[MUVERA] How to Run a Multi-Vector Retrieval Analysis with Screaming Frog

Metehan Yesilyurt

AI Search & Information Retrieval Researcher · metehan.ai

Published: July 01, 2025 • Canonical: https://metehan.ai/blog/screaming-frog-muvera-analysis/

Google's MUVERA (Multi-Vector Retrieval via Fixed Dimensional Encodings) represents a breakthrough in making complex multi-vector retrieval as fast as single-vector search. I wanted to build a Custom Javascript Snippet for Screaming Frog implementation aligns with the core principles of MUVERA research.

Read Google's research here: https://research.google/blog/muvera-making-multi-vector-retrieval-as-fast-as-single-vector-search/[1] It's not live yet.

Why This Implementation Is Experimental

Think of this as an inspired adaptation rather than an exact copy of Google's MUVERA system. Here's the thing: Google built MUVERA to make their massive search infrastructure faster, but we're using those same ideas in a completely different way, to help optimize content.

We don't have access to Google's secret sauce, their actual algorithms, calculations, or methods(even leaks). Instead, I am making educated guesses based on what they've shared publicly and what we know about how embeddings typically work. Those configuration numbers? They're my best estimates, not battle-tested values from Google's labs. I assume that there are many clever and more technical people who read this post and make it even better.

This experimental approach is actually pretty exciting. We're trying to adjust ourselves a new way to look at content quality through the "lens" of advanced retrieval systems. Test them out, see what works for your content, have an idea or make a plan.

It's a bit like being an early explorer. I'm charting this area by applying the newest research in ways. That makes this tool both innovative and a work in progress. The insights it provides are valuable, but they come with the caveat that we're all learning together what works best in the real world.

And of course, special thanks to Screaming Frog. Dan, Patrick... all the team! They've built the most flexible SEO tool for technical needs.

Let's start.

Access the custom javascript snippet here[2]. You need to enable JavaScript rendering and have an API key. Get here.[3]

Here is a full page level output: https://metehan.ai/muvera-sample.json[4]

muvera-screaming-retrieval

What I know and didn't know?

You may think I know everything mentioned in technical details here. No. I love Python and Javascript. I built many scripts, played with methods on my GitHub[5]. I know(let's say 6.5/10) how re-ranking works with weights, computional power, embeddings, fundamentals of LLM models. I even built this script with Claude Code almost over 50 failed tries. Learning Cursor, Python, Streamlit is highly recommended.

Many things are just new for me to work on it. I don't say I know every aspect of how MUVERA works or its elements like FDE, MIPS. I'm trying to learn more about how search works, everyday. Thanks to Dan Petrovic, I learn a lot from his finding, experiments. I ask a lot.

I should also say Mike King and Andrea Volpini, Emilia Gjorgjevska' works are very inspiring for me.

Is it live?

According to SearchEngineJournal: "Although the announcement did not explicitly say that it is being used in search, the research paper makes it clear that MUVERA enables efficient multi-vector retrieval that appears suitable for large-scale applications by reducing the problem to single-vector MIPS, allowing the use of off-the-shelf retrieval systems (existing infrastructure) and achieving lower latency and memory usage." Read here.[6]

1. Configuration Block: Optimizing for Vector Embeddings

```
TEXT
```

```
const CONFIG = {
    TARGET_LENGTH: 150,    // Optimal for vector embeddings
    MIN_LENGTH: 50,        // Minimum semantic coherence
    MAX_LENGTH: 250,       // Maximum before complexity loss
    OVERLAP: 30,          // Context preservation
    TEXT_PREVIEW: 300,    // Gemini analysis preview
    VECTOR_DIMENSIONS: 768 // Standard embedding size
};
```

Understanding MUVERA Configuration Parameters: Why These Numbers Matter

TARGET_LENGTH: 150 - The Sweet Spot for Vector Embeddings

Why 150 words? This aligns with MUVERA's challenge of "increased embedding volume." Google's research notes that "generating embeddings per token drastically increases the number of embeddings to be processed."

