BINARY SEMANTIC INDEXING FOR GENERATIVE RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval systems must organize large collections of knowledge while allowing queries to reach relevant information efficiently. As retrieval increasingly relies on learned representations and generative models, the representation itself becomes central to both semantic modeling and access. This raises a broader question: can different retrieval mechanisms share a common semantic foundation? We introduce **Binary Semantic Indexing (BSI)**, a framework that connects sparse indexing and generative retrieval through a **shared binary semantic space**. A jointly trained semantic encoder and binary spherical quantizer compress each document into a compact code that preserves retrieval information. The code's local components form addressable units in an inverted index. Complete-code and local-address generators provide complementary query interfaces, while direct query encoding accesses the same representation. By **separating document organization from query modeling**, BSI allows these mechanisms to operate over a common document space. Experiments show effective retrieval across query interfaces and domains, while controlled analyses characterize the tradeoffs between code capacity, index access, and retrieval quality.

1 INTRODUCTION

A long-standing goal of information retrieval is to organize and access large collections of knowledge through a common representation that is compact and efficient to search. As retrieval has evolved from term matching to semantic representation learning, the question has broadened from how to match surface forms to how to construct a representation space that supports **both semantic modeling and efficient search**. Sparse retrieval uses discrete features and inverted indexes for efficient access (Robertson & Zaragoza, 2009; Formal et al., 2021b); dense retrieval encodes semantic relationships in continuous vector spaces (Karpukhin et al., 2020); and generative retrieval represents documents through discrete identifiers that a query model can predict (Tay et al., 2022; Wang et al., 2022). Despite their differences, these approaches raise a common question: how should knowledge be represented to accommodate evolving ways of searching it?

This question matters because a retrieval representation does more than determine similarity scores. It shapes how knowledge is organized, which structures can be searched efficiently, and how queries interact with a corpus. Existing paradigms approach this problem from different directions: sparse representations emphasize addressable features, continuous representations offer rich semantic structure, and discrete semantic identifiers enable access through generation. Figure 1 illustrates the broader goal we pursue: **a common semantic foundation for multiple retrieval and query mechanisms**.

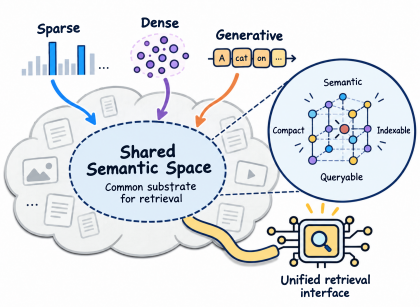


Figure 1: A shared semantic space.

This leads to the central question of this work: **Can different query and retrieval mechanisms share a discrete semantic space?**

Discrete representations offer a natural starting point. They can compress high-dimensional continuous repre-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sentations into compact, structured codes, reducing the cost of storing and organizing large document collections (Jégou et al., 2011; Yamada et al., 2021). Their constituent units can also be combined, addressed, and predicted, providing a basis for **both efficient indexing and generative access**. Recent learned sparse retrieval methods show that indexing units can extend beyond words to learned semantic features (Formal et al., 2026; Guo et al., 2026). Work on generative retrieval likewise shows how structured discrete representations can be predicted to access documents (Bevilacqua et al., 2022; Sun et al., 2023; Zhang et al., 2024). Together, these developments suggest that discrete representations can serve a broader role in retrieval: connecting semantic modeling, knowledge organization, and query mechanisms within a common framework.

For this purpose, a discrete representation must satisfy several requirements. It should preserve semantic information in a compact form, so that large collections remain practical to store and organize. It should have a well-defined geometry that permits meaningful comparisons between discrete codes. It also needs local structure that queries can manipulate to access the space efficiently (Norouzi et al., 2014). If these structures are suitable for prediction, the same representation may support both direct and generative retrieval, as well as other query mechanisms. The structure of the representation itself thus becomes central to connecting these forms of retrieval.

Binary Spherical Quantization (BSQ) (Zhao et al., 2025b) provides a suitable foundation for exploring this direction. BSQ maps continuous representations to compact codes in a regular binary space. This space has an explicit geometry: the inner product between normalized binary codes is exactly related to their Hamming distance, giving the discrete representation a clear similarity structure. Binary codes also have a natural component structure. A complete code can represent a document's overall semantics, while its components can serve as discrete units for indexed lookup (Yamada et al., 2021; Norouzi et al., 2014). These properties offer a basis for representations that combine **semantic structure, compactness, and indexability**, while allowing generative models to operate in the same discrete space.

We therefore propose **Binary Semantic Indexing (BSI)**, a framework that develops binary semantic representations into a shared foundation for retrieval. BSI organizes documents in a common binary semantic space and allows different query mechanisms to access it. A query encoder can map a query directly into this representation, or a generative model can predict the discrete semantic information used to retrieve documents from it. Both direct query quantization and the generative query interfaces use this shared representation. This design **separates the organization of documents from the choice of query mechanism**: the semantic space provides a common basis for organizing knowledge, while each interface determines how a query accesses it.

This view connects sparse and generative retrieval. Sparse retrieval offers efficient access to large collections through structured indexes (Formal et al., 2021b; Gao et al., 2021); generative retrieval offers a flexible way to predict the semantic information needed for search (Zeng et al., 2024a;b). With a common discrete representation, a generative model can act as **a query interface to a semantic index**, and efficient indexing can provide the search infrastructure for generative retrieval. Different retrieval paradigms can then be studied as query mechanisms over a shared representation. We hope this perspective encourages the joint design of semantic representations, knowledge organization, and query models, and opens possibilities for more flexible retrieval systems.

Our main contributions are as follows:

1. **A binary representation perspective on unified semantic retrieval.** Starting from the joint requirements of compactness, semantic structure, and indexability, we examine how discrete representations can provide a shared foundation for different retrieval mechanisms and introduce BSI as a framework for this purpose.

2. **A shared representation framework for sparse and generative retrieval.** BSI allows different query mechanisms to access a common binary semantic space, providing a shared basis for document representation and indexing, and showing how a generative model can serve as a query interface to a semantic index.

3. **An empirical study across retrieval settings.** Through different query interfaces and standard retrieval tasks, we study how binary semantic representations support multiple forms of retrieval and examine the behavior of the shared semantic index across Natural Questions, MS MARCO, and BEIR.

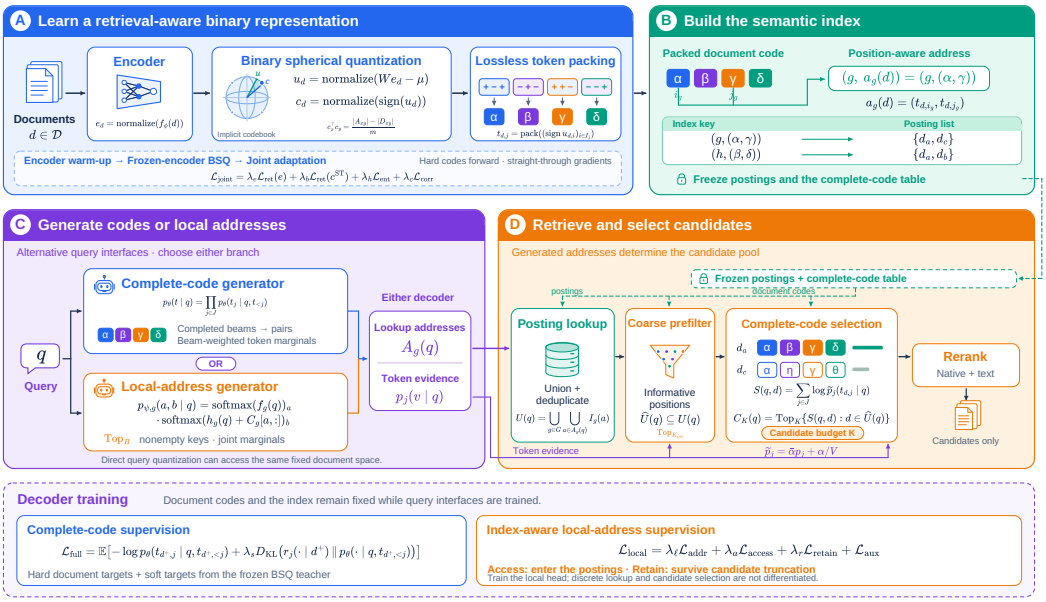


Figure 2: Framework of BSI.

2 Related Work

Generative retrieval. Generative retrieval treats access to a collection as prediction over discrete document representations. GENRE retrieves entities by generating their names, while DSI and NCI establish document retrieval through identifier prediction (De Cao et al., 2021; Tay et al., 2022; Wang et al., 2022). Subsequent work increasingly treats identifier construction as representation learning, learning discrete document semantics or incorporating query–document relevance into the identifier space (Sun et al., 2023; Jin et al., 2024; Zeng et al., 2024a). SEAL and TSGen further show that generated substrings or term sets can provide structured access to documents through an external index (Bevilacqua et al., 2022; Zhang et al., 2024). More recent studies strengthen the connection between generation and document relevance (Lu et al., 2025; Mekonnen et al., 2025), extend retrieval to zero-shot settings (Sun et al., 2026), and make retrieval decisions interpretable through hierarchical category paths (Lee et al., 2026). Shared semantic IDs also support joint search and recommendation (Penha et al., 2025). BSI examines sharing across access mechanisms: direct encoding and generative querying of the same document index.

