merely a missing technical score; it is a governance problem. If a generated assessment cannot be checked against its cited evidence, it should not be treated as a reliable institutional draft.

Faithfulness is closely related to the broader RAG evaluation concept of answer faithfulness. In RAGAS and ARES, faithfulness captures whether the generated answer is consistent with the retrieved context.

8.2 Context relevance

Context relevance measures whether retrieved and selected evidence is useful for the chapter and for the paragraph block to which it is assigned. In a RAG system, retrieval can fail in several ways. It may retrieve evidence that is topically unrelated, evidence that is too generic, evidence that is relevant to the chapter but not to the assigned paragraph, or evidence that is formally present but not useful for drafting. Context relevance therefore evaluates the quality of the evidence pool and the selected evidence, not just the final text. In this project, context relevance is approximated through several structural and coverage indicators. The first is the `block_coverage_score`, which measures whether the selected evidence covers the planned paragraph blocks. A chapter with five paragraph blocks should not rely entirely on evidence associated with only one or two of them. The second is the `paragraph_citation_coverage_score`, which checks whether paragraphs in the draft actually contain citations. The third is the `selected_distinct_category_count`, which measures how many evidence categories are represented among selected evidence items. The fourth is the `selected_distinct_paragraph_count`, which measures how many planned paragraph identifiers are represented in the selected evidence.

These metrics are proxies rather than perfect measures. They do not fully determine whether an individual piece of evidence is substantively relevant. However, they provide useful diagnostics. If selected evidence covers only one category or one paragraph block, the retrieval-plan-draft chain is likely too narrow. If paragraph citation coverage is low, the generated text may include unsupported parts. If category and paragraph diversity are high, the evidence base is more likely to support the full chapter structure.

8.3 Context recall and planned context recall

Context recall normally asks whether the retrieval system recovered the information needed to answer the query or complete the task. In a chapter-generation setting, there is no single gold answer and no exhaustive list of all relevant evidence. For this reason, the project uses a practical proxy: `planned_context_recall`.

Planned context recall measures the proportion of planned evidence that actually appears in the draft. The planner selects evidence items for use in the chapter. The draft should then cite a meaningful share of those planned items. If the planner selects a large number of evidence items but the draft cites only a small number of them, then part of the planned evidence base has not been used. This may indicate that the drafting stage ignored relevant evidence, that the selected evidence was redundant, or that the target text was too short to accommodate the planned evidence.

Although the metric is project-specific, it captures an important retrieval-generation link. Retrieval quality is not only about collecting evidence; the evidence must survive the planning and drafting stages. Planned context recall therefore measures whether selected evidence is actually integrated into the generated text. It complements faithfulness: a draft may be faithful to the few evidence items it cites, but still have weak planned context recall if it ignores most of the planned evidence.

8.4 Context precision

Context precision measures whether selected evidence is useful rather than noisy. In a classical retrieval setting, precision asks what proportion of retrieved items are relevant. In this project, the benchmark approximates this concept using evidence-quality and cleanliness indicators, because manual relevance labels are not available for every selected item.

The main proxies are:

- `selected_clean_ratio`
- `selected_non_boilerplate_ratio`
- `selected_url_coverage`
- `selected_quality_flag_counts`

A useful future metric would be the `cited_selected_evidence_ratio`, defined as the proportion of selected evidence that is actually cited in the draft. This would sharpen the link between precision and generation. If many selected items are never cited, they may be redundant, weak, or poorly aligned with the drafting plan. Conversely, if most selected items are cited and semantically support the draft, the selected evidence set is likely more precise.

8.5 Evidence quality and Composite retrieval-quality score

Evidence quality evaluates the intrinsic usability of the evidence items. This is distinct from relevance. An evidence item may be topically relevant but still low quality because it has no substantive text, lacks source traceability, contains mostly boilerplate, or is graph metadata rather than paragraph-level evidence.

The normalization stage assigns quality flags to make such problems explicit. Important flags include:

- `metadata_only_text`
- `missing_url`
- `possible_boilerplate`
- `short_text`
- `graph_metadata_only`

The evidence-quality score combines clean evidence, non-boilerplate evidence and URL/source traceability. This ensures that a retrieval mode is not rewarded simply for returning many items. The evidence must also be usable. This criterion is especially important in a multi-backend benchmark, because vector, graph and hybrid retrieval may produce evidence with different textual properties. Vector evidence often contains richer passages. Graph evidence may contain shorter relational records. Hybrid evidence may contain both. Evidence-quality diagnostics help interpret these differences.

