Beyond Content Relevance: Evaluating Multi-Instruction Following In Retrieval Models

Arey Pragna Sri
Nannepaga Vanaja
Matcha Jhansi Lakshmi
Bhagyasree Solanki

Abstract

Recent advances in instruction-following capabilities of large language models (LLMs) have enabled them to interpret and act on complex user prompts. However, most retrieval and ranking systems still emphasize content relevance, overlooking the need to understand user-specific multi-attribute constraints. In this work, we extend the concept of instruction-following to a multi-attribute query dataset, where each query is enriched with structured conditions spanning attributes such as source, format, audience, language, and length. Building on the InfoSearch framework, we evaluated various types of existing models. Using refined metrics like Strict Instruction Compliance Ratio (SICR), Weighted Instruction Sensitivity Evaluation (WISE), and Multi-Dimensional Compliance Ratio (MDCR) we systematically assess instruction adherence across original, instructed, and reversely instructed modes. Experimental results demonstrate that GPT-4o-mini consistently outperforms traditional retrieval models across both strict and soft compliance metrics, establishing itself as the most effective reranker in our evaluation. This work contributes a scalable methodology for evaluating multi-attribute instruction-following systems, moving retrieval models closer to genuine user-intent understanding.

CCS Concepts

• **Information systems** → **Retrieval models and ranking**; Search engine architectures and scalability; • **Computing methodologies** → *Machine learning approaches*; • **Information Retrieval Models** → Evaluation metrics.

Keywords

Retrieval models, Dense retrieval, Reranking models, Sparse retrieval, Multi-attribute evaluation, mSICR, mWISE, MDCR, Information retrieval

QuestQ Team, IIT Bhilai

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Arey Pragna Sri, Nannepaga Vanaja, Matcha Jhansi Lakshmi, and Bhagyasree Solanki. 2018. Beyond Content Relevance: Evaluating Multi-Instruction Following In Retrieval Models. In *Proceedings of XX (QuestQ Team).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Large Language Models (LLMs) have shown remarkable progress in following natural language instructions across a wide variety of tasks, ranging from question answering to contextual reasoning. However, while their general-purpose reasoning ability has improved substantially, retrieval and reranking systems that rely on LLMs often fail to capture the multi-attribute nature of real-world user queries. Unlike traditional keyword-based search, real queries frequently embed multiple constraints, for instance, "Find a short English news article for beginners discussing Bitcoin price trends." Such queries combine several attributes including topic, format, length, language, and audience level. Existing instruction-following frameworks primarily evaluate how well a model interprets single-dimensional instructions. However, real-world retrieval contexts demand models that can balance multiple, possibly interdependent, attributes simultaneously. This motivates the need for multi-attribute instruction-following evaluation.

In this work, we extend the instruction-following paradigm beyond single-attribute evaluation by constructing and analyzing a multi-attribute dataset derived from the MS MARCO passage collection. Each query in our dataset is enriched with structured attributes that describe its audience, format, language, source, and content length. We further generate instructed and reversely instructed variants of each query to test models under controlled instruction-following scenarios.

To systematically measure instruction adherence, we employ three metrics namely Mean Strict Instruction Compliance Ratio (mSICR), Mean Weighted Instruction Sensitivity Evaluation (mWISE), and Multi-Dimensional Compliance Ratio (MDCR) designed to quantify how well models respect attribute constraints during retrieval. These metrics capture distinct aspects of compliance: improvement under correct instructions, degradation under reversed instructions, and attribute-level satisfaction.

We benchmark several models across dense retrieval and LLM-based reranking paradigms, including BM25, BGE-Large, E5-Large, Instructor-XL, and GPT-4o-mini.

2 Motivation and Background

Information retrieval has evolved from traditional keyword-based search to context-aware and instruction-following retrieval, driven by the emergence of large language models (LLMs). Modern retrievers are expected not only to match content relevance but also to understand and execute user-specified constraints embedded in natural language instructions. This shift has given rise to a new paradigm, instruction-aware retrieval which measures a model's ability to align document relevance with task-specific guidance, tone, format, or audience attributes contained within the query.

The InfoSearch benchmark introduced a systematic framework for evaluating single-attribute instruction-following retrieval. Each query in InfoSearch is associated with an additional constraint such as "formal tone" or "summary format" and the benchmark quantifies how well retrieval models adapt their ranking behavior in response to that constraint. Through metrics like SICR, WISE, InfoSearch demonstrated that models fine-tuned on instruction datasets (e.g., Instructor or E5) exhibit stronger sensitivity to such attributes than traditional retrievers like BM25.

However, the InfoSearch benchmark evaluates performance under a single attribute condition per query. In realistic information-seeking scenarios, user intents are often multi-dimensional, requiring models to simultaneously respect multiple constraints for example:

"Summarize the causes of inflation briefly, in a formal tone, for school students."

Here, the retriever must jointly satisfy constraints on length, tone, and target audience, rather than focusing on a single one. Current benchmarks do not capture this compositional instruction-following ability, which is crucial for practical applications such as educational content retrieval, legal summarization, and personalized information access.

This motivates the development of a Multi-Attribute Instruction-Following Benchmark, extending InfoSearch to evaluate how retrieval systems perform when faced with compound user constraints. Each query in our dataset is enriched with two or more attributes drawn from predefined dimensions such as tone, audience, length, and format and paired with its instructed and reversed variants. This structure enables a controlled measurement of how model behavior scales with attribute complexity.