Sparse retrieval. Sparse retrieval combines selective feature matching with efficient inverted indexes, from classical term weighting such as BM25 (Robertson & Zaragoza, 2009) to learned lexical representations. DeepCT and DeepImpact learn context-dependent term weights, while SPLADE learns term weighting and expansion (Dai & Callan, 2020; Mallia et al., 2021; Formal et al., 2021b). Subsequent work improves these representations through stronger training and, in CSPLADE, causal language models (Formal et al., 2022; Lassance et al., 2024; Xu et al., 2025). A complementary line asks whether index dimensions must correspond to words at all. SNRM learns latent sparse representations compatible with inverted indexing (Zamani et al., 2018); more recently, SPLARE uses sparse-autoencoder features as a semantic indexing vocabulary, and SSR maps token embeddings to sparse semantic features for multi-vector retrieval (Formal et al., 2026; Guo et al., 2026). BSI extends this direction toward **shared representations that are both indexable and predictable**. Appendix A further traces how learned identifiers and semantic features connect generation with indexed access.

3 BINARY SEMANTIC INDEXING

Given a query q and a collection $\mathcal{D}$, BSI retrieves documents through a shared binary semantic space (Figure 2). **Each document code is both a semantic representation and a source of indexable units.** We learn these codes, then fix them while training complete-code and local-address generators. Both interfaces retrieve candidates through predicted index addresses and apply the same selection rule.

3.1 LEARNING THE BINARY SEMANTIC REPRESENTATION

Learning retrieval semantics. A shared encoder f_ϕ maps a query or document x to an embedding $e_x = \mathrm{normalize}(f_\phi(x))$. A learned projection reads the final non-padding transformer state. Before quantization, we train the encoder to distinguish relevant documents within a batch. For representations r and batch positives $\mathcal{P}_\mathcal{B}(q)$, the query-to-document loss is

$$\mathcal{L}_{q\to d}(r) = -\mathbb{E}_q \log \frac{\sum_{d\in\mathcal{P}_\mathcal{B}(q)} \exp(r_q^\top r_d/\tau)}{\sum_{d\in\mathcal{D}_\mathcal{B}} \exp(r_q^\top r_d/\tau)}.$$

Here $\mathcal{D}_\mathcal{B}$ is the batch document collection and τ is a temperature. Let $\mathcal{L}_{\mathrm{ret}}(r)$ denote the average of this loss and its document-to-query counterpart. Summing probability mass over relevant entries avoids treating another known positive as a negative. Encoder training minimizes $\mathcal{L}_{\mathrm{ret}}(e)$.

Learning discrete codes. BSQ maps the continuous representation to the unit-normalized vertices of a binary hypercube. With projection W and a running centering vector μ, the spherical projection, hard code, and straight-through representation are

$$u_x = \mathrm{normalize}(We_x - \mu), \qquad c_x = \mathrm{normalize}(\mathrm{sign}(u_x)), \qquad c_x^{\mathrm{ST}} = u_x + \mathrm{sg}(c_x - u_x).$$

The operator sg stops gradients: the forward pass uses the hard code, while the backward pass follows u_x. We first freeze the encoder and learn W, then unfreeze the encoder and optimize the representation end to end:

$$\mathcal{L}_{\mathrm{BSQ}} = \lambda_b\mathcal{L}_{\mathrm{ret}}(c^{\mathrm{ST}}) + \lambda_h\mathcal{L}_{\mathrm{ent}} + \lambda_c\mathcal{L}_{\mathrm{corr}},$$
$$\mathcal{L}_{\mathrm{joint}} = \lambda_e\mathcal{L}_{\mathrm{ret}}(e) + \mathcal{L}_{\mathrm{BSQ}}.$$

Entropy regularization encourages confident assignments and varied bit use; correlation regularization discourages redundancy. Joint training updates the entire encoder and BSQ projection under both continuous and quantized retrieval supervision, while the center follows a running mean. Appendix C defines the regularizers and schedule.

From codes to semantic units. For code dimension m, Hamming distance determines the inner product between normalized codes exactly. Retrieval training gives this geometry semantic meaning. Packing groups of w bits into tokens gives $t_d = (t_{d,j})_{j\in\mathcal{J}} \in \mathcal{V}^n$, where $\mathcal{V}$ is the token vocabulary, $\mathcal{J}$ is the ordered position set, $n = |\mathcal{J}|$, and $m = nw$. Packing preserves the full code. **The complete sequence represents the document, while combinations of its tokens form searchable semantic units.** Appendix B develops the geometry and the soft token distributions used as training targets.

3.2 BUILDING AND QUERYING THE SEMANTIC INDEX

Local addresses. An inverted index lets a prediction retrieve documents through partial agreement with their codes. Let $\mathcal{G}$ partition token positions into fixed disjoint pairs. For each pair $g = (i_g, j_g)$, the document address and its posting list are

$$a_g(d) = (t_{d,i_g}, t_{d,j_g}), \qquad I_g(a) = \{d \in \mathcal{D} : a_g(d) = a\}.$$

Keys include the position pair. Documents activate a sparse subset of addresses, and each posting preserves all matching document identifiers.

Lookup and selection. A decoder proposes addresses $A_g(q)$ to find candidates and token distributions $p_j(v \mid q)$ to distinguish documents sharing a local match. The lookup union and candidate score are

$$U(q) = \bigcup_{g \in \mathcal{G}} \bigcup_{a \in A_g(q)} I_g(a), \qquad S(q,d) = \sum_{j \in \mathcal{J}} \log \widetilde{p}_j(t_{d,j} \mid q).$$

Here $\widetilde{p}_j$ mixes p_j with a uniform distribution to temper uncertain predictions. Code evidence selects a budgeted candidate set from $U(q)$; relevance scoring can then order these candidates. Appendix D specifies the selection rules.

3.3 Complete-code generation

The complete-code decoder learns to predict a relevant document's token sequence from the query. Its autoregressive distribution is

$$p_\theta(t \mid q) = \prod_{j \in \mathcal{J}} p_\theta(t_j \mid q, t_{<j}).$$

Observed token pairs in completed beams supply lookup addresses; beam-score-weighted token frequencies supply p_j for selection. The index resolves these local components into documents.

Training the decoder. Hard document codes specify the desired sequence, but discard uncertainty near a quantization boundary. We therefore also use soft token targets $r_j(\cdot \mid d^+)$ obtained from the frozen BSQ projection of a positive document d^+. With teacher-forced prefixes, training minimizes

$$\mathcal{L}_{\text{full}} = \mathbb{E}_{q,d^+,j}\Big[-\log p_\theta(t_{d^+,j} \mid q, t_{d^+,<j}) + \lambda_s \, D_{\text{KL}}\big(r_j(\cdot \mid d^+) \,\|\, p_\theta(\cdot \mid q, t_{d^+,<j})\big) \Big].$$

Likelihood teaches the discrete target, while distillation preserves uncertainty over alternative token values. Only the generator is updated; the encoder, quantizer, and document targets remain fixed.

3.4 Local-address generation

The local decoder predicts a joint token distribution for each position pair. Query-conditioned logits $f_g(q)$ and $h_g(q)$ provide the base predictions, and a learned conditional table C_g captures dependence within an address:

$$p_{\psi,g}(a, b \mid q) = \text{softmax}(f_g(q))_a \; \text{softmax}(h_g(q) + C_g[a, :])_b.$$

For each pair, the decoder selects the highest-probability nonempty addresses under budget B. Summing this joint distribution over either token yields the marginals used in $S(q,d)$. The query backbone is first trained to predict token-level evidence; it is then frozen while the local head learns conditional addresses.

Learning addresses that retrieve documents. Positive-address likelihood provides the basic supervision:

$$\mathcal{L}_{\text{addr}} = -\mathbb{E}_{(q,d^+)\in\mathcal{P},\, g\in\mathcal{G}} \log p_{\psi,g}(a_g(d^+) \mid q),$$

where $\mathcal{P}$ contains positive query–document pairs. We also train for access and candidate retention. The margin $M_{\text{access}}(q, d^+)$ is a normalized smooth maximum over positive-address margins at probe boundaries; $M_{\text{keep}}(q, d^+, s)$ compares the positive code score with stored boundary competitors at selection stage s. With access temperature T_a, the losses are

$$\mathcal{L}_{\text{access}} = \mathbb{E}_{q,d^+} T_a \, \text{softplus}(-M_{\text{access}}/T_a), \qquad \mathcal{L}_{\text{retain}} = \mathbb{E}_{q,d^+,s} \, \text{softplus}(-M_{\text{keep}}).$$

The access loss encourages reaching a relevant document through an available posting. The retention loss encourages preserving it through the prefilter and final code shortlist. Together with auxiliary query-teacher, relevance-teacher, and posting-cost supervision, the local objective is

$$\mathcal{L}_{\text{local}} = \lambda_\ell \mathcal{L}_{\text{addr}} + \lambda_a \mathcal{L}_{\text{access}} + \lambda_r \mathcal{L}_{\text{retain}} + \mathcal{L}_{\text{aux}}.$$

Local generation is trained for both semantic prediction and the retrieval decisions that follow it. The document representation and index remain fixed throughout this stage. Appendix C specifies the margins, auxiliary targets, and trainable parameters for each stage.