The benchmark combines the previous dimensions into a composite `overall_retrieval_quality_score`. The score is defined as a weighted average of six components:

- $0.20 \cdot \text{block coverage}$
- $0.20 \cdot \text{planned context recall}$
- $0.15 \cdot \text{paragraph citation coverage}$
- $0.20 \cdot \text{traceability score}$
- $0.15 \cdot \text{semantic faithfulness}$

- 0.10· evidence quality score

The weights reflect the benchmark’s priorities. Block coverage and planned context recall each receive substantial weight because a chapter must cover the intended analytical structure and use the evidence selected for drafting. Traceability also receives substantial weight because institutional drafting requires generated statements to be linked to identifiable evidence markers. Semantic faithfulness is essential because valid markers are not enough: the cited evidence must support the generated claims. Paragraph citation coverage ensures that support is distributed across the draft rather than concentrated in isolated sections. Evidence quality receives a smaller but still meaningful weight, because weak, boilerplate or metadata-only evidence can undermine the reliability of the entire pipeline.

This composite score is not intended to be a universal RAG metric. It is a project-specific benchmark score designed for evidence-grounded Economic Bulletin drafting. Its purpose is to compare vector, graph and hybrid retrieval under a common evaluation protocol. The score should therefore be interpreted together with its components. A mode may have high faithfulness but lower context recall, or strong traceability but weaker evidence quality. The composite score summarizes the trade-off, while the submetrics explain it.

8.6 Interpretation of results

The benchmark results showed a clear but nuanced pattern. Vector retrieval generally dominated by overall retrieval quality. Table 1 reports the average benchmark scores across the eleven Economic Bulletin chapters. The results confirm that vector retrieval is the strongest mode by overall retrieval quality, while hybrid retrieval achieves the highest average faithfulness and graph retrieval becomes a viable, fully passing mode after the schema-aware Neo4j fallback. This likely reflects the importance of text-rich evidence for the current

Table 1: Average benchmark scores across the eleven Economic Bulletin chapters. Overall, faithfulness and traceability are expressed on a 0–100 scale; selected evidence reports the average number of evidence items picked by the planner.

Mode	Overall	Faithfulness	Traceability	Selected evidence
Vector	95.36	98.18	97.62	19.64
Hybrid	91.67	100.00	94.32	24.00
Graph	86.68	99.09	91.70	24.00

scoring function. Vector retrieval tends to return longer and more directly usable textual passages. These passages support paragraph drafting, lexical alignment, citation coverage and semantic verification. As a result, vector mode often performed strongly on the composite score.

Hybrid retrieval often achieved very high faithfulness and good traceability, but it did not systematically outperform vector retrieval. This result is not surprising under the current planning policy. Hybrid retrieval combines vector evidence with graph evidence, but the planning stage imposes a fixed evidence-selection cap. When graph evidence enters the selected evidence set, it can displace richer vector evidence. The hybrid evidence pool may therefore be more diverse and more relationally informative, but not always better under a scoring function that rewards text-rich support, paragraph citation coverage and direct semantic alignment.

Graph retrieval became reliable after the schema-aware fallback was introduced. Nevertheless, graph mode usually scored below vector mode. The likely reason is that graph evidence was more structural and less text-rich. It was useful for surfacing relationships among entities, concepts, documents and chunks, but less directly useful for supporting fluent paragraph-level drafting under the current metrics.

These findings suggest that each retrieval mode has a different strength. Vector retrieval is strong as a drafting evidence source. Graph retrieval is strong as a relational evidence source. Hybrid retrieval is promising, but its performance depends heavily on the selection policy and scoring function. A future version of the benchmark should therefore consider retrieval-type aware planning, separate evidence budgets for vector and graph evidence, and graph-specific contribution metrics. Such refinements would allow hybrid retrieval to be evaluated not only by how much it resembles vector retrieval, but also by how well it adds relational structure to the drafting process.

9 Concluding remarks

This paper has presented a reproducible end-to-end RAG pipeline for institutional economic drafting. The objective was not simply to generate fluent text, but to design a controlled workflow in which each stage of the drafting process can be inspected, reproduced and evaluated. The pipeline integrates evidence collection, evidence normalization, section planning, deterministic drafting, structural verification, semantic support evaluation, benchmark comparison, open-source language refinement and LaTeX assembly. Each stage produces explicit artifacts, so that the final chapter can be traced back to the sources, evidence items, planning decisions, citation markers, verification reports and retrieval-quality metrics that generated it.

The main contribution is therefore methodological. The proposed workflow shows how RAG can be adapted to the constraints of institutional economic writing, where outputs must remain grounded in official evidence and consistent with established analytical terminology. By separating source configuration, retrieval, planning, drafting and verification, the pipeline avoids treating the language model as a black box. Instead, the language model is embedded in a broader evidence-management and quality-control architecture.

9.1 Main empirical finding

The benchmark compared three retrieval modes: vector, graph and hybrid. Vector retrieval was the strongest mode in most chapters according to the overall retrieval-quality score. This result reflects the importance of rich textual evidence for paragraph-level drafting, citation support and semantic verification. Vector retrieval generally supplied detailed passages that were directly usable by the deterministic drafting stage and well aligned with the evaluation metrics.