By evaluating a diverse set of retrievers including BM25, BGE-Large-v1.5, E5-Large-v2, Instructor-XL, and GPT-4o-mini—on this dataset, we aim to analyze the instruction adherence, sensitivity, and compositional robustness of current retrieval models under multi-attribute query settings. The results provide insights into the extent to which existing retrievers generalize from single-attribute to multi-attribute reasoning, forming a foundation for future advancements in multi-instruction retrieval research.

3 Multi Info Search Data Construction

The Multi-InfoSearch dataset was developed as an extended version of the InfoSearch benchmark, designed to evaluate retrieval models on multi-attribute queries that reflect complex real-world information needs. This section details the dataset construction pipeline, including base data collection, multi-attribute augmentation, generation of instructed and reversed queries, incorporation of hard negatives, and document restructuring. Two key files—final_sorted.jsonl and query-doc.json—serve as the backbone for model evaluation.

3.1 Base Dataset

The foundation of the Multi-InfoSearch benchmark was established using publicly available MS MARCO and BEIR corpora, selected for their high-quality query–passage–relevance annotations and domain diversity. These datasets provide a reliable basis for extending instruction-following evaluation into multi-attribute search contexts.

The MS MARCO Passage dataset (msmarco-passage/train) and selected BEIR subsets (e.g., CQADupStack/Programmers, SciFact) were parsed to extract canonical query–document pairs with explicit relevance labels. Each entry represents a natural language query, a corresponding passage, and its graded relevance judgment.

A typical record in the base dataset follows this JSONL structure:

Example Dataset Entry

```
{
  "dataset": "msmarco-passage/train",
  "query_id": "737282",
  "query": "what is daddy fingers for kids",
  "document": "Finger Family (sometimes called
Daddy Finger) is one of the popular nursery rhymes
about fingers with a family on it that features
Daddy Finger as the thumb/first finger, Mommy
Finger as the second finger, Brother Finger as
the third finger, Sister Finger as the fourth
finger and Baby Finger as the fifth finger.",
  "relevance": 1
}
```

Each record thus defines:

dataset — identifies the source collection (e.g., MS MARCO, BEIR component).

query_id — a unique numeric identifier for the query.

query — the user's original search question or instruction.

document — a short passage or paragraph judged relevant.

relevance — binary label (1 for relevant, 0 for non-relevant).

This structure provides a clean and interpretable foundation upon which multi-attribute extensions were built.

3.2 Multi-Attribute Sampling and Assignment

The Multi-InfoSearch dataset extends the single-attribute setup of the original InfoSearch benchmark by introducing multiple document-level attributes per query. Each attribute represents a semantic dimension that governs how a document should satisfy an instruction beyond surface-level textual relevance.

Attribute Schema

The dataset uses six structured metadata attributes that define diverse contextual properties of each document as shown in Table 1

Each base query–document pair sampled from the BEIR/MS MARCO source pool was augmented with a subset of these attributes to simulate complex, real-world information needs. This design encourages models to reason over both semantic and meta-contextual constraints for instance, retrieving "short English developer-focused guides" rather than simply "guides on SQL."

Attribute Sampling Process

For each base query record, 2–3 active attributes were randomly selected from the six available dimensions. The sampling mechanism was implemented through the following steps:

- **Attribute Pool Formation:** Each record inherited its base metadata (if available) from the source dataset.Missing fields were auto-filled from a pre-defined value pool to maintain diversity.
- **Combination Generation:** The function sample_attr_combinations() was used to produce 3–5 random logical combinations of 2–3 attributes per query: This ensures that a single query could appear in multiple configurations.
- **Attribute Binding:** Each sampled combination was then appended to the base query entry as a dictionary under the key "attributes". This dictionary directly conditioned both query rewriting and document rewriting prompts during generation.

Example

Below is an example showing how a base query from BEIR/cqadupstack/programmers was enriched with attribute information:

Example Attribute Annotation

```
{
  "query_id": "2654",
  "query": "What parts of your coding standard
contribute to quality code?",
  "attributes": {
    "audience": "Developer",
    "format": "Guide",
    "language": "English",
    "length": "Medium",
    "source": "TechBlog",
    "keyword": ["Software Engineering", "Coding
Standards"]
  }
}
```

3.3 Instructed and Reversed Query Variants

To evaluate instruction-sensitivity and directionality, every query was extended into two semantically controlled variants:

- **Instructed Query (instructed_query):** The original query was reformulated to include all assigned attributes explicitly.This represents the intended user instruction that the model should follow.
  Example:
  Original query: "Explain the process of photosynthesis."
  Instructed query: "Explain the process of photosynthesis in a concise, formal paragraph for students."

- **Reversed Query (reversed_query):** This variant intentionally contradicts one or more attributes, creating a counterfactual query form.
  Example:
  Reversed query: "Explain the process of photosynthesis in an informal, detailed report for researchers."

This dual structure allows directional evaluation through metrics such as mSICR, which verifies whether a model's ranking improves for instructed queries and worsens for reversed ones relative to the original query. Such contrastive forms make it possible to measure a model's instruction-following consistency and attribute sensitivity.

3.4 Hard Negatives and Document Rewriting

Unlike earlier IR frameworks that derive hard negatives through statistical or embedding-based retrieval (e.g., BM25 + Dense Retrievers + Cross-Encoder), the Multi-InfoSearch dataset employs a generative hard negative construction process fully driven by GPT-4o-mini. This approach allows the dataset to control attribute-level violations directly during generation rather than relying on external retrieval heuristics.