4 EXPERIMENTS

Our experiments examine whether a shared binary semantic space can support different query mechanisms while retaining useful retrieval quality and a compact index. We organize the evaluation around four research questions:

- **RQ1:** Can both generative interfaces retrieve effectively from the same document space?
- **RQ2:** Does this representation remain useful across collections and query mechanisms?
- **RQ3:** How much retrieval quality can a compact, directly indexable binary code preserve?
- **RQ4:** How do representation and query-access budgets affect retrieval coverage?

4.1 EXPERIMENTAL SETUP

Datasets and metrics. We evaluate on Natural Questions (NQ) (Kwiatkowski et al., 2019), MS MARCO Passage (Bajaj et al., 2016), and all thirteen BEIR datasets in our comparison (Thakur et al., 2021), including Touché. The primary metrics are Recall@10 for NQ, Recall@10 and MRR@10 for MS MARCO, and nDCG@10 for BEIR. BEIR averages weight each dataset equally. Six MAIR tasks (Sun et al., 2024) provide a further comparison with dense retrieval. Recall@10 measures whether a relevant document occurs in the first ten results; MRR@10 also reflects its rank. Full metric definitions and shared evaluation settings are given in Appendix E.

Comparisons. We compare with classical and learned sparse retrieval, dense retrieval, and generative methods, including recent DOGR, PDMR, and SSR results (Lu et al., 2025; Oussaidene & Boughanem, 2026; Guo et al., 2026). BSI denotes the complete retrieval system, with Local and Global distinguishing its two generative interfaces. Main tables report the best recorded BSI results; BEIR uses target-aware per-dataset selection. Published scores provide context: NQ splits and MS MARCO recall definitions differ from ours, so they do not establish matched rankings. Appendix E specifies these distinctions and the complete-system evaluation scope.

4.2 GENERATIVE RETRIEVAL EFFECTIVENESS (**RQ1**)

Document and passage retrieval. BSI reaches **94.28 Recall@10 with Local** and **94.15 with Global** on NQ (Table 1). The interfaces achieve closely matched quality despite predicting different objects: complete document codes in Global and local index addresses in Local.

On MS MARCO, Local achieves 69.86 Recall@10 and **41.98 MRR@10**, while Global reaches **71.67 Recall@10** and 41.23 MRR@10 (Table 2). Global retrieves a relevant passage within the first ten for more queries, whereas Local achieves a higher mean reciprocal rank. Together, the NQ and MS MARCO results answer **RQ1**: **both generative interfaces support effective retrieval from a shared binary space**.

Access through the shared index. Table 7 in Appendix F isolates binary-index coverage from complete-system scoring: NQ Global and Local reach 95.41 and 96.47 Recall@2000; MS MARCO Local reaches 93.21. The held-out NQ analysis in the same appendix reports 96.17 coverage for Local. These results show that generation reaches relevant documents through fixed codes, although coverage alone does not explain the final ranking gains.

4.3 EVALUATION ACROSS COLLECTIONS (**RQ2**)

Table 3 reports BEIR-13 with separate panels for the 0.6B and 8B encoders. Its tasks span argument retrieval, scientific search, question answering, and fact verification. Queries use direct encoding here, testing the representation through another interface on collections with different relevance criteria.

Performance across domains. BSI obtains **57.63 mean nDCG@10 with the 0.6B encoder** and **67.84 with the 8B encoder**. The larger system exceeds the SSR-CLS reference on eight of thirteen datasets, with macro averages of 67.84 and 67.54. It reaches 83.23 on ArguAna, 93.96 on TREC-COVID, and 84.94 on HotpotQA, although gains remain task dependent. The smaller system also

Table 1: NQ Recall@10 (%). Published methods use the 7,830-query test set; BSI uses 15,324 calibration queries. The splits differ; these are not matched comparisons.

Method	Recall@10
BM25[G]	60.30
DocT5Query[G]	69.30
UniCOIL[T]	86.20
SPLADEv2[T]	87.30
DPR[G]	77.70
ANCE[G]	78.50
Sentence-T5[G]	83.00
GTR[G]	84.40
DSI[T]	71.50
GENRE[T]	75.60
DSI-QG[G]	80.70
CGR-Contra[G]	81.10
SEAL[T]	81.20
Ultron (URL)[T]	85.40
NCI[G]	85.70
GLEN[L]	86.00
MINDER[T]	86.90
GenRet[G]	88.80
NOVO[V]	89.70
TSGen (2024)[T]	88.90
LTRGR (2024)[D]	88.30
DOGR (2025)[D]	89.10
Few-Shot GR[F]	87.60
HyPE (2026)[H]	84.20
PDMR (2026)[P]	85.80
BSI Local	**94.28**
BSI Global	94.15

Local and Global denote local-address and complete-code generation; bold compares these interfaces only. Superscripts give reporting sources: G: GenRet (Sun et al., 2023); T: TSGen (Zhang et al., 2024); L: GLEN (Lee et al., 2023); V: NOVO (Wang et al., 2023); D: DOGR (Lu et al., 2025); F: Few-Shot GR (Askari et al., 2026); H: HyPE (Lee et al., 2026); P: PDMR (Oussaidene & Boughanem, 2026). Sentence-T5 and GTR use Base models; LTRGR uses the keyword/T5-base variant; Few-Shot GR uses Llama-3-8B; HyPE uses atomic identifiers.

Table 2: MS MARCO Passage results (%). BSI uses all 6,980 development queries. Recall definitions differ between panels (see below).

Published methods

Method	Recall@10[d]	MRR@10
TSGen[T]	—	19.50
LTRGR[L]	53.10	25.50
DGR[D]	—	26.60
RIPOR[R]	56.20	33.30
PAG[P]	67.00	38.50
DOGR[O]	—	22.50
PAG (repr.)[Q]	67.10	38.60

BSI

Interface	Recall@10[q]	MRR@10
Local	69.86	**41.98**
Global	**71.67**	41.23

Local and Global generate local addresses and complete codes, respectively.

[d]Document recall; [q]query-level recall (at least one relevant result). Bold marks the best scores among the BSI interfaces; — denotes an unreported metric. T: Zhang et al. (2024); L: Li et al. (2024a); D: Li et al. (2024b); R: Zeng et al. (2024a); P: Zeng et al. (2024b); O: Lu et al. (2025); Q: the independent PAG reproduction of Mekonnen et al. (2026).

supports the full suite, compared with 53.99 for the BERT-based SSR-CLS reference. These are comparisons of complete systems and their best recorded results, rather than controlled measures of encoder capacity or target-blind transfer.

Preserving dense retrieval quality. Table 14 in Appendix G compares binary and dense access with the same encoder and relevance scoring. Across six tasks and 419 queries, BSI and Dense obtain **38.92 and 38.91 mean nDCG@10**. The observed averages are comparable; this small difference does not establish superiority or statistical equivalence.

For **RQ2**, direct access remains useful across domains. Table 13 further separates uniform scoring from per-dataset selection on BEIR-12; neither comparison establishes target-blind transfer.

Table 3: BEIR-13 nDCG@10 (%), with target-aware per-dataset BSI selection and published baselines (Guo et al., 2026). Sizes refer to encoders. Dataset names and shared settings are in Appendix E. Bold marks the best score in each panel.

