Azure Cognitive Search: Outperforming vector search with hybrid retrieval and ranking capabilities

Subscribe

By  Alec Berntson

Published Sep 18 2023 08:47 AM 31.4K Views

A common practice for implementing the retrieval step in retrieval-augmented generation (RAG) applications is to use vector search. This approach finds relevant passages using semantic similarity. We fully support this pattern in Azure Cognitive Search and offer additional capabilities that complement and build on vector search to deliver markedly improved relevance.

In this blog post, we share the results of experiments conducted on Azure Cognitive Search and present a quantitative basis to support the use of **hybrid retrieval + semantic ranking** as the most effective approach for improved relevance out-of–the-box. This is especially true for Generative AI scenarios where applications use the RAG pattern, though these conclusions apply to many general search use cases as well.

1. The Technology Behind Azure Cognitive Search

The query stack in Azure Cognitive Search follows a pattern that's often used in sophisticated search systems, where there are two main layers of execution: **retrieval** and **ranking**.

Retrieval – Often called L1, the goal of this step is to quickly find all the documents from the index that satisfy the search criteria -possibly across millions or billions of documents. These are scored to pick the top few (typically in the order of 50) to return to the user or to feed to the next layer. Azure Cognitive Search supports (3) different L1 modes:

- **Keyword**: Uses traditional full-text search methods – content is broken into terms through language-specific text analysis, inverted indexes are created for fast retrieval, and the BM25 probabilistic model is used for scoring.
- **Vector**: Documents are converted from text to vector representations using an embedding model. Retrieval is performed by generating a query embedding and finding the documents whose vectors are closest to the query's. We used Azure Open AI text-embedding-ada-002 (Ada-002) embeddings and cosine similarity for all our tests in this post.
- **Hybrid**: Performs both keyword and vector retrieval and applies a fusion step to select the best results from each technique. Azure Cognitive Search currently uses Reciprocal Rank Fusion (RRF) to produce a single result set.

Ranking – also called L2, takes a subset of the top L1 results and computes higher quality relevance scores to reorder the result set. The L2 can improve the L1's ranking because it applies more computational power to each result. The L2 ranker can only reorder what the L1 already found – if the L1 missed an ideal document, the L2 can't fix that. L2 ranking is critical for RAG applications to make sure the best results are in the top positions.

- **Semantic ranking** is performed by Azure Cognitive Search's L2 ranker which utilizes multi-lingual, deep learning models adapted from Microsoft Bing. The Semantic ranker can rank the top 50 results from the L1.

2. Hybrid Retrieval + Semantic Ranking yields the best grounding results for Generative AI Applications

Generative AI applications need to be grounded by content retrieved from indexes that contain the knowledge necessary for relevant responses. If irrelevant content is passed to the LLM, it works counter to this objective, and requires the model to filter extraneous information. This can reduce the quality of generated responses and increase latency and operating costs.

Skip to footer content

We performed tests on representative customer indexes as well as popular academic benchmarks to test the quality of content retrieval results. Across the board, the most effective retrieval engine for most scenarios is achieved by:

1. chunking long form content,
2. using hybrid retrieval (combining keyword and vector search), and
3. enabling semantic ranking

Search Configuration	**Customer datasets** *[NDCG@3]*	**Beir** *[NDCG@10]*	**Multilingual Academic (MIRACL)** *[NDCG@10]*
Keyword	40.6	40.6	49.6
Vector (Ada-002)	43.8	45.0	58.3
Hybrid (Keyword + Vector)	48.4	48.4	58.8
Hybrid + Semantic ranker	**60.1**	**50.0**	**72.0**

***Table 1**: Retrieval comparison using Azure Cognitive Search in various retrieval modes on customer and academic benchmarks. See §6.1 How we generated the numbers in this post and §6.2 Search and Dataset configuration for Table 1 for the setup and measurement details.*

3. Hybrid Retrieval brings out the best of Keyword and Vector Search

Keyword and vector retrieval tackle search from different perspectives, which yield complementary capabilities. Vector retrieval semantically matches queries to passages with similar meanings. This is powerful because embeddings are less sensitive to misspellings, synonyms, and phrasing differences and can even work in cross lingual scenarios. Keyword search is useful because it prioritizes matching specific, important words that might be diluted in an embedding.

User search can take many forms. Hybrid retrieval consistently brings out the best from both retrieval methods across query types. With the most effective L1, the L2 ranking step can significantly improve the quality of results in the top positions.