Hard Negative Generation

Attribute	Description	Example Values
length	Expected textual granularity of the document.	Short, Medium, Long
audience	Target readership or expertise level for the query or document.	Developer, Researcher, Student, General User
source	Provenance or publication platform of the content.	StackOverflow, Wikipedia, TechBlog, Journal, NewsSite
language	Language of the document or query.	English, Spanish, French, etc.
format	Content structure or presentation style.	Forum Discussion, Guide, Research Paper, Blog Post, Manual
keyword	Topical anchor terms or key subject identifiers.	[“Programming”, ‘SQL”], [Machine Learning”], [‘Data Privacy”]

Table 1: Attributes and their corresponding descriptions and example values.

For each base query–document pair, the system prompts GPT-4o-mini to produce two outputs simultaneously:

- A rewritten positive document that fully satisfies all assigned attributes.
- A hard negative document that remains topically similar but violates one or two of those attributes.

This is implemented through the prompt template:

Example

```
{
  "Task 1": "Rewrite the base document so that it fully satisfies all given attributes.",
  "Task 2": "Create one hard negative document that is topically similar but violates one or two attributes.",
  "Return JSON": {
    "positive_doc": "...",
    "hard_negative_doc": "..."
  }
}
```

During generation, GPT-4o-mini is constrained to produce JSON-formatted responses for deterministic parsing. Each hard negative differs subtly from the positive example often preserving lexical overlap but contradicting one or more semantic or stylistic constraints (e.g., tone, audience, or length). This ensures controlled contrastive supervision, where both positive and negative examples share topic semantics yet differ along the attribute dimensions being tested.

Document Rewriting Pipeline

Document rewriting proceeds asynchronously across thousands of queries using a rate-controlled coroutine system. Each rewritten output is appended to 4_rewritten_docs_hard_negatives_with_attri.jsonl. The rewriting stage ensures that:

- All generated texts comply with JSON schema using a safe_json_parse() fallback.
- ach document explicitly encodes attribute satisfaction or violation (tracked via violated_attributes).
- Linguistic consistency and length normalization are preserved by GPT-4o-mini itself, eliminating the need for downstream preprocessing.

Output Example

```
{
  "query_id": "Q204",
  "document": "Explain how blockchain ensures data security.",
  "attributes": {
    "tone": "formal",
    "audience": "professionals",
    "length": "concise"
  },
  "positive_doc": "Blockchain ensures data security by using cryptographic hashing and distributed consensus across nodes.",
  "hard_negative_doc": "Blockchain is great fun to use and keeps your data super safe with lots of computers working together.",
  "violated_attributes": ["tone", "audience"]
}
```

This structured output captures the semantic proximity and attribute violation essential for evaluating instruction-sensitive retrieval performance. Together, the rewritten positives and generative hard negatives form the backbone of the dataset’s contrastive supervision signal used in model evaluation and fine-tuning.

3.5 Final Dataset Schema (final_sorted.jsonl)

After generating the instructed and reversed query variants, along with their rewritten positive and hard negative documents, all processed entries were consolidated into two structured files: final_sorted.jsonl and query-doc.json. These files form the foundational assets for downstream model evaluation within the Multi-InfoSearch benchmark.

A. final_sorted.jsonl — Comprehensive Query–Document Tuples

The final_sorted.jsonl file contains the complete multi-attribute

dataset in a flat JSON Lines format, with each entry representing a single query instance and its corresponding attribute combination, instruction variants, and documents.
Each record encapsulates both the semantic context (instructed and reversed queries) and content-level ground truth (positive and hard-negative documents).
Field Descriptions - Check Table 2

Example Entry


```
{
  "dataset": "beir/cqadupstack/programmers",
  "query_id": "2654",
  "query": "What parts of your coding standard
contribute to quality code?",
  "document": "What should be in a good
(read:useful) coding standard? * Things the code
should have. * Things the code shouldn't have. *
Should the coding standard include definitions of
things the language, compiler, or code formatter
enforces? * What about metrics like cyclomatic
complexity, lines per file, etc?",
  "relevance": 1,
  "audience": "Developer",
  "keyword": ["Software Engineering", "Coding
Standards"],
  "format": "Guide",
  "language": "English",
  "length": "Medium",
  "source": "TechBlog",
  "attributes": {
    "length": "Medium",
    "audience": "Developer",
    "source": "TechBlog"
  },
  "combo_id": 1,
  "instructed_query": "As a developer, what
medium-length sections of your coding standard
outlined in TechBlog contribute to quality
code?",
  "reversed_query": "As a developer, what short
or long sections of your coding standard not
mentioned in TechBlog do not contribute to quality
code?",
  "query_type": "expanded",
  "positive_doc": "A good coding standard should
encompass several key components to ensure code
quality and maintainability. Firstly, it should
specify essential elements...",
  "hard_negative_doc": "When creating a coding
standard, it's important to consider various
aspects. It should list things ...",
  "violated_attributes": ["length"]
}
```


Each JSONL entry thus represents a self-contained query–document–attribute tuple, ready for evaluation across multiple instruction variants.

B. query-doc.json — Evaluation Mapping

While final_sorted.jsonl stores the complete dataset, the query-doc.json file provides a structured mapping between query IDs and their corresponding documents. This file enables lightweight, model-agnostic evaluation during retrieval, reranking, and metric computation stages.