(a) BSI with a 0.6B encoder and compact retrieval baselines

Method	AR	CF	CV	DB	FE	FQ	HQ
ColBERTv2	46.30	17.30	73.50	44.90	78.70	36.10	66.60
SPLADE-v2	47.60	23.50	71.50	43.40	78.70	33.80	68.40
SPLADE-v3	50.80	23.30	74.80	45.20	79.80	37.50	69.20
SSR-tok	46.10	22.80	76.40	48.20	81.90	39.40	69.50
SSR-CLS	46.60	23.50	76.80	**48.40**	82.80	40.00	**69.90**
BSI	**51.34**	**41.23**	**86.56**	45.34	**84.52**	**46.08**	65.25

Method	NF	NQ	QU	SD	SF	TO	Avg.
ColBERTv2	33.50	56.30	85.30	15.20	69.10	26.10	49.92
SPLADE-v2	33.50	52.10	83.80	15.90	69.40	36.30	50.61
SPLADE-v3	35.80	58.60	81.40	15.80	71.00	29.60	51.75
SSR-tok	38.80	60.70	88.30	17.50	72.90	33.10	53.51
SSR-CLS	**39.10**	61.20	88.70	17.90	73.30	33.70	53.99
BSI	35.20	**68.82**	**90.47**	**21.50**	**74.87**	**38.01**	**57.63**

(b) BSI with an 8B encoder and large embedding baselines

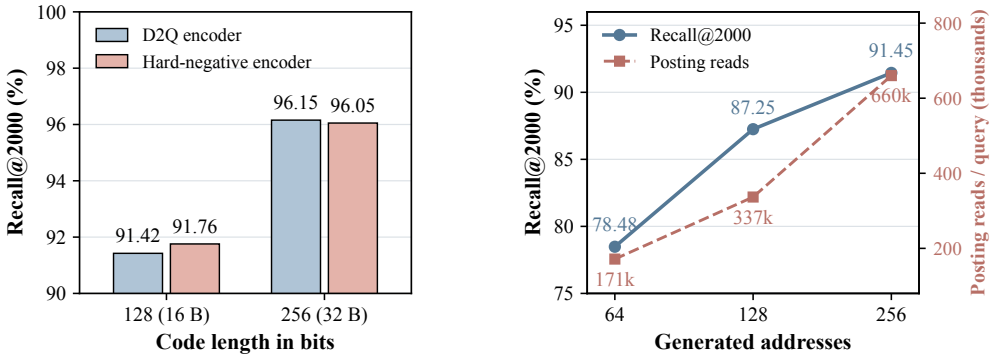
Method	AR	CF	CV	DB	FE	FQ	HQ
Llama-Nemotron	75.70	44.90	89.20	40.10	93.50	61.40	76.70
Qwen3	76.90	46.00	91.30	38.80	92.70	**64.60**	77.00
SSR-tok	75.40	45.50	89.60	41.50	93.90	62.10	77.20
SSR-CLS	76.20	**46.40**	90.10	43.30	**94.60**	62.80	77.90
BSI	**83.23**	43.65	**93.96**	**52.95**	94.44	62.83	**84.94**

Method	NF	NQ	QU	SD	SF	TO	Avg.
Llama-Nemotron	45.10	66.20	88.10	28.20	82.00	68.80	66.15
Qwen3	41.30	64.60	87.80	29.70	78.40	**73.00**	66.32
SSR-tok	46.00	67.70	87.50	28.10	81.80	69.40	66.59
SSR-CLS	**46.80**	68.40	88.60	29.50	**82.90**	70.50	67.54
BSI	43.83	**76.33**	**90.78**	**29.82**	79.87	45.32	**67.84**

4.4 Representation quality and compactness (**RQ3**)

Quality under a binary constraint. Table 9 in Appendix F shows competitive quality at a fixed code length. At 256 bits on NQ, BSQ reaches 84.36 Recall@10, close to 84.38 for residual quantization, and 66.48 MRR@10 versus 65.11. Optimized product quantization remains stronger in this static comparison, at 87.45 Recall@10. BSQ additionally requires binary queries and documents, whereas the other quantizers score reconstructed documents against continuous queries. This supports **BSQ as an indexable semantic representation** that balances retrieval quality with predictable, addressable structure.

One document, one compact code. Each document is represented by a 256-bit code, or 32 bytes. This is 64 times smaller than a 1,024-dimensional FP16 vector in representation payload. Its token combinations supply the index keys directly, so the same code supports similarity, lookup, and generation. SSR is a **purely sparse retrieval method** whose index derives from learned token-level features, with an additional global feature in SSR-CLS (Guo et al., 2026). BSI instead compresses document semantics into **one code per document**, reducing the amount of information required for index access.



(a) Representation budget **(b) Index access budget**

Figure 3: Representation and query-access budgets on MS MARCO. (a) Code length with each encoder fixed. (b) Addresses per pair with the local generator fixed.

Table 15 in Appendix H shows that BSI's codes, document identifiers, and posting index occupy **1.98 GiB for 8.84 million MS MARCO passages**. SSR reports an 18.5 GB persistent sparse index (Guo et al., 2026, Table 4), a substantially larger reported footprint. These separate measurements concern index storage, not latency or total memory: complete systems also require scoring representations and models. For **RQ3**, BSI combines useful semantic quality with a compact access structure; online efficiency requires matched runtime measurements.

4.5 Representation and access budgets (**RQ4**)

Binary representation capacity. Figure 3(a) isolates code length with the encoder held fixed, using 25,197 calibration queries. Increasing the code from 128 to 256 bits raises Recall@2000 from 91.42 to 96.15 for the query-augmented encoder and from 91.76 to 96.05 for the hard-negative encoder. The gains under both training approaches require sixteen additional bytes of code payload per document.

Query effort in a fixed space. Figure 3(b) varies the local-address budget on all 6,980 development queries while holding the generator fixed. Increasing addresses per pair from 64 to 128 raises Recall@2000 from 78.48 to 87.25; increasing to 256 raises it to 91.45. Posting reads grow from 0.17 to 0.34 and 0.66 million per query. The first doubling gains 8.77 recall points and the second gains 4.20, exposing a diminishing return in coverage. The budget controls search effort without rebuilding the document representation.

Table 11 in Appendix F shows a similar pattern for other query interfaces: wider complete-code beams improve coverage, while direct-query anchors show diminishing returns as search expands. For **RQ4**, **code length and address budget offer distinct controls** over representation capacity and query effort, allowing different balances between compactness and coverage.

5 Conclusion

We introduced **Binary Semantic Indexing**, a shared discrete foundation for **semantic modeling, sparse indexing, and generative querying**. Joint encoder and BSQ training produces compact document codes whose local structure supports complete-code generation, local-address generation, and direct query encoding. Experiments establish the usefulness of these interfaces and characterize the tradeoffs between representation quality, compactness, and access effort. A natural next step is to learn query mechanisms that adapt to new tasks and collection structure while preserving the document index, extending the separation between knowledge organization and query modeling. A shared index makes this distinction testable: query models can be compared over fixed document codes, separating changes in access quality from changes in the representation itself.

AI USE STATEMENT

Large language models (LLMs) were used to aid and polish the writing and to assist in proving mathematical claims. Writing assistance included improving clarity, grammar, and readability, refining technical explanations, and reducing redundant expressions. Mathematical assistance included suggesting proof strategies, developing intermediate derivations, and checking the consistency of assumptions, notation, and logical steps. The authors are responsible for reviewing and verifying the resulting arguments and for revising all LLM-assisted text before inclusion. The authors retain final judgment over the formal claims and their justification and take full responsibility for the manuscript.

ETHICS STATEMENT

This work evaluates information retrieval methods using publicly available pretrained models and established benchmarks, including Natural Questions, MS MARCO, BEIR, and the evaluated MAIR tasks. All evaluations use existing document collections and relevance annotations. The study involves no recruitment of human participants, human-subject interventions, or animal experiments, and no private user data was collected for this research. The datasets and models remain subject to their respective licenses and terms of use. The intended purpose of this work is to study semantic representations and efficient access to document collections.

REPRODUCIBILITY STATEMENT

The experiments use publicly accessible pretrained models from the Qwen family and established retrieval benchmarks. Section 3 defines the BSI framework and training objectives. Appendices B–D specify the binary representation, encoder and quantizer training, query-decoder objectives, and index-access procedures. Appendix E documents dataset subsets, evaluation metrics, common model settings, training schedules, retrieval budgets, and model-selection conventions. Appendices F–H provide supplementary comparisons, sensitivity analyses, and resource measurements. Together, these descriptions specify the construction and evaluation of BSI and the conditions under which the reported results are obtained.

REFERENCES

Arian Askari, Chuan Meng, Mohammad Aliannejadi, Zhaochun Ren, Evangelos Kanoulas, and Suzan Verberne. Generative retrieval with few-shot indexing. In *Advances in Information Retrieval: 48th European Conference on Information Retrieval*, pp. 606–614, 2026. doi: 10.1007/978-3-032-21300-6_52. URL https://arxiv.org/abs/2408.02152.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated MAchine Reading COmprehension dataset. arXiv:1611.09268, 2016. URL https://arxiv.org/abs/1611.09268.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. Autoregressive search engines: Generating substrings as document identifiers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 31668–31683, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/cd88d62a2063fdaf7ce6f9068fb15dcd-Paper-Conference.pdf.

Yufei Chen, Zihan Wang, Yubao Tang, Yukun Zhao, Maarten de Rijke, and Zhaochun Ren. Closing the indexing-decoding gap in multimodal generative retrieval via prefix retention optimization. *arXiv preprint arXiv:2606.09241*, 2026. URL https://arxiv.org/abs/2606.09241.

Zhuyun Dai and Jamie Callan. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020. doi: 10.1145/3397271.3401204. URL https://www.cs.cmu.edu/~callan/Papers/sigir20-Zhuyun_Dai.pdf.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.00904.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086*, 2021a. URL https://arxiv.org/abs/2109.10086.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021b. URL https://arxiv.org/abs/2107.05720.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. From distillation to hard negative sampling: Making sparse neural IR models more effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022. URL https://arxiv.org/abs/2205.04733.