At 150 words:

- **Semantic Completeness**: Long enough to capture a complete thought or concept (typically 5-8 sentences)
- **Computational Efficiency**: Not so long that the embedding becomes computationally expensive
- **Retrieval Precision**: Matches the typical length of a search query's ideal answer snippet
- **Vector Density**: Creates dense, meaningful vectors without sparse representations

This is trying to mirror how ColBERT (mentioned in MUVERA) handles passages - not too granular (token-level) but not too broad (document-level).

MIN_LENGTH: 50 - Minimum Semantic Coherence Threshold

Why 50 words minimum? MUVERA emphasizes that "a document might have a token with high similarity to a single query token, but overall, the document might not be very relevant."

At 50 words:

- **Semantic Validity**: The minimum needed to form a coherent idea (2-3 sentences)
- **Context Sufficiency**: Enough words to establish topic and intent
- **Noise Reduction**: Filters out fragments that would create poor embeddings
- **Chamfer Similarity**: Provides enough tokens for meaningful similarity calculations

Passages shorter than this would create unreliable vectors that could mislead the retrieval system.

MAX_LENGTH: 250 - The Complexity Ceiling

Why cap at 250 words? This addresses MUVERA's "complex and compute-intensive similarity scoring" challenge.

Beyond 250 words:

- **Semantic Drift**: Multiple concepts start appearing, diluting the vector's focus
- **Computational Cost**: Chamfer matching becomes exponentially more expensive
- **Retrieval Accuracy**: Longer passages match too many queries, reducing precision
- **Vector Ambiguity**: The embedding starts representing multiple semantic spaces

This aligns with MUVERA's goal of maintaining "efficient retrieval at scale."

OVERLAP: 30 - Context Preservation Between Passages

Why 30-word overlap? MUVERA uses "randomized partitioning scheme" because "we don't know the optimal matching between query and document vectors beforehand."

The 30-word overlap:

- **Semantic Continuity**: Preserves ~20% context between adjacent passages
- **Boundary Handling**: Ensures important concepts split across passages aren't lost
- **Query Matching**: Increases chances of matching queries that span passage boundaries
- **FDE Robustness**: Creates more robust Fixed Dimensional Encodings by maintaining relationships

This implements MUVERA's space partitioning while maintaining semantic coherence.

TEXT_PREVIEW: 300 - Gemini Analysis Window

Why 300 characters for preview? While not directly from MUVERA, this supports the analysis phase.

At 300 characters:

- **Semantic Sampling**: Enough to understand passage content (~50-60 words)
- **API Efficiency**: Keeps prompt size manageable for faster processing
- **Quality Assessment**: Sufficient for Gemini to evaluate vector quality
- **Pattern Recognition**: Allows identification of content type and structure

This enables efficient quality assessment without processing entire passages.

VECTOR_DIMENSIONS: 768 - (Almost) Industry Standard Embedding Size

Why 768 dimensions? This matches modern transformer-based embedding models that MUVERA would use.

The 768 dimensions:

- **BERT Compatibility**: Matches BERT-base architecture (12 layers × 64 dimensions)
- **Semantic Richness**: Sufficient dimensionality to capture nuanced meaning
- **Computational Balance**: Not as heavy as 1024-dim models, but richer than 512
- **FDE Transformation**: Provides good input for MUVERA's dimension reduction to FDE

This ensures compatibility with the "highly-optimized MIPS algorithms" MUVERA leverages.

How These Parameters Work Together

These configurations create an optimal pipeline for MUVERA's three-step process:

1. **Multi-Vector Generation**: TARGET_LENGTH and boundaries ensure each passage creates a high-quality vector
2. **FDE Creation**: OVERLAP and VECTOR_DIMENSIONS provide rich input for space partitioning
3. **Efficient Retrieval**: MIN/MAX constraints ensure the search space remains manageable

The result: Content perfectly structured for "reducing complex multi-vector retrieval back to single-vector maximum inner product search" - exactly what MUVERA achieves.