Each record associates a query with one or more candidate documents, each annotated by type (positive or hard_negative).

File Structure


```
{
  "query_id": "2654",
  "query": "What parts of your coding standard
contribute to quality code?",
  "documents": [
    {
      "doc_id": "doc_1",
      "type": "positive",
      "text": "A good coding standard should
encompass several key components to ensure code
quality and maintainability..."
    },
    {
      "doc_id": "doc_2",
      "type": "hard_negative",
      "text": "When creating a coding standard,
it's important to consider various aspects. It
should list things ..."
    },
    {
      "doc_id": "doc_3",
      "type": "positive",
      "text": "A comprehensive coding standard is
essential for maintaining high-quality code. It
should ..."
    }
  ]
}
```


Each documents list ensures that retrieval evaluation models (e.g., BM25, dense retrievers, or LLM-based rerankers) can uniformly operate by comparing how well each candidate aligns with the query intent and attributes.

C. Data Flow in Evaluation

- **final_sorted.jsonl** → provides full query, attributes, and rewritten text variants for analysis and fine-tuning.
- **query-doc.json** → serves as the minimal structured input for all retrieval and ranking pipelines, from traditional models like BM25 to advanced LLM-based evaluators such as GPT-4o-mini.

This design supports modular data access, low memory footprint during embedding computations, and easy adaptation to different evaluation frameworks.

3.6 Dataset Summary

The final Multi-InfoSearch Dataset consists of:

Field	Description
`dataset`	Original dataset name (e.g., *MS MARCO*, *BEIR*) from which the base query–document pair was sampled.
`query_id`	Unique identifier for the query instance.
`query`	Base unmodified query text.
`document`	Original reference passage or snippet retrieved from the source dataset.
`relevance`	Binary or graded relevance label (1 for relevant, 0 for non-relevant).
`audience`, `format`, `language`, `length`, `source`	Metadata attributes associated with the query context.
`attributes`	Dictionary of active attributes for this variant, used to condition the instruction.
`combo_id`	ID representing the attribute combination assigned to this specific variant.
`instructed_query`	Natural-language reformulation of the base query incorporating all attributes.
`reversed_query`	Attribute-negating version used to test instruction consistency.
`query_type`	Denotes the generation stage (e.g., `"expanded"`).
`positive_doc`	Rewritten version of the original document that *satisfies all given attributes.*
`hard_negative_doc`	Semantically similar but attribute-violating document variant.
`violated_attributes`	List of attributes intentionally contradicted in the hard negative.

Table 2: Description of dataset fields and their roles.

Component	Count
Total queries	9,596
Documents	9,576
Query variants	28,000+ (original, instructed, reversed)
Avg. attributes/query	3
Docs/query	3–4 (positive, hard negative, negative)

Table 3: Overview of dataset composition and statistics.

Together, these design choices enable comprehensive evaluation of both semantic retrieval capability and instruction-following robustness, bridging the gap between traditional IR and instruction-grounded retrieval.

4 Evaluation Metrics

Traditional information retrieval benchmarks primarily assess lexical relevance that is, how well a retrieved document matches a given query in surface form. However, in multi-attribute search scenarios, queries are instructional and contextual in nature: they specify constraints such as audience type, language, or content length. Consequently, a suitable evaluation framework must capture whether the retriever:

- Understands attribute-conditioned instructions.
- Distinguishes negated or reversed instructions.
- Ensures semantic alignment between the retrieved content and all attribute dimensions.

The following metrics are carefully designed to quantify different aspects of instruction following and attribute awareness in multi-constraint retrieval.

4.1 mSICR: Multi-Attribute Semantic Instruction Compliance Rate

The mSICR metric measures whether the model correctly interprets attribute-based instructions in query variants. It compares the ranking of the positive document across original, instructed, and reversed queries: The SICR metric is generalized from a single condition to multiple conditions. For a query with conditions $\{c_1, c_2, \ldots, c_m\}$, each condition has its gold document.

- *Instructed mode:* The gold document must satisfy all conditions simultaneously.
- *Reverse mode:* The gold document must be demoted if any of the negated conditions apply.

Success is defined as the model ranking the multi-condition gold document higher with instruction and lower with reversed instruction:

$$I(q) = \begin{cases} 1 & \text{if } (R_{\text{ins}} < R_{\text{ori}} \wedge S_{\text{ins}} > S_{\text{ori}}) \wedge (R_{\text{rev}} > R_{\text{ori}} \wedge S_{\text{rev}} < S_{\text{ori}}) \\ 0 & \text{otherwise.} \end{cases}$$

The final metric is the average across all queries:

$$\text{mSICR} = \frac{1}{J} \sum_{j=1}^{J} I(q_j).$$

This extends the strict binary test to multi-condition queries.

