# GenAI Firmwide Search Architecture – Technical Summary

Morgan Stanley’s GenAI Firmwide Search Architecture is a sophisticated and scalable retrieval-augmented generation (RAG) system that enables firmwide semantic and keyword search across structured and unstructured enterprise data. The architecture integrates traditional keyword-based search via Solr, semantic search via dense vector embeddings stored in Redis, and OpenAI-based LLM completion to answer user queries. Key features include real-time ingestion, hybrid retrieval using Reciprocal Rank Fusion (RRF), zero-shot classification for query filtering, and prompt engineering strategies to reduce hallucinations. This system enhances knowledge discovery while maintaining strict compliance and content integrity.

## 1. Data Ingestion and Indexing

Lucidworks Fusion Connectors are used to ingest content from various enterprise sources such as Microsoft 365 (Outlook, Teams, OneDrive, SharePoint), Confluence, Stack Overflow, and more. The ingestion pipeline is incremental, with updates occurring hourly or daily depending on the platform. During ingestion, several validation and transformation rules are applied to ensure metadata quality and consistency, including:  
- Removal of documents with fewer than 10 words  
- Enforcing presence of fields like title, body, and lastModifiedDate  
- Auto-generating missing titles and cleaning up HTML content  
- Schema mapping across different data sources  
- Applying governance and transformation logic based on firmwide standards documented at https://jive.ms.com/docs/DOC-1055680

## 2. Embedding and Vector Storage

The system uses OpenAI’s `text-embedding-ada-002` model via Scalar2 API gateway to generate dense vector embeddings at index time and query time. These embeddings are stored in Redis Stack, which uses a Flat index structure to support fast similarity search. Each embedding entry also includes associated metadata such as document titles, ACLs, labels, and governance data.

Redis is deployed in clustered mode with 3 shards and 3 replicas (total 12 nodes, including disaster recovery). Each shard stores a subset of the vector data but shares a unified index schema, ensuring consistent performance and horizontal scalability. Queries are broadcast to all shards without any targeting logic, and results are coordinated internally to return the most relevant matches. This setup is optimized for high-volume, low-latency retrieval.

## 3. Hybrid Retrieval with Reciprocal Rank Fusion (RRF)

Results from Solr and Redis are merged using RRF scoring, which allows hybrid search that balances semantic depth and keyword precision. The RRF score for a document is computed as:  
Score(i) = Σ (1 / (rank(i,n) + 1))  
Where:  
- rank(i,n) = position of document i in result list n  
- n = source (e.g., Solr, Redis)  
The constant ‘+1’ ensures no division by zero and prevents a single source from dominating the final rank. This method emphasizes documents that rank highly across multiple lists, increasing robustness.

## 4. Evaluation and Performance Gains

An evaluation using 100 real-world queries compared Solr-only, Redis-only, and hybrid (RRF-based) strategies across metrics like CTR, nDCG, and Alpha-nDCG. Key results:  
- Hybrid search yielded the highest CTR (93%)  
- Best overall nDCG: 0.95  
- Balanced Alpha-nDCG: 0.60  
- Improved relevance by over 80% and reduced search time by up to 50%  
These results demonstrate hybrid RRF’s superiority in both accuracy and user efficiency.

## 5. Zero-Shot Query Categorization and Guardrails

All queries are passed through a multi-stage LLM-based classifier to determine the query type: Valid Search, HR, Injection, or Abusive. Categories such as injection or HR are blocked from LLM invocation to prevent hallucination or policy violations. The classifier uses a prompt-based template (e.g., genaiguardrail = “”) to return a query type, which is then used to control downstream prompting. Examples include Python queries, Kerberos codes, or misuse attempts. Only valid queries proceed to GenAI for completion.

## 6. Chunking and Query Enhancement

Chunking is customized per source type to retain semantic continuity:  
- Stack Overflow: Each QA pair is treated as one chunk  
- Confluence/Live: Chunks are segmented by sentence or paragraph with overlap  
This minimizes retrieval noise and improves answer grounding.  
Additional query enhancement includes:  
- Keyphrase extraction  
- LLM-based spell correction  
- Temporal normalization (e.g., “last year” → 2024)  
- Phonetic and abbreviation handling  
- Noise filtering before forming Solr queries

## 7. Similarity and Scoring

In Redis, cosine similarity is used to rank embedding proximity. In Solr, traditional metrics such as BM25 and TF-IDF are used along with recency scores (recip(ms(NOW,lastModifiedDate))) for ranking. ACL metadata is applied for filtering results but not ranking.