Query type	**Keyword** *[NDCG@3]*	**Vector** *[NDCG@3]*	**Hybrid** *[NDCG@3]*	**Hybrid + Semantic ranker** *[NDCG@3]*
Concept seeking queries	39.0	45.8	46.3	**59.6**
Fact seeking queries	37.8	49.0	49.1	**63.4**
Exact snippet search	51.1	41.5	51.0	**60.8**
Web search-like queries	41.8	46.3	50.0	**58.9**
Keyword queries	**79.2**	11.7	61.0	66.9
Low query/doc term overlap	23.0	36.1	35.9	**49.1**
Queries with misspellings	28.8	39.1	40.6	**54.6**
Long queries	42.7	41.6	48.1	**59.4**
Medium queries	38.1	44.7	46.7	**59.9**
Short queries	53.1	38.8	53.0	**63.9**

Skip to footer content

***Table 2**: NDCG@3 comparison across query types and search configurations. See §6.3 Query Type definitions for Table 2 for a more detailed description of each query type. All vector retrieval modes used the same document chunks (512 token chunks w/25% overlap with Ada-002 embedding model over customer query/document benchmark). Sentence boundaries were preserved in all cases.*

4. Your Document Chunking strategy matters

Chunking solves 3 problems for Generative AI applications:

1. Splitting long documents into limited-length passages allows multiple retrieved documents to be passed to the LLM within its context window limit.
2. Chunking provides a mechanism for the most relevant passages of a given document to be ranked first.
3. Vector search has a per-model limit to how much content can be embedded into each vector.

Embedding each chunk into its own vector keeps the input within the embedding model's token limit and enables the entire document to be searchable in an ANN search index without truncation. Most deep embedding models have a limit of 512 tokens. Ada-002 has a limit of 8,192 tokens. Moderate length documents can have tens of thousands of tokens. The benefit of chunking is particularly strong when the documents are very long or the answers to queries are found later in the document:

Retrieval Configuration	**Single vector per document** *[Recall@50]*	**Chunked documents** *[Recall@50]*
Queries whose answer is in long documents	28.2	**45.7**
Queries whose answer is deep into a document	28.7	**51.4**

***Table 3**: Recall@50 comparison using (1) a single vector to represent each document (first 4096 tokens of each document were vectorized and the rest were truncated) vs (2) chunking each document into 512 token chunks w/25% overlap with Ada-002 embedding model over customer query/document benchmark. Sentence boundaries were preserved in all cases. Metric computed with vector retrieval only (no Semantic ranking).*

Another key consideration is that embedding models must compress all the semantic content of a passage into a limited number of floating-point numbers (e.g. Ada-002 uses 1,536 dimensions). If developers encode a long passage with multiple topics into a single vector, important nuance can get lost. Our analysis shows that using large chunks reduces retrieval performance.

Retrieval Configuration	**Recall@50**
512 input tokens per vector	**42.4**
1024 input tokens per vector	37.5
4096 input tokens per vector	36.4
8191 input tokens per vector	34.9

***Table 4**: Recall@50 comparison of different chunk sizes with Ada-002 embedding model over customer query/document benchmark. Sentence boundaries were preserved in all cases. Metric computed with vector retrieval only (no Semantic ranking).*

There are many ways developers can build the input for each vector. For example, they can overlap each chunk so there is shared context between them, or they can add in the document title or key topics into each vector to give more context. A strategy of terminating vectors at natural sentence and paragraph breaks is both simple and effective.

Skip to footer content

Chunk boundary strategy	Recall@50
512 tokens, break at token boundary	40.9
512 tokens, preserve sentence boundaries	42.4
512 tokens with 10% overlapping chunks	43.1
512 tokens with 25% overlapping chunks	**43.9**

***Table 5**: Recall@50 comparison of different chunk boundary strategies using 512 tokens with Ada002 embedding model over customer query/document benchmark. Metric computed with vector retrieval only (no semantic ranking).*

5. Semantic Ranking Puts the Best Results at the Top

Generative AI scenarios typically use the top 3 to 5 results as their grounding context to prioritize the most important results. AI Search applications work best with a calibrated relevance score that can be used to filter out low quality results.

The semantic ranker runs the query and documents text simultaneously though transformer models that utilize the cross-attention mechanism to produce a ranker score. The query and document chunk score is calibrated to a range that is consistent across all indexes and queries. A score of 0 represents a very irrelevant chunk, and a score of 4 represents an excellent one. In the chart below, Hybrid + Semantic ranking finds the best content for the LLM at each result set size.