4.2 mWISE: Multi-attribute Weighted Instruction Sensitivity Estimate

The mWISE metric generalizes WISE's reward–penalty scheme to multi-attribute instructions, where a query may specify multiple conditions that must be satisfied by the retrieved document. Let m denote the total number of attribute conditions for a query.

$$\Delta_{\text{ori}\to\text{ins}} = R_{\text{ori}} - R_{\text{ins}}, \quad \Delta_{\text{ori}\to\text{rev}} = R_{\text{rev}} - R_{\text{ori}}$$

Case 1: Reward (Instruction Followed). When the instructed retrieval improves upon the original ($R_{\text{ins}} \le R_{\text{ori}} < R_{\text{rev}}$), a reward proportional to the improvement and the number of satisfied attributes is applied:

$$f_{\text{reward}}(q) = \begin{cases} 1, & R_{\text{ori}} \le N \;\&\; R_{\text{ins}} = 1, \\ \dfrac{\#\text{conditions satisfied}}{m}\left(1 - \sqrt{\dfrac{R_{\text{ori}} - R_{\text{ins}}}{K}}\right)\dfrac{1}{\sqrt{R_{\text{ins}}}}, & R_{\text{ori}} \le K, \\ 0.01 \cdot \dfrac{\#\text{conditions satisfied}}{m}, & \text{otherwise.} \end{cases}$$

Case 2: Penalty (Instruction Not Followed). When the instruction is ignored or reversed, a penalty proportional to the number of violated attributes is imposed:

$$f_{\text{penalty}}(q) = \frac{\#\text{conditions violated}}{m} \times \begin{cases} -1, & R_{\text{rev}} < R_{\text{ori}} < R_{\text{ins}}, \\ \dfrac{R_{\text{ori}} - R_{\text{ins}}}{R_{\text{ins}}}, & R_{\text{ori}} \le R_{\text{ins}}, \\ \dfrac{R_{\text{rev}} - R_{\text{ori}}}{R_{\text{ori}}}, & R_{\text{rev}} \le R_{\text{ori}}. \end{cases}$$

Final metric:

$$F(q) = \begin{cases} f_{\text{reward}}(q), & R_{\text{ins}} \le R_{\text{ori}} < R_{\text{rev}}, \\ f_{\text{penalty}}(q), & \text{otherwise.} \end{cases}$$

$$\text{mWISE} = \frac{1}{J}\sum_{j=1}^{J} F(q_j)$$

Here, $R_{\text{ori}}, R_{\text{ins}}, R_{\text{rev}}$ denote the ranks of the correct (positive) document under the original, instructed, and reversed queries respectively; K controls the top-K rank sensitivity (typically $K = 10$), and N indicates the number of top documents considered "ideal" (commonly $N = 1$).

4.3 MDCR: Multi-Dimensional Consistency Recall

(1) Notation:

$$Q = \{q_1, q_2, \ldots, q_J\} \quad \text{(set of queries)}$$

Each query q requires a subset of dimensions $D_q \subseteq D$ (e.g., Audience, Format, Language).

$$m_q = |D_q| \quad \text{(number of required dimensions)}$$

$$R_q(K) = [r_{q,1}, r_{q,2}, \ldots, r_{q,K}] \quad \text{(top-}K\text{ retrieved documents)}$$

Satisfaction indicator:

$$s_{q,i,d} = \begin{cases} 1 & \text{if document } r_{q,i} \text{ satisfies dimension } d, \\ 0 & \text{otherwise.} \end{cases}$$

(2) Per-query compliance (strict, single-document): A query is successful if at least one document in the top-K satisfies *all* requested dimensions simultaneously:

$$\text{MDCR}_{\text{strict}}(q) = \max_{1 \le i \le K}\left(\prod_{d \in D_q} s_{q,i,d}\right).$$

If there exists a document $r_{q,i}$ such that $s_{q,i,d} = 1$ for all $d \in D_q$, then $\text{MDCR}_{\text{strict}}(q) = 1$. Otherwise, $\text{MDCR}_{\text{strict}}(q) = 0$.

(3) Per-query compliance (soft, single-document): Sometimes no document matches all dimensions, but some may partially satisfy them. We define a soft score using the best document (highest fraction of satisfied dimensions):

$$\text{MDCR}_{\text{soft}}(q) = \max_{1 \le i \le K}\left(\frac{1}{m_q}\sum_{d \in D_q} s_{q,i,d}\right).$$

Range: $[0, 1]$.

- $= 1$ if some document meets all dimensions.
- $= 0.67$ if the best document satisfies 2 out of 3 dimensions, etc.

This captures partial compliance.

(4) Dataset-level aggregation: Across all queries Q:

$$\text{MDCR}_{\text{strict}} = \frac{1}{J}\sum_{j=1}^{J} \text{MDCR}_{\text{strict}}(q_j),$$

$$\text{MDCR}_{\text{soft}} = \frac{1}{J}\sum_{j=1}^{J} \text{MDCR}_{\text{soft}}(q_j).$$

4.4 Interpretation of Metrics(Refer Table 4)

5 Models Evaluation

5.1 Classification of Retrieval Models and Their Working Principles

In our project, we evaluate three broad families of retrieval models:

5.1.1 ***Sparse Retrieval: BM25.*** BM25 is a classic bag-of-words retrieval model. It represents each document and query as a sparse vector over terms and ranks documents based on:

- How often the query terms appear in the document (term frequency),
- How rare those terms are across the whole corpus (inverse document frequency),
- Document length normalization (to avoid always favoring very long documents).

Intuitively, BM25 works on lexical overlap: if a document shares many important words with the query, it is likely to be ranked higher. It doesn't know anything about meaning beyond exact or near-exact word matches.

5.1.2 ***Dense Retrieval Models (Embedding Models).*** // Dense retrieval models encode queries and documents into continuous vectors (embeddings) in a high-dimensional space. Semantic similarity is then computed using cosine similarity between vectors. This allows them to capture meaning rather than just shared words.