Thibault Formal, Maxime Louis, Hervé Déjean, and Stéphane Clinchant. Learning retrieval models with sparse autoencoders. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f276154a603f183464a1b856fd9e3344-Abstract-Conference.html.

Kairui Fu, Tao Zhang, Shuwen Xiao, Ziyang Wang, Xinming Zhang, Chenchi Zhang, Yuliang Yan, Junjun Zheng, Xiangheng Kong, Shengyu Zhang, Kun Kuang, and Yuning Jiang. FORGE: Forming semantic identifiers for generative retrieval in industrial datasets. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, 2026. doi: 10.1145/3770855.3817477. URL https://arxiv.org/abs/2509.20904.

Luyu Gao, Zhuyun Dai, and Jamie Callan. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3030–3042, 2021. doi: 10.18653/v1/2021.naacl-main.241. URL https://aclanthology.org/2021.naacl-main.241/.

Lixuan Guo, Yifei Wang, Tiansheng Wen, Aosong Feng, Stefanie Jegelka, and Chenyu You. No more k-means: Single-stage sparse coding for efficient multi-vector retrieval. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.30120.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57. URL https://inria.hal.science/inria-00514462.

Shujie Ji, Yawei Kong, Yilin Zhao, Li Wang, Xialong Liu, and Peng Jiang. UniGD: A unified generative-discriminative framework for industrial retrieval. *arXiv preprint arXiv:2608.03150*, 2026. URL https://arxiv.org/abs/2608.03150.

Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, Suhang Wang, Jiawei Han, and Xianfeng Tang. Language models as semantic indexers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22244–22259. PMLR, 2024. URL https://proceedings.mlr.press/v235/jin24h.html.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020. URL https://arxiv.org/abs/2004.12832.

Kirill Khrylchenko. Variable-length semantic IDs for recommender systems. *arXiv preprint arXiv:2602.16375*, 2026. URL https://arxiv.org/abs/2602.16375.

Jaeyoung Kim, Dohyeon Lee, Soona Hong, and Seung-won Hwang. D3: Dynamic docid decoding for multi-intent generative retrieval. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 789–800, 2026. doi: 10.18653/v1/2026.eacl-industry.58. URL https://aclanthology.org/2026.eacl-industry.58/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. SPLADE-v3: New baselines for SPLADE. *arXiv preprint arXiv:2403.06789*, 2024. URL https://arxiv.org/abs/2403.06789.

Sangam Lee, Ryang Heo, SeongKu Kang, Susik Yoon, Jinyoung Yeo, and Dongha Lee. Why these documents? explainable generative retrieval with hierarchical category paths. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21807–21824, 2026. URL https://aclanthology.org/2026.findings-acl.1097/.

Sunkyung Lee, Minjin Choi, and Jongwuk Lee. GLEN: Generative retrieval via lexical index learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7693–7704, 2023. URL https://aclanthology.org/2023.emnlp-main.477/.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Multiview identifiers enhanced generative retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6636–6648, 2023. doi: 10.18653/v1/2023.acl-long.366. URL https://aclanthology.org/2023.acl-long.366/.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8716–8723, 2024a. doi: 10.1609/aaai.v38i8.28717. URL https://ojs.aaai.org/index.php/AAAI/article/view/28717.

Yongqi Li, Zhen Zhang, Wenjie Wang, Liqiang Nie, Wenjie Li, and Tat-Seng Chua. Distillation enhanced generative retrieval. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11119–11129, 2024b. URL https://aclanthology.org/2024.findings-acl.662/.

Jimmy Lin and Xueguang Ma. A few brief notes on DeepImpact, COIL, and a conceptual framework for information retrieval techniques. *arXiv preprint arXiv:2106.14807*, 2021. URL https://arxiv.org/abs/2106.14807.

Penghao Lu, Xin Dong, Yuansheng Zhou, Lei Cheng, Chuan Yuan, and Linjian Mo. DOGR: Leveraging document-oriented contrastive learning in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24732–24740, 2025. doi: 10.1609/aaai.v39i23.34654. URL https://ojs.aaai.org/index.php/AAAI/article/view/34654.

Antonio Mallia, Omar Khattab, Nicola Tonellotto, and Torsten Suel. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021. URL https://arxiv.org/abs/2104.12016.

Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. DSI++: Updating transformer memory with new

documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8198–8213, 2023. doi: 10.18653/v1/2023.emnlp-main.510. URL https://aclanthology.org/2023.emnlp-main.510/.

Kidist Amde Mekonnen, Yubao Tang, and Maarten de Rijke. Lightweight and direct document relevance optimization for generative information retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1327–1338, 2025. doi: 10.1145/3726302.3730023. URL https://arxiv.org/abs/2504.05181.

Kidist Amde Mekonnen, Yongkang Li, Yubao Tang, Simon Lupart, and Maarten de Rijke. Lost in decoding? reproducing and stress-testing the look-ahead prior in generative retrieval. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2994–3005, 2026. doi: 10.1145/3805712.3808567. URL https://arxiv.org/abs/2604.23396.

Thong Nguyen and Andrew Yates. Generative retrieval as dense retrieval. In *Workshop on Generative Information Retrieval at SIGIR*, 2023. URL https://arxiv.org/abs/2306.11397.

Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast exact search in hamming space with multi-index hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6): 1107–1119, 2014. doi: 10.1109/TPAMI.2013.231. URL https://www.cs.toronto.edu/~fleet/research/Papers/MIH-pami2014.pdf.

Smail Oussaidene and Mohand Boughanem. PDMR: Passage-driven multi-id document retrieval. *arXiv preprint arXiv:2609.08762*, 2026. URL https://arxiv.org/abs/2609.08762.

Gustavo Penha, Edoardo D'Amico, Marco De Nadai, Enrico Palumbo, Alexandre Tamborrino, Ali Vardasbi, Max Lefarov, Shawn Lin, Timothy Heath, Francesco Fabbri, and Hugues Bouchard. Semantic IDs for joint generative search and recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, 2025. doi: 10.1145/3705328.3759300. URL https://arxiv.org/abs/2508.10478.

Ronak Pradeep, Kai Hui, Jai Gupta, Adam D. Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Q. Tran. How does generative retrieval scale to millions of passages? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/abs/2305.11841.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://www.staff.city.ac.uk/~sbrp622/papers/foundations_bm25_review.pdf.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. PLAID: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pp. 1747–1756, 2022a. doi: 10.1145/3511808.3557325. URL https://arxiv.org/abs/2205.09707.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, 2022b. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272/.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. Learning to tokenize for generative retrieval. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2304.04171.

Weiwei Sun, Zhengliang Shi, Wu Jiu Long, Lingyong Yan, Xinyu Ma, Yiding Liu, Min Cao, Dawei Yin, and Zhaochun Ren. MAIR: A massive benchmark for evaluating instructed retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14044–14067, 2024. doi: 10.18653/v1/2024.emnlp-main.778. URL https://aclanthology.org/2024.emnlp-main.778/.

Weiwei Sun, Keyi Kong, Xinyu Ma, Shuaiqiang Wang, Dawei Yin, Maarten de Rijke, Zhaochun Ren, and Yiming Yang. ZeroGR: A generalizable and scalable framework for zero-shot generative retrieval. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8f7dc9855bb4ae8ae4f2d649f3f4efe6-Abstract-Conference.html.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. Generative retrieval meets multi-graded relevance. In *Advances in Neural Information Processing Systems*, 2024a. URL https://arxiv.org/abs/2409.18409.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. Bootstrapped pre-training with dynamic identifier prediction for generative retrieval. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10303–10317, 2024b. URL https://aclanthology.org/2024.findings-acl.614/.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2202.06991.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2104.08663.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. A neural corpus indexer for document retrieval. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper_files/paper/2022/file/a46156bd3579c3b268108ea6aca71d13-Paper-Conference.pdf.

Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. NOVO: Learnable and interpretable document identifiers for model-based IR. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 2656–2665, 2023. doi: 10.1145/3583780.3614993. URL https://www.zhouyujia.cn/attaches/CIKM2023_learndocid.pdf.

Leah Woldemariam, Sudhanshu Garg, Taha Belkhouja, Charles Kim-Yip, and Ali Sahami. Exploring bottom-up clustering for creating semantic IDs. *arXiv preprint arXiv:2609.08310*, 2026. URL https://arxiv.org/abs/2609.08310.

Shiguang Wu, Zhaochun Ren, Xin Xin, Jiyuan Yang, Mengqi Zhang, Zhumin Chen, Maarten de Rijke, and Pengjie Ren. Constrained auto-regressive decoding constrains generative retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025. doi: 10.1145/3726302.3729934. URL https://arxiv.org/abs/2504.09935.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zeFrfgyZln.