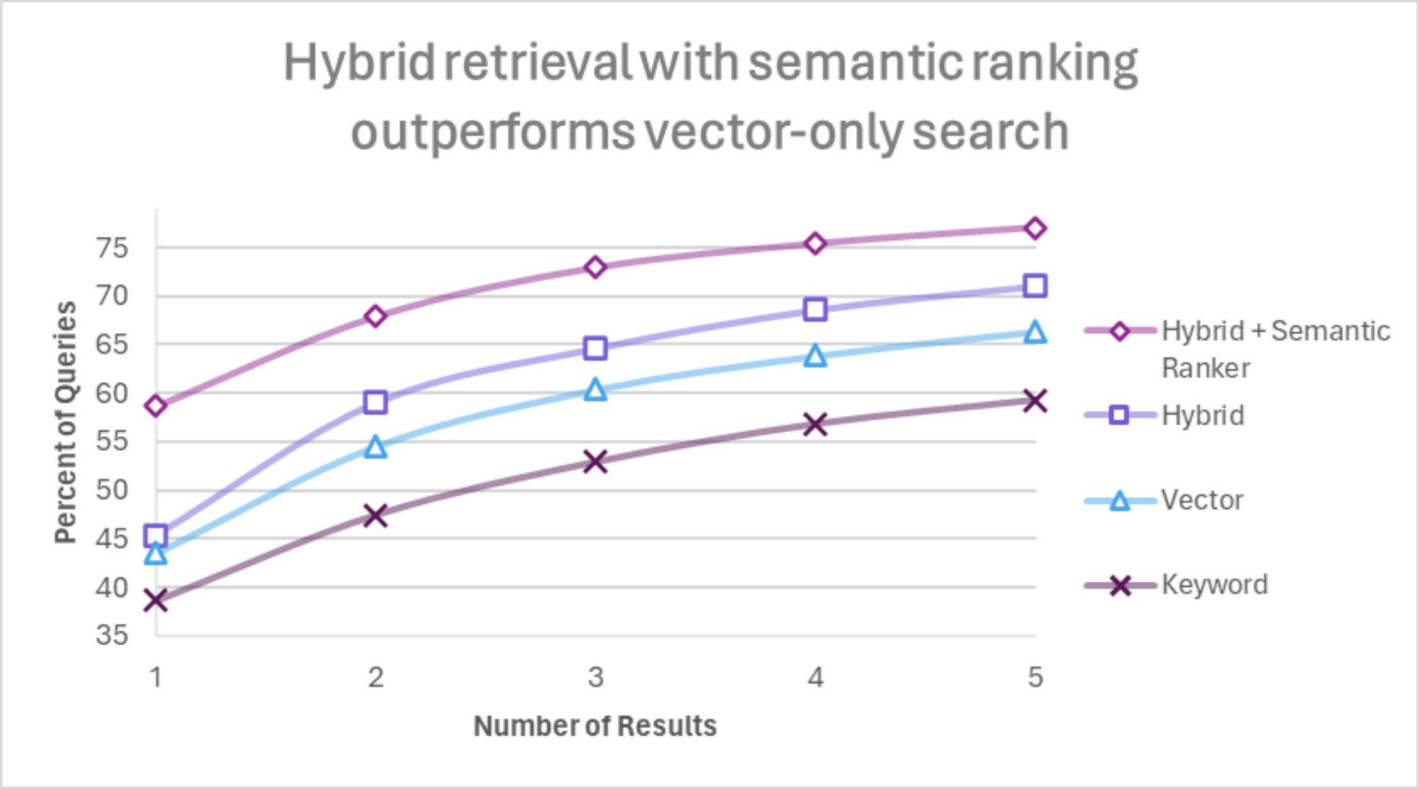

***Chart 1**: Percentage of queries where high-quality chunks are found in the top 1 to 5 results, compared across search configurations. All retrieval modes used the same set of customer query/document benchmark. Document chunks were 512 tokens with 25% overlap. Vector and hybrid retrieval used Ada-002 embeddings.*

In conclusion, the results of the above experiments on real-world and benchmark datasets lead us to recommend the combined strategies of chunked content, hybrid search, and semantic ranking. To test these findings against your users' questions and datasets, please try the

resources linked below to get started:

- Azure Cognitive Search documentation | Microsoft Learn
- Semantic search - Azure Cognitive Search | Microsoft Learn

6. Appendix – How we ran these tests

6.1 How we generated the numbers in this post

To assess which retrieval systems and configurations performed the best, we followed best practices to generate comparable metrics. The high-level process was to replay a list of queries against several document indexes for each configuration and produce scores of how good the retrieval and ranking was.