We evaluate several dense models:

- **BGE-Large-en-v1.5**
  A powerful English embedding model optimized for general-purpose retrieval. It maps semantically similar sentences/documents close together, even when they don't share words.

Table 4: Evaluation Metrics and Their Interpretations

Metric	Evaluates	Value Range	Ideal Model Behavior
mSICR	Instruction directionality (benefit vs. reversal)	{0, 1}	1 → Improves with correct instruction, worsens with reversal
mWISE	Magnitude of instruction sensitivity	$(-\infty, +\infty)$	Positive values → model gains with instruction
MDCR_soft	Semantic attribute alignment	[0, 1]	High → consistent alignment with attributes
MDCR_strict	Complete attribute satisfaction	{0, 1}	1 → all attributes semantically satisfied

- **E5-Large-v2**
  A strong general embedding model designed with instructions in mind (e.g., "retrieve documents that…"). It is good at understanding query intent and aligning that with document content.
- **Instructor-XL (hkunlp/instructor-large)**
  This model conditions embeddings on a "task instruction" (e.g., "represent the document for retrieval"). It aims to produce embeddings that are more tailored to specific tasks like search and classification.
- **GTE-Qwen2**
  A family of multilingual / English-focused text embedding models capable of handling varied domains. It emphasizes efficiency with competitive retrieval performance.
- **E5 Mistral-Instruct (intfloat/e5-mistral-7b-instruct)**
  A larger instruction-tuned model that uses Mistral as the backbone. It is designed to understand complex queries and align them with relevant documents in an instruction-following setting.
- **GritLM**
  A retrieval-oriented language model designed to produce robust embeddings under instruction-heavy or noisy environments. It emphasizes robustness and generalization.
- **SFR-Embedding-2-R**
  A specialized sentence embedding model tuned for retrieval and ranking tasks, focusing on high-quality representations for both queries and documents.
- **NV-Embed-v2**
  An embedding model from NVIDIA optimized for high-performance semantic search and retrieval, often tuned for large-scale use cases.

Across all these dense models, the working principle is the same:

> *Represent text as vectors in a semantic space and retrieve documents that are closest to the query vector.*

5.1.3 ***Reranking Models***. Reranking models work in a second stage. Instead of scanning the entire corpus, they take a short list of candidate documents (e.g., top-k from BM25 or an embedding model) and re-score each query–document pair using a more expressive model.

We consider two broad reranking paradigms:

- **Point-wise Rerankers**
  These treat reranking as a "score this (query, document) pair" problem, one pair at a time.
  – **Mistral-ins-v0.2**
    An instruction-tuned Mistral model used to score how relevant a document is to a given query. It can capture subtle semantics and follow instructions embedded in the query.
  – **Llama-3.1**
    A modern LLM backbone used as a point-wise reranker by feeding query and document together and asking the model to output a relevance score.
  – **FollowIR**
    A point-wise reranker specifically tuned for instruction-following retrieval. It focuses on whether the retrieved document follows the instruction-like constraints embedded in the query.
  These models are more expensive than embeddings but give higher-quality judgments at the top of the ranking.
- **List-wise Rerankers**
  List-wise rerankers consider multiple documents at once when deciding their final ordering. Instead of scoring each document independently, they try to optimize the ranking of the whole list.
  – **Zephyr-beta (HuggingFaceH4/zephyr-7b-beta)**
    A chat / instruction model adapted as a list-wise reranker. It can consider relationships between candidate documents and better promote the most compliant ones.
  – **RankVicuna-v1**
    Based on the idea of using an LLM (Vicuna-style) to improve ranking quality, often by comparing documents and reasoning about which is better for the query.
  – **RankZephyr-v1**
    A Zephyr-based model fine-tuned specifically for ranking tasks, aiming to combine instruction-following capabilities with ranking sensitivity.
  Conceptually, list-wise models can compare candidates against each other, which is especially important in

multi-attribute and instruction-heavy scenarios like ours.

5.2 Evaluation Framework: Embedding-Based Retrieval and Reranking Retrieval

In this section, we describe what we concretely do, step by step, to evaluate all these models (BM25, dense embedding models, and rerankers) on our multi-attribute dataset. No code, just the actual workflow.

5.2.1 *BM25(Sparse)-Based Retrieval*.

- **Global corpus building**
  We flatten all documents from all queries into a single corpus list:
  - Each entry has:
    1. Document text
    2. A unique ID combining query_id and document's own doc_id
  - We also keep a mapping:
    For each query_id, a dictionary from cleaned document text → document ID. This helps us quickly find the positive document ID for that query.
- **BM25 index construction (for sparse baseline)**
  For the BM25 baseline:
  1. We tokenize all documents.
  2. We build a BM25 index over the entire corpus.
  This allows us to run BM25 retrieval directly over all documents for any given query.

5.2.2 *Embedding-Based Retrieval (Dense Models)*.