Zhichao Xu, Aosong Feng, Yijun Tian, Haibo Ding, and Lin Lee Cheong. CSPLADE: Learned sparse retrieval with causal language models. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 99–114, 2025. doi: 10.18653/v1/2025.ijcnlp-long.7. URL https://aclanthology.org/2025.ijcnlp-long.7/.

Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. Efficient passage retrieval with hashing for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

Processing (Volume 2: Short Papers), pp. 979–986, 2021. doi: 10.18653/v1/2021.acl-short.123. URL https://aclanthology.org/2021.acl-short.123/.

Peiwen Yuan, Xinglin Wang, Shaoxiong Feng, Boyuan Pan, Yiwei Li, Heda Wang, Xupeng Miao, and Kan Li. Generative dense retrieval: Memory can be a burden. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2835–2845, 2024. doi: 10.18653/v1/2024.eacl-long.173. URL https://aclanthology.org/2024.eacl-long.173/.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018. doi: 10.1145/3269206.3271800. URL https://ciir-publications.cs.umass.edu/getpdf.php?id=1302.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. Scalable and effective generative information retrieval. In *Proceedings of the ACM Web Conference 2024*, 2024a. URL https://arxiv.org/abs/2311.09134.

Hansi Zeng, Chen Luo, and Hamed Zamani. Planning ahead in generative retrieval: Guiding autoregressive generation through simultaneous decoding. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024b. doi: 10.1145/3626772.3657746. URL https://arxiv.org/abs/2404.14600.

Hailin Zhang, Yujing Wang, Qi Chen, Ruiheng Chang, Ting Zhang, Ziming Miao, Yingyan Hou, Yang Ding, Xupeng Miao, Haonan Wang, Bochen Pang, Yuefeng Zhan, Hao Sun, Weiwei Deng, Qi Zhang, Fan Yang, Xing Xie, Mao Yang, and Bin Cui. Model-enhanced vector index. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ac112e8ffc4e5b9ece32070440a8ca43-Abstract-Conference.html.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. Generative retrieval via term set generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. doi: 10.1145/3626772.3657797. URL https://www.zhouyujia.cn/attaches/SIGIR2024_TSGen.pdf.

Ruohan Zhang, Jiacheng Li, Julian McAuley, and Yupeng Hou. Purely semantic indexing for LLM-based generative recommendation and retrieval. *arXiv preprint arXiv:2509.16446*, 2025a. URL https://arxiv.org/abs/2509.16446.

Wenhao Zhang, Yuhan Wang, Yi Bai, Mengqi Zhang, Zhaochun Ren, Zhumin Chen, and Pengjie Ren. Conversational generative retrieval with contextual denoising. *Information Processing & Management*, 63(4):104642, 2026. doi: 10.1016/j.ipm.2026.104642. URL https://www.sciencedirect.com/science/article/pii/S0306457326000348.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b. URL https://arxiv.org/abs/2506.05176.

Xinpeng Zhao, Zhaochun Ren, Yukun Zhao, Zhenyang Li, Mengqi Zhang, Jun Feng, Ran Chen, Ying Zhou, Zhumin Chen, Shuaiqiang Wang, Dawei Yin, and Xin Xin. DiffuGR: Generative document retrieval with diffusion language models. *arXiv preprint arXiv:2511.08150*, 2025a. URL https://arxiv.org/abs/2511.08150.

Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e25198b6a75f74277ee3a2bd4165d9ef-Abstract-Conference.html.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257*, 2022. URL https://arxiv.org/abs/2208.09257.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*, 2022. doi: 10.48550/arXiv.2206.10128. URL https://arxiv.org/abs/2206.10128.

Yuxuan Zong, Mathias Vast, Basile Van Cooten, Laure Soulier, and Benjamin Piwowarski. From tokens to concepts: Leveraging SAE for SPLADE. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2026. URL https://arxiv.org/abs/2604.21511.

Appendix Overview

Appendix A: Extended related work. Generative document representations, sparse semantic indexing, and the connections between continuous and discrete retrieval spaces.

Appendix B: Binary representation and geometry. The geometry of BSQ, lossless token packing, and the soft targets used to train query decoders.

Appendix C: Learning the representation and query decoders. Encoder training, BSQ initialization, joint optimization, and the objectives for complete-code and local-address generation.

Appendix D: Retrieval through the generative interfaces. How model predictions query the index and how complete document codes support retrieval from the accessed postings.

Appendix E: Experimental settings. Datasets, metric definitions, common model settings, and the evaluation scope of the main comparisons.

Appendix F: Supplementary experiments. Query interfaces, quantization, model scale, generator learning, and sensitivity to retrieval and decoding budgets.

Appendix G: Additional transfer results. Further BEIR baselines and a matched comparison of binary and dense retrieval across six MAIR tasks.

Appendix H: Resource analysis. Representation and index storage, together with the measured computation used by representative training and retrieval stages.

Appendix I: Scope and limitations. The shared-index setting, representation and access tradeoffs, and extensions to evolving collections and adaptive query interfaces.

A Extended related work

BSI brings together questions studied in generative retrieval, sparse indexing, and representation learning: what a document representation should preserve, which of its components should be searchable, and how a query model should access them. We review these questions in turn, with particular attention to the relationship between a shared document space and the mechanisms used to query it.

A.1 Generative retrieval and document representations

Generative retrieval must choose prediction targets that connect a query to documents. Early work established several possibilities: entity names in GENRE, document identifiers in DSI, and semantically organized identifiers in NCI (De Cao et al., 2021; Tay et al., 2022; Wang et al., 2022). Ultron further examines how the semantic content of an identifier affects retrieval (Zhou et al., 2022). The resulting question is representational as well as predictive: which document properties should a short discrete target retain?

Learning the identifiers allows their structure to reflect the corpus and retrieval task. GenRet, GLEN, NOVO, and LMIndexer investigate this through document reconstruction, lexical index learning, and retrieval supervision (Sun et al., 2023; Lee et al., 2023; Wang et al., 2023; Jin et al., 2024). Other work examines the information that identifiers expose: Purely Semantic Indexing addresses collisions without arbitrary disambiguation tokens, HyPE uses hierarchical category paths, and PDMR

represents documents through multiple passage-derived identifiers (Zhang et al., 2025a; Lee et al., 2026; Oussaidene & Boughanem, 2026). Together, these approaches treat discrete identifiers as structured representations of document content.

Recent studies examine the construction and allocation of semantic IDs more directly. FORGE benchmarks identifier construction choices and proposes quality metrics that do not require training a downstream generative retriever (Fu et al., 2026). Bottom-up clustering builds unique identifiers while preserving local structure in the original embedding space (Woldemariam et al., 2026). Variable-length semantic IDs learn adaptive code lengths with a discrete variational autoencoder, allocating shorter descriptions to frequent items (Khrylchenko, 2026). These studies examine discriminability, locality, and code capacity. BSI additionally requires predictable code components that serve as index addresses.

Sharing also arises across retrieval tasks. Penha et al. (2025) study semantic IDs for joint search and recommendation, constructing a common identifier space from embeddings trained on both tasks. Conversational retrieval introduces another consideration: CGR4CD uses proposition-based document identifiers and context denoising to accommodate queries whose meaning depends on interaction history (Zhang et al., 2026). Sharing representations across tasks and supporting different query contexts complement BSI's focus on access mechanisms. In BSI, direct query encoding, complete-code generation, and local-address generation all operate over the same document codes and associated index. The contribution therefore concerns the interaction between document representation, query prediction, and index execution.

That structure must also support relevance. Query generation helps align indexing and retrieval inputs in DSI-QG (Zhuang et al., 2022). Relevance-oriented identifiers, iterative identifier learning, and graded-relevance objectives further connect the representation to what queries seek (Zeng et al., 2024a; Tang et al., 2024b;a). Learning-to-rank and distillation methods supply ranking evidence (Li et al., 2024a;b), while DOGR and DDRO develop document-oriented representation learning and document-preference optimization (Lu et al., 2025; Mekonnen et al., 2025). This line motivates learning BSI's binary space from retrieval supervision before training a decoder to access it.

A generated target can also identify evidence that an external index resolves into documents. SEAL makes this connection through generated substrings and an FM-index; MINDER uses multiple textual views; TSGen generates term sets; and PAG combines lexical and semantic identifiers (Bevilacqua et al., 2022; Li et al., 2023; Zhang et al., 2024; Zeng et al., 2024b). These methods establish a useful separation between the model's output and the search structure that executes it. BSI develops this separation through a shared binary representation: a decoder can predict complete codes or local addresses, while an inverted index retrieves documents and their complete codes support candidate selection.

The usefulness of a document space also depends on how it is maintained and searched. Scaling studies examine model capacity and training data, while DSI++, few-shot indexing, and ZeroGR address changing collections and limited supervision (Pradeep et al., 2023; Mehta et al., 2023; Askari et al., 2026; Sun et al., 2026). Constrained generation, diffusion-based retrieval, query-adapted identifiers, prefix retention, and look-ahead decoding study different ways to navigate that space (Wu et al., 2025; Zhao et al., 2025a; Kim et al., 2026; Chen et al., 2026; Mekonnen et al., 2026). These concerns complement our focus on the interface between predicted semantic units and an explicit document index.