- **Documents -** We use a consistent set of documents sourced from either Azure customers (with their permission) or publicly available benchmarks.
- **Queries** – We used a combination of end user queries and/or queries generated by several different GPT prompts using random snippets from the document index as grounding.
- **Scoring** – We used benchmark-provided labels (and the official scoring library) for BEIR and other datasets. For customer datasets, we use a GPT prompt that was vetted against a library of high quality (internally reviewed) human ground-truth labels.

We used 3 metrics to determine our recommendations:

- **NDCG@10** – NDCG is a common information retrieval metric that provides a score between 0 and 100 based on how well a retrieval system (1) found the best results and (2) put those results in the ideal order (i.e. a sorted list from the best document to the worst) for all the queries in a given query set. The @10 means that the top 10 documents were considered in the score calculation. We used this metric and the pool of available labels for public benchmarks to be consistent with previous runs. Normalized Discounted Cumulative Gain (NDCG)
- **NDCG@3** – The same NDCG metric but computed on the top 3 documents. We use @3 because we aim to get the most accurate results in the top (3) for generative AI scenarios. We score the top 50 documents because Azure Cognitive Search's semantic ranker works on the top 50 results.
- **Recall@50** – We count the number of documents that our scoring prompt rates as high quality within the top 50 retrieved results and divide it by the number of known good documents for that query.

6.2 Search and Dataset configuration for Table 1

Search Configuration

For this table of results, all documents were broken into 512 token chunks w/25% overlap.

- **Keyword**: The full set of chunks were indexed as if each chunk was a full document. Searches were performed as usual with the keyword-based index (BM25 similarity) and we labeled the top 50.
- **Vector**: All the chunks were embedded using Ada-002 and an ANN index was built. Each query was also embedded with Ada-002 and searched using cosine similarity. The top 50 were labeled.
- **Hybrid**: The keyword index and vector index of the chunks were searched (taking the top 50 from each) and then the results were fused together using RRF. The top 50 documents (chunks) from RRF were labeled.
- **Hybrid + semantic ranking**: Queries were performed against the hybrid search configuration with semantic ranking enabled.

Dataset details

- **Customer datasets** – retrieval benchmarks built from 4 different customer datasets spanned industries and document structures. All documents were imported from raw inputs (e.g. pptx, pdf, html) using Azure Cognitive Search's document ingestion pipeline. Queries for each dataset are a mixture of provided and GPT-generated queries. For ANN vector retrieval tests, all documents were chunked into 512 tokens with 25% overlap and embedded using Ada-002.
- **BEIR** – Imported the following datasets from the official source (beir-cellar/beir: A Heterogeneous Benchmark for Information Retrieval. Easy to use, evaluate your mo...) and used the official labels and official scoring library to generate metrics. For ANN

vector retrieval, each document was indexed using Ada-002 (in 99.9% of cases 1 vector was sufficient to represent the full document). Beir datasets used: nq, scidocs, touche2020, arguana, climate-fever, dbpedia, fever, hotpotqa, covid.

- **Miracl** – Imported directly from the official source (project-miracl/miracl: A large-scale multilingual dataset for Information Retrieval. Thorough human-...) and used official labels for scoring. Due to the size of the dataset the following simplifications were used: we sampled 250 queries per language and only documents with labels were vectorized and a combined multi-lingual index was built. The reported metric is the average across ar, bn, de, en, es, fa, fi, fr, hi, id, ja, ko, ru, sw, te, th, yo, zh languages.

6.3 Query Type definitions for Table 2

Query type	**Explanation**	**Example**
Concept seeking queries	Abstract questions that require multiple sentences to answer	“Why should I use semantic search to rank results?”
Exact snippet search	Longer queries that are exact sub-strings from the original paragraph	“enables you to maximize the quality and value of your LLM investments most efficiently by feeding only relevant information”
Web search-like queries	Shortened queries similar to those commonly entered into a search engine	“Best retrieval concept queries”
Low query/doc term overlap	Queries where the answer uses different words and phrases from the question [which can be challenging for a retrieval engine to find]	“greatest technology for sorting” searching for a document that says: “Azure Cognitive Search has the best models for ranking your content”
Fact seeking queries	Queries with a single, clear answer	“How many documents are semantically ranked”
Keyword queries	Short queries that consist of only the important identifier words.	“semantic ranker”
Queries with misspellings	Queries with typos, transpositions and common misspellings introduced	“Ho w mny documents are samantically r4nked”
Long queries	Queries longer than 20 tokens	“This is a very long query that uses a lot of tokens in its composition and structure because it is verbose”
Medium queries	Between 5 and 20 tokens long	“This is a medium length query”
Short queries	Queries shorter than 5 tokens	“Short query”