For each embedding model (BGE, E5, Instructor, GTE-Qwen2, E5-Mistral, GritLM, SFR-Embedding-2-R, NV-Embed-v2), we repeat the same evaluation pipeline:

- **Document embedding**
  1. We pass every document in the corpus through the model to obtain its embedding.
  2.These embeddings are stored in memory (or on disk if needed) and reused for all queries for that model.
- **Query embedding and retrieval**
  For each query instance (per row in final_sorted.jsonl):
  - We take three query variants:
    - Original query (query)
    - Instructed query (instructed_query, with explicit conditions)
    - Reversed query (reversed_query, with negated conditions)
  - For each variant:
    - Encode the query into an embedding using the same model.
    - Compute similarity between this query embedding and all document embeddings (cosine similarity).
    - Sort documents by similarity and take the top-k documents (e.g., top-10).This is the retrieved ranking for that query variant:
      1. R_ori for original query.
      2. R_ins for instructed query.
      3. R_rev for reversed query.
- **Finding the rank of the positive document**
  - We look for the gold document ID (positive document) in each of the three ranked lists.
  - We record:
    - Rank under original query: Rori_rank
    - Rank under instructed query: Rins_rank
    - Rank under reversed query: Rrev_rank
  - We also keep the corresponding similarity scores at those positions: S_ori, S_ins, S_rev

  This tells us:
  1. Does the gold document move up when instructions are added?
  2. Does it move down when instructions are reversed?

5.2.3 *Reranking Evaluation (Point-wise and List-wise)*.

For reranking models (Mistral-ins-v0.2, Llama-3.1, FollowIR, Zephyr-beta, RankVicuna-v1, RankZephyr-v1), we follow a similar overall logic but with a different first step.

- **Candidate generation**
  - Instead of searching the whole corpus with the reranker, we first obtain a candidate set using an embedding model or BM25 (top-k per query).
- **Reranking with cross-encoders or LLMs**
  - For each query variant (original, instructed, reversed):
    - We take the query and each candidate document and feed them together into the reranking model.
    - The model outputs a relevance score for each query–document pair.
    - We sort candidates by this score to form the new ranking R_ori,R_ins,R_rev
- **Metrics over reranked lists**
  - Once we have these three reranked lists, we compute the same three metrics:
    - mSICR: Does instructed move the gold doc up and reversed push it down?
    - mWISE: How strong is the reward/penalty signal based on rank changes?
    - MDCR: Over the top-K reranked documents, how well do they satisfy all attributes?

6 Performance Benchmarking

6.1 Cross Model Comparisons

6.1.1 *Sparse Retrieval Model (BM25)*.

- **Performance:**
  - mSICR: 0.0198
  - mWISE: 0.0314
  - MDCR_soft: 0.2892
  - MDCR_strict: 0.0

- **Interpretation:** BM25, a lexical matching model, relies purely on keyword frequency and term weighting without understanding semantic relationships. This explains its low mSICR and mWISE, as it fails to adapt to instruction changes or semantic nuances.The MDCR_strict metric is nearly zero because attribute-driven document selection requires contextual understanding that BM25 lacks.

6.1.2 *Dense Retrieval Models (Metric Results - Table 5)*.

Observations

- Dense models generally outperform BM25, demonstrating better semantic generalization.
- E5-Mistral-7B-Instruct achieves the highest scores among dense models:
  - Its instruction-tuned nature enables better response to query variations and attribute sensitivity.
- Instructor-Large also performs well due to its multi-task and instruction-based training.
- GTE models maintain stable MDCR_soft values (0.53–0.54) but have low mSICR, showing strong semantic consistency but weaker sensitivity to query direction changes.
- BGE-Large has a slightly lower mSICR, possibly because it prioritizes general semantic matching over instruction-based alignment.

Dense retrieval models balance semantic accuracy and instructional adaptability, outperforming BM25 but still limited compared to rerankers that directly model pairwise relevance.

6.1.3 *Reranking Models(Metric Results (refer Table 6))*.

Observations

- Cross-Encoders like MiniLM and Zephyr outperform all dense retrieval models significantly.
  - This is because they jointly encode query–document pairs, allowing deeper interaction modeling.
- Zephyr-7B-Beta (list-wise tuned) exhibits strong instructional sensitivity and semantic consistency, reflected in higher mWISE and MDCR.
- GPT-4o-Mini, though evaluated on fewer queries, achieves the best overall performance:
  - Its high mSICR (0.0758) and MDCR_soft (0.8178) show strong multi-attribute compliance and semantic understanding.
  - The higher mWISE indicates exceptional ability to reward correctly ranked documents under instruction changes.

  Reranking models, especially large instruction-tuned ones like Zephyr and GPT-4o-Mini, excel in instruction-following and attribute reasoning, reflecting richer contextual understanding and precise relevance ranking.

6.2 Comparison across Attribute Complexity

When comparing single-attribute and multi-attribute query evaluations, the metrics exhibit a clear trend in how model sensitivity and compliance behave as attribute complexity increases.

1. General Trend

Across all model categories sparse, dense, and reranking both mSICR and mWISE values decrease slightly in the multi-attribute setting compared to their single-attribute counterparts. This reduction reflects the greater challenge of aligning retrieval models with multiple, sometimes interdependent, semantic conditions simultaneously.

2. Sparse Model (BM25)

In the single-attribute evaluation (from the paper), BM25 shows moderate SICR and WISE, as lexical matching can handle isolated attribute cues. However, in multi-attribute evaluation, mSICR 0.02 and mWISE 0.03, demonstrating a sharp decline. Reason: BM25's purely term-based scoring fails to capture joint attribute semantics; it cannot generalize when multiple attribute descriptors are expressed in varying lexical forms.

3. Dense Models

Dense embedding models such as BGE-Large, E5-Large, and Instructor-XL maintain a moderate drop in mSICR (0.01–0.02) and mWISE (0.05–0.07) relative to single-attribute values. Reason: These models encode semantic similarity and can handle paraphrasing, but when multiple attributes are introduced, the embedding space becomes less discriminative, averaging across multiple semantic signals leads to diluted sensitivity (hence lower mSICR) but stable alignment (consistent mWISE).