A.2 Sparse retrieval and semantic indexing

Sparse retrieval links a representation to an execution strategy: active query features select posting lists, which restrict the documents that must be examined. Classical term-based models such as BM25 make this connection through lexical overlap (Robertson & Zaragoza, 2009). Learned lexical methods preserve the indexing structure while improving its semantic evidence. DeepCT and DeepImpact learn term importance and impacts; uniCOIL assigns contextual scalar weights; and COIL stores contextualized token vectors in inverted lists (Dai & Callan, 2020; Mallia et al., 2021; Lin & Ma, 2021; Gao et al., 2021). The common design problem is how to make indexable features capture more than surface-form agreement.

Vocabulary expansion offers one route. SPLADE learns sparse weights over the vocabulary, including terms absent from a document's text (Formal et al., 2021b). Its subsequent versions improve the training data, distillation, and negative sampling used to learn these representations (Formal et al., 2021a; 2022; Lassance et al., 2024); CSPLADE extends the approach to causal language models (Xu et al., 2025). This family shows how learned semantic matching can retain direct access through inverted lists. Its indexing coordinates remain tied to a lexical vocabulary.

Learning the coordinates themselves allows the index to organize latent semantic features. SNRM demonstrates sparse latent dimensions for inverted indexing (Zamani et al., 2018). Recent sparse-autoencoder approaches develop this direction through learned indexing features in SPLARE, concept representations in SAE-SPLADE, and sparse coding of token embeddings for multi-vector retrieval in SSR (Formal et al., 2026; Zong et al., 2026; Guo et al., 2026). BSI shares their interest in learned semantic units, with a different representation structure: local combinations of a compact binary code define index keys. Sparsity arises from the keys a document activates in this address space; the complete code retains the geometry used to compare documents. This connects sparse execution to the units a generative query model predicts.

A.3 Continuous and discrete representations for retrieval

Continuous representations make semantic similarity available as a geometric relation. DPR and ANCE learn query and document vectors for this purpose (Karpukhin et al., 2020; Xiong et al., 2021). Multi-vector methods retain finer-grained matching evidence: ColBERT operates over token representations, while ColBERTv2 and PLAID address their compression and efficient search (Khattab & Zaharia, 2020; Santhanam et al., 2022b;a). Foundation-model-based embeddings and relevance models further broaden the tasks and languages supported by learned semantic representations (Zhang et al., 2025b). Across these designs, the representation must balance the semantic information it retains against the cost of searching it.

Quantization makes this tradeoff explicit by replacing continuous vectors with compact discrete codes. Product quantization encodes vector subspaces, while binary passage retrieval learns compact binary representations for text retrieval (Jégou et al., 2011; Yamada et al., 2021). Multi-index hashing uses code partitions to support search in Hamming space (Norouzi et al., 2014). BSQ supplies an implicit binary codebook on the unit sphere, with an exact relationship between inner products and Hamming distance (Zhao et al., 2025b). BSI uses this geometry to learn a retrieval representation whose complete codes support comparison and whose local components support indexing. Its predicted-address lookup and token-marginal scoring define a retrieval procedure distinct from exact Hamming-radius search.

Connections between continuous and generative retrieval suggest a broader role for such discrete spaces. Nguyen & Yates (2023) interpret forms of generative retrieval through dense query–document scoring. MEVI links generated semantic cluster identifiers to vector search through residual quantization, and GDR combines generative cluster selection with dense retrieval inside the selected clusters (Zhang et al., 2023; Yuan et al., 2024). UniGD integrates generative retrieval and relevance scoring using representations anchored to semantic codebooks (Ji et al., 2026). BSI studies the shared document representation underlying these kinds of connections. Retrieval-trained binary codes provide its semantic space; complete-code and local-address decoders supply alternative ways to query the associated index. Subsequent scoring can then operate within the candidates reached through that generative access.

B Binary representation and geometry

The binary representation has to support semantic comparison, discrete prediction, and indexed access. Compact codes have long supported efficient nearest-neighbor search through vector quantization (Jégou et al., 2011), and learned binary codes have also been used to compress passage representations for retrieval (Yamada et al., 2021). We first describe its geometry, then explain how the same code yields both hard document tokens and soft training targets. All constructions use the representation learned jointly by the encoder and BSQ. Figure 4 illustrates nearest-code assignment and the resulting Hamming geometry.

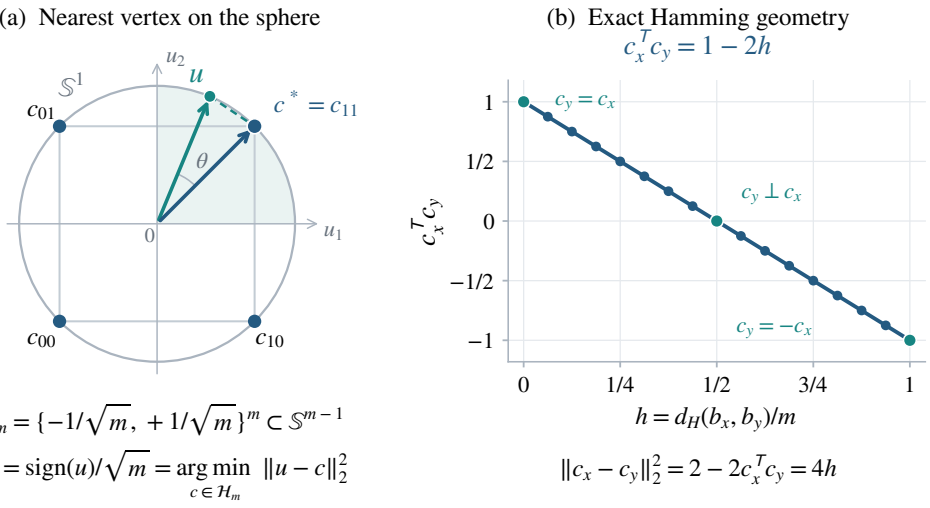


Figure 4: Binary spherical geometry: nearest-vertex assignment and the exact relation between Hamming distance and code similarity.

A SPHERICAL SPACE WITH CLOSED-FORM ASSIGNMENT

Let $\mathcal{I}$ be the coordinate set, with $|\mathcal{I}| = m$, and let $\mathcal{S}$ denote the negative and positive unit signs. Following BSQ (Zhao et al., 2025b), the implicit codebook consists of the normalized vertices of the centered binary hypercube:

$$\mathcal{H}_m = \{\text{normalize}(s) : s \in \mathcal{S}^m\}, \qquad c(u) = \text{normalize}(\text{sign}(u)).$$

Every code has unit Euclidean norm. For a unit projection u, maximizing its inner product with a code separates over coordinates, so choosing each coordinate's sign gives a nearest code:

$$c(u) \in \arg\max_{c \in \mathcal{H}_m} u^\top c = \arg\min_{c \in \mathcal{H}_m} \|u - c\|, \qquad \max_{c \in \mathcal{H}_m} u^\top c = \frac{\sum_{i \in \mathcal{I}} |u_i|}{\sqrt{m}}.$$

We choose the positive sign at a vanishing coordinate. Assignment requires no search over an explicit table of centroids. The nearest-code statement concerns the projected sphere; retrieval supervision is still needed to make that sphere useful for relevance matching.

The code geometry also gives an exact relation between code similarities. For codes c_x, c_y, let $\mathcal{A}_{xy}$ and $\mathcal{D}_{xy}$ be the sets of agreeing and disagreeing coordinates. Then

$$c_x^\top c_y = \frac{|\mathcal{A}_{xy}| - |\mathcal{D}_{xy}|}{m}, \qquad d_H(c_x, c_y) = |\mathcal{D}_{xy}|.$$

Since the coordinate sets partition $\mathcal{I}$, Hamming distance determines the inner product exactly. This correspondence also underlies Hamming-based candidate retrieval with learned binary passage codes (Yamada et al., 2021). Unit norms also make squared Euclidean distance an affine function of that inner product. These identities survive token packing, although equality of a local token pair does not specify a Hamming-radius neighborhood.

LOSSLESS TOKENS AND SOFT TARGETS

Partition $\mathcal{I}$ into ordered groups $\mathcal{I}_j$ of width w, indexed by $j \in \mathcal{J}$. A fixed bijection $\text{pack} : \mathcal{S}^w \to \mathcal{V}$ maps each sign group to a token:

$$t_{d,j} = \text{pack}\big((\text{sign}(u_{d,i}))_{i \in \mathcal{I}_j}\big), \qquad V = |\mathcal{V}| = |\mathcal{S}|^w, \qquad m = |\mathcal{J}|w.$$

The inverse map recovers every sign. Thus a token sequence and a binary code carry the same information; tokenization changes the units available to the generator and index. Using code substrings as lookup keys has a precedent in multi-index hashing (Norouzi et al., 2014). Here, the groups define

prediction and access units; the retrieval procedure does not inherit the exact Hamming-neighbor guarantees of that algorithm.