MUVERA Alignment: Google's research emphasizes that multi-vector models generate "multiple embeddings per query or document, often one embedding per token." This configuration block establishes optimal passage lengths (150 words) that balance between:

- Having enough semantic content for meaningful embeddings
- Avoiding overly complex passages that would dilute vector quality
- Maintaining the 768-dimensional standard that aligns with modern embedding models

The overlap parameter (30 words) ensures context preservation between passages, addressing MUVERA's goal of maintaining semantic relationships while keeping vectors independent.

2. Semantic Passage Extraction: Creating Multi-Vector Representations

```
TEXT
function extractSemanticPassages() {
    const clone = document.body.cloneNode(true);
    clone.querySelectorAll('script, style, noscript, nav, header, footer, .ads, .sidebar').forEach
    // ... targeting semantic content elements
}
```

MUVERA Alignment: This mirrors MUVERA's approach to creating "multi-vector sets" where each set describes a datapoint. By removing non-content elements and targeting semantic HTML elements, we ensure each passage represents a coherent semantic unit - exactly what MUVERA requires for effective multi-vector retrieval.

3. Semantic Weight Calculation: Approximating Chamfer Similarity

```
TEXT
function calculateSemanticWeight(element, text) {
    let weight = 1.0;
    // Element importance, content quality indicators, semantic richness
    const uniqueWords = new Set(text.toLowerCase().split(/\s+/));
    const lexicalDiversity = uniqueWords.size / text.split(/\s+/).length;
    weight += lexicalDiversity * 0.5;
    return Math.round(weight * 100) / 100;
}
```

MUVERA Alignment: Google's research describes Chamfer similarity as measuring "the maximum similarity between each query embedding and the closest document embedding." This semantic weight calculation pre-computes factors that will influence vector similarity:

- Element importance (h1 = 3.0, p = 1.0) approximates hierarchical relevance
- Lexical diversity ensures rich semantic content for better embeddings
- Query intent indicators (questions, interrogatives) align with retrieval objectives

4. Optimal Passage Creation: Space Partitioning Strategy

```
TEXT
function createOptimalPassages(textBlocks) {
    // Smart sentence-based chunking
    if (words.length > CONFIG.MAX_LENGTH) {
        const sentences = splitIntoSentences(block.text);
        // Maintain context overlap
        tempBuffer = tempBuffer.slice(-CONFIG.OVERLAP).concat(sentWords);
    }
}
```

MUVERA Alignment: This implements MUVERA's "space partitioning" concept. The research states: "The core idea behind FDE generation is to partition the embedding space into sections." Our passage creation:

- Partitions content into optimal chunks (space partitioning)

- Maintains context overlap (addressing the "randomized partitioning scheme")
- Ensures passages are neither too sparse nor too dense for effective embeddings

5. Vector Quality Assessment: Preparing for Fixed Dimensional Encodings

```
TEXT
function assessVectorQuality(text, wordCount, semanticWeight) {
    // Optimal length for vector embeddings
    const lengthOptimal = Math.max(0, 100 - Math.abs(wordCount - CONFIG.TARGET_LENGTH) * 2);
    // Query-answering potential
    if (text.match(/\b(what|how|why|when|where|who)\b/i)) score += 15;
}
```

MUVERA Alignment: MUVERA's FDE approach requires high-quality input vectors. This function ensures passages will generate effective embeddings by:

- Optimizing for ideal vector embedding lengths (avoiding the "increased embedding volume" challenge)
- Prioritizing query-answerable content (aligning with IR objectives)
- Assessing lexical diversity for richer vector representations

6. Retrieval Score Calculation: MIPS Optimization

```
TEXT
function calculateRetrievalScore(text, wordCount) {
    // Content type scoring
    if (text.match(/\b(step|method|process|guide|tutorial)\b/i)) score += 20;
    // Question-answer format
    if (text.includes('?') && text.length > 100) score += 20;
}
```