Graph retrieval initially performed poorly in several chapters, not because the graph approach was intrinsically unsuitable, but because of a schema mismatch between the retrieval query and the actual Neo4j database. The original query assumed metadata fields such as chapter identifiers that were not present in the graph. After a schema-aware fallback was introduced, graph retrieval became reliable across the benchmark and all graph runs passed verification. This showed that graph retrieval can provide a meaningful relational evidence layer when the query is aligned with the actual graph schema.

Hybrid retrieval was promising but did not automatically dominate vector retrieval. It often achieved high faithfulness and good traceability, but the simple hybrid selection policy sometimes allowed graph-derived evidence to displace textually richer vector evidence under a fixed evidence cap. As a result, hybrid retrieval was more diverse and relationally informative, but not always superior under the current composite score.

9.2 Practical significance

The practical significance of the pipeline lies in its auditability. The experiment demonstrates that LLM-assisted economic drafting can be organized as a sequence of controlled, inspectable steps rather than as a single opaque generation process. Retrieval, planning, drafting, verification and refinement are separated. This makes it possible to diagnose failures, compare retrieval architectures, preserve evidence links and review the generated text against its sources.

This is especially relevant for institutional publications such as Economic Bulletins. In these settings, the value of AI assistance depends not only on productivity or fluency, but also on traceability, reproducibility and accountability. The proposed pipeline allows economists and technical editors to inspect which sources were used, which evidence items were selected, how they were assigned to paragraph blocks, which claims were supported, and where verification failed or succeeded. In this sense, the workflow provides a practical foundation for using LLMs as controlled drafting assistants rather than autonomous authors.

9.3 Limitations

Several limitations remain. First, the semantic verification stage is still lexical and deterministic. It provides a useful approximation of paragraph-level support, but it cannot fully capture all forms of economic reasoning, implicit continuity or nuanced interpretation. A paragraph may be economically sound while receiving a lower support score because the lexical overlap with the cited evidence is limited. Conversely, a paragraph may share wording with the evidence without fully capturing the underlying analytical meaning.

Second, open-source LLM refinement is constrained but still requires verification. The refinement script is designed to preserve evidence markers, avoid unsupported content and fall back to the original block if repair fails. Nevertheless, any language model used for stylistic improvement can introduce subtle changes in emphasis or interpretation. Refined text should therefore remain subject to marker checks and semantic support verification.

Third, hybrid mode currently uses a simple selection policy. Vector and graph evidence are pooled and then selected under a common evidence cap. This can create displacement effects: graph evidence may increase relational diversity but reduce the amount of text-rich vector evidence available for drafting. A more mature hybrid policy should distinguish the roles of vector and graph evidence more explicitly.

Fourth, graph retrieval remains keyword-based rather than fully paragraph-aware. Although schema-aware fallback made graph retrieval reliable, the graph retriever does not yet exploit the full paragraph structure defined in the chapter retrieval configurations. It retrieves graph evidence using chapter-level themes and keywords, but it does not yet perform fine-grained paragraph-level graph retrieval for each planned block.

9.4 Future work

Future work should address these limitations in several directions. First, graph retrieval should become paragraph-aware by using the `retrieval.config` structure more directly. Instead of retrieving graph evidence only at chapter level, the system should query the graph separately for each paragraph block, using block-specific categories, themes, entities and keywords. This would improve the alignment between graph evidence and the chapter structure.

Second, the hybrid selection policy should be improved. Rather than allowing vector and graph evidence to compete for the same fixed evidence budget, the planner could introduce separate evidence budgets or retrieval-type-aware selection rules. For example, vector evidence could be prioritized for textual drafting support, while graph evidence could be used to enrich planning, entity coverage and relational continuity.

Third, future evaluation could include an LLM-as-judge component under strict evidence constraints. Such a judge should not evaluate style alone. It should be required to compare generated claims against cited evidence and to produce structured support labels. This could complement the current deter-

ministic semantic verifier, especially for cases where support is conceptually valid but lexically indirect.

Fourth, context precision and answer relevance metrics should be strengthened. The current proxies capture useful aspects of evidence quality and coverage, but future metrics should better assess whether selected evidence is genuinely useful for the paragraph it supports. This could include cited-selected evidence ratios, paragraph-level relevance labels, and more precise measures of noisy or redundant evidence.

Fifth, the benchmark should support versioned quarterly reruns. Economic Bulletins are recurrent publications, and the same pipeline should be executable across different quarters with versioned sources, retrieval configurations, outputs and metrics. This would make it possible to assess stability over time and to compare how retrieval performance changes as new economic information enters the corpus.