## 8. Context Injection and Hallucination Management

Top-N documents (typically Top-10) are injected into the prompt sent to the LLM. This number was tuned through manual reviews for completeness and faithfulness. Hallucination risk is managed by:  
- Using deterministic prompts (temperature=0)  
- Setting fixed seeds for reproducibility  
- Limiting LLM output strictly to injected context  
This grounding-first strategy has significantly reduced hallucinations in production deployments.

# Key Effective Challenges

• MRM requested clarification on the Lucidworks Fusion Connectors, particularly regarding the frequency of ingestion and transformation logic applied. Developers confirmed that ingestion is incremental with scheduled updates and outlined the transformation steps (e.g., filtering short documents, HTML cleanup, and schema mapping). MRM noted that governance documentation (Jive DOC-1055680) is applied during ingestion, and no further action was required.

• MRM questioned the embedding model used and latency controls, noting that OpenAI’s text-embedding-ada-002 is invoked via Scalar2 at both index and query time. Developers clarified that no additional batching, caching, or latency-reduction strategies are currently applied. MRM encouraged documenting performance expectations if latency proves material to user experience.

• MRM asked how Redis is scaled and whether any indexing strategies are used for dense vectors. Developers explained that Redis is deployed in clustered mode with 3 shards and 3 replicas (total 12 nodes including disaster recovery), and all shards use a Flat index structure with synchronized schema. MRM requested clarification on query routing logic and sharding coordination, and developers confirmed that queries are broadcast to all shards, with Redis managing response merging.

• MRM asked whether the team conducted any performance comparisons between keyword-only, vector-only, and hybrid RRF (Reciprocal Rank Fusion) retrieval. Developers confirmed a test using 100 real-world queries and reported that RRF outperformed other methods in CTR (93%), nDCG (0.95), and Alpha-nDCG (0.60). Transitioning to hybrid search reduced query time by 50% and improved user relevance by 80%. No concerns were raised, as this validation demonstrated hybrid retrieval’s effectiveness.

• MRM requested details on how zero-shot classification is used for query categorization, and how it influences downstream retrieval or GenAI prompting. Developers explained that an LLM-based classifier groups queries into four categories (Valid, HR, Injection, Abusive) and that blocked categories do not trigger LLM completions. The prompt used for classification is internal (e.g., genaiguardrail = ''). MRM asked whether prompt examples could be shared and whether any prompt-injection filtering logic is applied.

• MRM asked how document chunking is optimized to preserve context, especially for downstream grounding and hybrid retrieval. Developers confirmed that chunking is source-specific: Stack Overflow QA pairs are chunked together, while Confluence/Live data is chunked at paragraph/sentence level with intentional overlap to improve context preservation. MRM agreed with the approach.

• MRM questioned whether semantic-to-Solr query enhancements are used and how these affect performance. Developers described built-in enhancements such as keyphrase extraction, spell correction, phonetic/technical vocabulary fixes, temporal filters, and noise suppression. MRM found the approach reasonable and requested monitoring of query failure or no-hit rates as part of ongoing validation.

• MRM inquired whether any similarity metric other than cosine similarity is used for vector retrieval. Developers clarified that cosine is used in Redis, while traditional metrics like TF-IDF and BM25 are used in Solr, along with recency ranking (recip(ms(NOW,lastModifiedDate))). Metadata (e.g., ACL) is used for filtering but not for scoring.

• MRM asked for justification of the aggregation function used in RRF scoring, specifically why the formula uses a constant '+1' in the denominator. Developers responded that the constant avoids division-by-zero and ensures all ranks contribute meaningfully. It also prevents top-ranked documents from one source from dominating the fused result. MRM accepted the rationale but recommended documenting the mathematical reasoning formally.

• MRM requested details on sensitivity testing for Top-N document selection, which is injected into the LLM for grounding. Developers confirmed testing across N = {5, 10, 15}, with Top-10 providing the best trade-off between quality and performance. This is now the default in production. MRM found the approach robust and reproducible.

• MRM questioned how hallucinations are managed and what temperature settings are used for GenAI completions. Developers confirmed a strict grounding-first approach: injecting only Top-10 results into the prompt, using temperature = 0 for deterministic output, and fixing the seed for reproducibility. MRM found this approach appropriate and aligned with internal hallucination mitigation standards.