MUVERA Alignment: This directly supports MUVERA's "MIPS-based retrieval" phase. The research notes that "FDEs of documents are indexed using a standard MIPS solver." By pre-calculating retrieval scores, we:

- Identify passages most likely to match user queries
- Prioritize content formats that perform well in retrieval systems
- Enable efficient candidate selection before re-ranking

7. Gemini Analysis Integration: Multi-Vector to FDE Transformation

```
TEXT
```

```
function analyzeWithGemini(passages) {
    const passageData = passages.map(p => ({
        id: p.id,
        vector_quality: p.vector_quality,
        retrieval_score: p.retrieval_score,
        semantic_weight: p.semantic_weight
    }));
}
```

MUVERA Alignment: This represents the transformation from multi-vector representations to analyzable features - conceptually similar to MUVERA's FDE generation. The research describes "mappings to convert query and document multi-vector sets into FDEs." Our implementation:

- Aggregates multiple passage vectors into analyzable metrics
- Preserves essential similarity information in a fixed format
- Enables rapid analysis without processing all original vectors

8. MUVERA Analysis Requirements: Implementing the Complete Pipeline

The comprehensive analysis prompt implements MUVERA's three-stage approach:

1. **FDE Generation** → "VECTOR EMBEDDING OPTIMIZATION"
2. **MIPS-based Retrieval** → "MULTI-VECTOR RETRIEVAL STRATEGY"
3. **Re-ranking** → "Top 10 passages for primary vector index"

9. Output Formatting: Performance Metrics Alignment

```
TEXT
```

```
const qualityTiers = {
    excellent: passages.filter(p => p.vector_quality >= 80),
    good: passages.filter(p => p.vector_quality >= 60 && p.vector_quality  p.vector_quality < 60)
};
```

MUVERA Alignment: Google's research emphasizes "achieving better recall while retrieving significantly fewer candidate documents." This tiered approach ensures:

- Only high-quality passages enter the vector index (reducing computational load)
- Clear identification of optimization opportunities
- Metrics that directly correlate with retrieval performance

Key MUVERA Principles Implemented

1. **Multi-Vector Independence**: Each passage is treated as an independent semantic unit

2. **Space Partitioning**: Content is intelligently chunked into optimal segments
3. **Quality-First Indexing**: Only high-quality passages are recommended for primary index
4. **Efficient Retrieval**: Pre-computed scores enable fast candidate selection
5. **Semantic Preservation**: Context overlap and weight calculations maintain meaning

Performance Benefits Aligned with MUVERA

Google reported that MUVERA achieves "10% higher recall with a remarkable 90% reduction in latency." This implementation supports similar gains by:

- Pre-computing vector quality metrics (reducing runtime calculations)
- Identifying optimal passages for indexing (reducing search space)
- Maintaining semantic coherence (improving recall)
- Structuring content for efficient retrieval (reducing latency)

Conclusion

This implementation successfully translates Google's MUVERA research into a practical SEO tool. By focusing on creating high-quality, independent passage vectors and pre-computing retrieval metrics, it achieves MUVERA's core goal: making multi-vector retrieval as fast and effective as single-vector search, while maintaining the semantic richness that makes multi-vector approaches superior for information retrieval.

Muvera multi-vector retrieval is the future-shape of AEO. AI citations in ChatGPT, Perplexity and Gemini are moving toward multi-vector retrieval, which means answer engine optimization has to plan for it now.

REFERENCES & SOURCES

[1] **https://research.google/blog/muvera-making-multi-vector-retrieval-as-fast-as-single-vec...** — https://research.google/blog/muvera-making-multi-vector-retrieval-as-fast-as-single-vector-search/

[2] **javascript snippet here** — https://github.com/metehan777/screaming-frog-muvera-analysis

[3] **Get here.** — https://aistudio.google.com/apikey

[4] **https://metehan.ai/muvera-sample.json** — https://metehan.ai/muvera-sample.json

[5] **GitHub** — https://github.com/metehan777

[6] **Read here.** — https://www.searchenginejournal.com/googles-new-muvera-algorithm-improves-search/550070/