Finally, the pipeline should be integrated more closely with publication workflows. LaTeX assembly already provides a first publication layer, but future development could connect the system to editorial review, document versioning, issue tracking and internal approval processes. This would move the prototype closer to a controlled institutional production environment, where AI assistance is embedded in a transparent and accountable workflow from evidence collection to final publication.

Acknowledgments

The author thanks Juri Marcucci and Sivia Vori for helping in the preparation of the manuscript. I am responsible for all remaining errors.

References

- Araujo, D., Bruno, G., et al. (2026). Generative artificial intelligence in central banking. *IFC Bulletin n. 67*, 1–20.
- Batthula, V., et al. (2025). Case Study on Understanding the Power of Retrieval Augmented Generation (RAG). *Proceedings of the 29th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2025)*, 57–65.
- Blinder, A., et al. (2008). Central Bank Communication and Monetary Policy: A Survey of Theory and Evidence. *Journal of Economic Literature*, (46), 910–945.
- Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. *ArXiv*, 1–75.
- Brynjolfsson, E., Li, D., & Raymond., L. (2023). Generative AI at work. *National Bureau of Economic Research Working Paper*, (31162).

- Edge, D., et al. (2024). From Local to Global: A GraphRAG Approach to Query-Focused Summarization. *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 150–158.
- Es, D., et al. (2024). RAGAs: Automated Evaluation of Retrieval Augmented Generation. *ArXiv*.
- Furnas, G. W., Landauer, T., L., G., & Dumais, s. T. (1987). The Vocabulary Problem in Human-System Communication. *Communication of the ACM*, 30(11), 964–971.
- Gao, A. K. (2023). Vec2Vec: A Compact Neural Network Approach for Transforming Text Embeddings with High Fidelity. *ArXiv:2306.12689v1*, 1–14.
- Gao, Y. G., et al. (2024). Retrieval-Augmented Generation for Large Language Models: A Survey. *ArXiv:2312.10997*, 1–21.
- Guu, K., et al. (2020). Realm: Retrieval-Augmented Language Models pre-training. *ArXiv*.
- Hogan, A., Blomqvist, E., Cochez, M., d’Amato, C., de Melo, G., Gutierrez, C., Labra Gayo, J. E., Kirrane, S., Neumaier, S., Polleres, A., Navigli, R., Ngonga Ngomo, A.-C., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys*, 54(4), 1–37. <https://doi.org/10.1145/3447772>
- Huang, Y., & Huang, J. X. (2024). A Survey on Retrieval-Augmented Text Generation for Large Language Models. *ArXiv:2404.10981v2*, 1–37.
- Izacard, G., & Edouard, G. (2021). Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. *ArXiv*.
- Ji, Z., et al. (2023). Survey of Hallucination in Natural Language Generation. *ACM computing surveys*, (55), 1–38.
- Ji, Z., et al. (2024). Survey of Hallucination in Natural Language Generation. *ArXiv:2303.17564*.
- Kleppmann, M. (2017). *Designing data-intensive applications* (1st). O’ Reilly Media Inc.
- Leland, C., Karra, A., & Soto, P. (2025). Total Recall? Evaluating the Macroeconomic Knowledge of Large Language Models. *FRB Finance and Economics discussion series*. <https://doi.org/https://doi.org/10.17016/FEDS.2025.044>
- Lewis, P., et al. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *ArXiv*, 1–19.
- Maynez, A., et al. (2020). On Faithfulness and Factuality in Abstractive Summarization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.
- Nagori, A., et al. (2025). Open-Source Agentic Hbrid RAG Framework for Scientific Literature Review. *ArXiv:2508.05660v1*.
- NIST. (2024). *Artificial intelligence risk management framework: Generative artificial intelligence profile* (tech. rep. No. NIST AI 600-1).

- Noy, S., & Whitney, Z. (2023). Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence. *Science*, (381), 187–192.
- Robertson, S. E., & Spärk, J. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3), 129–146.
- Saad-Falcon, J., et al. (2024). ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems. *ArXiv:2311.09476*.
- Sarmah, B., et al. (2024). Hybrid RAG: Integrating Knowledge-Graph and Vector Retrieval Augmented Generation for Efficient Information Extraction. *ArXiv:2408.04948v1*.
- Singh, A., et al. (2025). Agentic Retrieval-Augmented Generation: A survey on Agentic RAG. *ArXiv:2501.09136*.
- Vaswani, A., et al. (2017). Attention is All you Need. *ArXiv*, 1–15.
- Wu, S., et al. (2023). BloombergGPT: A Large Language Model for finance. *ArXiv:2303.17564*.
- Yang, Y., & Mark Christopher, o. (2020). Finbert: A Pretrained Language Model for financial communications. *ArXiv:2006.08097*.
- Yuqi, N. ., Kong, Y., et al. (n.d.). A Survey of Large Language Models for Financial Applications: Progress, prospects and challenges.