4. Reranking Models

Cross-encoders like Mistral-ins-v0.2, Zephyr-7B-beta, and GPT-4o-mini show the smallest performance degradation and even achieve the highest absolute mSICR and mWISE values (up to 0.24 for mWISE).The main reason is that these models explicitly process query–document pairs jointly, enabling fine-grained contextual reasoning about attribute satisfaction. Even when multiple attributes are present, they can weigh attribute relevance at the sentence level, preserving ranking sensitivity and instruction compliance.

Summary

- Single-Attribute → Multi-Attribute transition: introduces semantic interference, each attribute adds a new constraint, increasing retrieval difficulty
- SICR decrease: due to lower sensitivity to rank improvements for individual attribute cues.
- WISE decrease: due to the compounded effect of satisfying multiple instructions simultaneously.

7 Conclusion

This study systematically evaluated sparse, dense, and reranking retrieval systems under instruction-sensitivity and multi-attribute constraint settings two aspects that modern retrieval systems increasingly need to support. Our results reveal a consistent performance hierarchy: sparse BM25 < dense embedding models < reranking models < LLM-based rerankers. Sparse lexical retrieval suffers due to its inability to capture semantic and attribute-level constraints, yielding very low

Model	mSICR	mWISE	MDCR_soft	MDCR_strict
BGE-Large-v1.5	0.0086	0.0545	0.5313	0.0102
E5-Large-v2	0.0154	0.0538	0.5346	0.0107
Instructor-Large	0.0214	0.0605	0.5272	0.0196
GTE-Small	0.0081	0.0489	0.5308	0.0218
GTE-Base-en-v1.5	0.0151	0.0521	0.5408	0.0224
E5-Mistral-7B-Instruct	0.0196	0.0713	0.5524	0.0237

Table 5: Evaluation metrics for different dense retrieval models.

Model	mSICR	mWISE	MDCR_soft	MDCR_strict
FollowIR (MiniLM)	0.0359	0.1057	0.6384	0.0380
Zephyr-7B-Beta	0.0597	0.1899	0.6771	0.0419
GPT-4o-Mini	0.0758	0.2387	0.8178	0.0597

Table 6: Evaluation metrics for different reranking models.

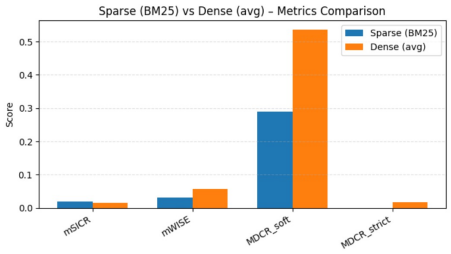

Figure 1: Comparison between sparse and dense retrieval approaches.

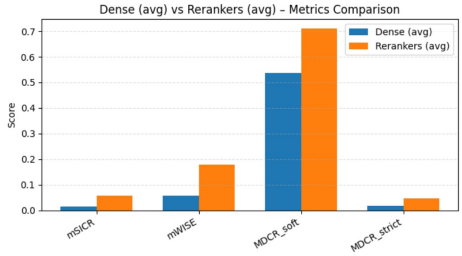

Figure 2: Comparison between dense and reranking retrieval approaches.

mSICR and zero strict compliance. Dense embedding models show clear improvements due to their ability to encode global semantics, but they still struggle with fine-grained instruction following, reflecting in modest mSICR values. Instruction-tuned dense models (e.g., E5-Mistral-7B-Instruct, Instructor-Large) perform notably better, confirming that instruction-aware training directly improves compliance.

Reranking models, especially GPT 4o mini demonstrate the strongest alignment with instructions and complex attribute constraints. Their substantially higher mSICR, mWISE, and MDCR scores confirm that pairwise or listwise relevance modeling is fundamentally more suited to multi-attribute retrieval tasks. GPT-4o-mini, with its large context understanding and reasoning abilities, achieves the strongest performance overall, showing that LLM-based reranking is currently the most effective strategy for structured, attribute-guided retrieval. However, this improvement comes with computational cost LLM reranking is heavier and less scalable than dense retrieval. Overall, the findings highlight a clear trade-off between efficiency and attribute-controlled retrieval accuracy, suggesting a hybrid pipeline (dense retrieval → LLM reranker) as the most practical solution for real-world multi-attribute search systems.

8 References

- Jianqun Zhou, Yuanlei Zheng, Wei Chen, Qianqian Zheng, Hui Su, Wei Zhang, Rui Meng, and Xiaoyu Shen. *Beyond Content Relevance: Evaluating Instruction Following in Retrieval Models.* arXiv preprint arXiv:2410.23841, 2024. Available at: https://arxiv.org/abs/2410.23841
- Swayambhu Nandi, Debjit Paul, and Tanmoy Chakraborty. *Towards Better Instruction Following Retrieval Models.* 2024. Available at: https://www.researchgate.net/publication/392134668_Towards_Better_Instruction_Following_Retrieval_Models
- Tingyu Song, Guo Gan, Mingsheng Shang, and Yilun Zhao. *IFIR: A Comprehensive Benchmark for Evaluating Instruction-Following in Expert-Domain Information Retrieval.* In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 10186–10204, Albuquerque, New Mexico. Association for Computational Linguistics. Available at: https://aclanthology.org/2025.naacl-long.511/

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009