For learning, we use the soft assignment associated with BSQ (Zhao et al., 2025b). Its Gibbs distribution over spherical codes has a normalizer that factorizes over coordinates:

$$P_\beta(c \mid u) = \frac{\exp(\beta u^\top c)}{Z_\beta(u)}, \qquad Z_\beta(u) = \prod_{i \in \mathcal{I}} \left[\exp(\beta u_i/\sqrt{m}) + \exp(-\beta u_i/\sqrt{m})\right].$$

Consequently, the soft sign probabilities and the probability of a token are

$$p_i^{\pm}(u) = \frac{\exp(\pm\beta u_i/\sqrt{m})}{\exp(\beta u_i/\sqrt{m}) + \exp(-\beta u_i/\sqrt{m})}, \qquad r_j(v \mid x) = \prod_{i \in \mathcal{I}_j} p_i^{\sigma_i(v)}(u_x).$$

Here $\sigma_i(v)$ is the sign decoded from token v, and $\beta = m/T_{\mathrm{BSQ}}$ for quantizer temperature T_{BSQ}. These probabilities can be evaluated without enumerating the codebook. Factorization is a property of the conditional distribution; it does not require empirical bit frequencies across the corpus to be independent.

The full-code decoder distills $r_j(\cdot \mid d^+)$ from a positive document. The local decoder's query teacher instead supplies $r_j(\cdot \mid q)$. This distinction matters: the former describes uncertainty about a target document's code, while the latter describes the frozen representation's response to a query. Both teachers use the same learned binary space.

C Learning the representation and query decoders

Training establishes a document space before learning how queries should access it. We first optimize the shared encoder, introduce quantization with the encoder frozen, and then adapt both components jointly. The resulting document codes are fixed for all subsequent generator training. The complete-code decoder learns a sequence distribution; the local decoder first learns token-level query evidence and then learns address dependence and index-aware decisions.

Establishing a retrieval-aware document space

The shared encoder applies a LayerNorm–linear readout to the final non-padding transformer state and normalizes the result. Its contrastive objective treats all known relevant entries in the global batch as positives. For query entries $\mathcal{Q}_\mathcal{B}$, document entries $\mathcal{D}_\mathcal{B}$, and the relevant query set $\mathcal{Q}_\mathcal{B}^+(d)$, the reverse direction of the main-text loss is

$$\mathcal{L}_{d\to q}(r) = -\mathbb{E}_d \log \frac{\sum_{q \in \mathcal{Q}_\mathcal{B}^+(d)} \exp(r_q^\top r_d/\tau)}{\sum_{q \in \mathcal{Q}_\mathcal{B}} \exp(r_q^\top r_d/\tau)}, \qquad \mathcal{L}_{\mathrm{ret}}(r) = \mathrm{mean}\{\mathcal{L}_{q\to d}(r), \mathcal{L}_{d\to q}(r)\}.$$

Representations are gathered across workers with gradients preserved. The losses operate on batch entries, retaining positive mass for repeated or otherwise relevant entries. The encoder stage minimizes $\mathcal{L}_{\mathrm{ret}}(e)$.

BSQ adds a projection and a synchronized running center before spherical normalization. The center is updated from detached projected features; gradient descent updates the projection. In the frozen-encoder stage, quantized retrieval loss aligns hard codes through the straight-through estimator. After that initialization, the entire encoder is unfrozen, and dense and quantized retrieval losses are optimized together. This final joint checkpoint supplies both the document codes and the soft teachers used by the generators.

The regularizers shape how the code capacity is used. Let $p_{x,i}$ be the soft sign distribution from Appendix B, and H its entropy. The information term is

$$\mathcal{L}_{\mathrm{ent}} = \mathbb{E}_{i \in \mathcal{I}} \left[\mathbb{E}_x H(p_{x,i}) - H(\mathbb{E}_x p_{x,i})\right].$$

Minimization favors confident assignments for individual examples while preserving diversity across examples. For decorrelation, form soft signs $Y_{x,i} = p_{x,i}^+ - p_{x,i}^-$, standardize each coordinate across examples, and penalize off-diagonal correlations:

$$Z_{x,i} = \frac{Y_{x,i} - \mathbb{E}_x Y_{x,i}}{\sqrt{\mathrm{Var}_x(Y_{x,i}) + \epsilon}}, \qquad R_{ij} = \mathbb{E}_x[Z_{x,i} Z_{x,j}], \qquad \mathcal{L}_{\mathrm{corr}} = \mathbb{E}_{i \neq j}[R_{ij} R_{ij}].$$

These statistics use the concatenated query and document examples on each worker. The complete objectives are $\mathcal{L}_{\mathrm{BSQ}}$ and $\mathcal{L}_{\mathrm{joint}}$ in the main text. The selected training recipe has no additional reconstruction or commitment objective.

Learning complete document codes

The complete-code generator conditions on the query and a teacher-forced positive prefix. At each position, its output distribution is normalized over that position's code vocabulary $\mathcal{V}$. Both causal cross-entropy and soft-target distillation use this same distribution. A positive document is selected for each training query, cycling through its relevant documents across epochs; the loss averages over query instances and token positions.

The hard target is $t_{d^+,j}$, and its soft counterpart is $r_j(\cdot \mid d^+)$ from the frozen document projection. The soft teacher is independent of the generated prefix, whereas the student's prediction conditions on the gold prefix. The objective $\mathcal{L}_{\mathrm{full}}$ therefore combines exact sequence supervision with the geometry retained in the teacher's spherical margins. All generator parameters are updated, while the document encoder, quantizer, and target table remain fixed. The selected continuation uses these likelihood and distillation terms without local-access or boundary-retention losses.

Preparing the local decoder's query model

Before fitting address dependence, we train a query model to predict token distributions $\pi_{\eta,j}(\cdot \mid q)$. This stage gives the later local head query states and base logits already aligned with the binary space. Query-BSQ targets provide distillation, and positive document tokens provide relevance supervision:

$$\mathcal{L}_{\mathrm{tokenKD}} = \mathbb{E}_{q,j} D_{\mathrm{KL}}(r_j(\cdot \mid q) \,\|\, \pi_{\eta,j}(\cdot \mid q)), \qquad \mathcal{L}_{\mathrm{tokenNLL}} = -\mathbb{E}_{q,d^+,j} \log \pi_{\eta,j}(t_{d^+,j} \mid q).$$

A frozen dense teacher additionally provides a small training candidate set $\mathcal{T}(q)$ and a distribution over its documents:

$$w_q(d) = \frac{\exp(e_q^\top e_d / T_t)}{\sum_{d' \in \mathcal{T}(q)} \exp(e_q^\top e_{d'} / T_t)}.$$

For this query-model stage, define a student's document score as the mean token log probability, $A_\eta(q,d) = \mathbb{E}_j \log \pi_{\eta,j}(t_{d,j} \mid q)$. A softmax over $\mathcal{T}(q)$ gives a student distribution ν_η, which is trained with $D_{\mathrm{KL}}(w_q \| \nu_\eta)$.

An anchor loss further encourages a positive token to outrank strong alternatives. If $\mathcal{V}^-_{q,d^+,j}$ contains the highest-scoring competing values, excluding the positive token, set

$$A^-_{q,d^+,j} = \log \sum_{v \in \mathcal{V}^-_{q,d^+,j}} \exp(\ell_{\eta,j,v}) - \log |\mathcal{V}^-_{q,d^+,j}|,$$

$$\mathcal{L}_{\mathrm{anchor}} = \mathbb{E}_{q,d^+,j} \operatorname{softplus}(\delta + A^-_{q,d^+,j} - \ell_{\eta,j,t_{d^+,j}}).$$

Here ℓ_η are the pre-softmax logits and δ is the desired margin. The query-model objective combines token distillation, positive-token likelihood, candidate-distribution distillation, and this anchor term with scalar weights. It updates the query backbone; after training, we freeze that backbone for the local-head stages.

Learning local address distributions

The local head adds trainable residual logits to the frozen query model's base predictions and learns a query-independent table $C_g[a,b]$ for each position pair. It uses the conditional factorization in the main text, with no additional query-dependent interaction tensor. The residuals adapt query evidence, while the table learns which token values form useful addresses together.

Positive-address likelihood averages over all valid positive pairs and position groups in the batch:

$$\mathcal{L}_{\mathrm{addr}} = -\frac{\sum_{q \in \mathcal{Q}_\mathcal{B}} \sum_{d \in \mathcal{P}(q)} \sum_{g \in \mathcal{G}} \log p_{\psi,g}(a_g(d) \mid q)}{|\mathcal{G}| \sum_{q \in \mathcal{Q}_\mathcal{B}} |\mathcal{P}(q)|}.$$

The auxiliary supervision preserves query-teacher predictions, transfers document relevance, and discourages expensive addresses. Write $\pi_g^f = \mathrm{softmax}(f_g(q))$ and $\pi_g^h = \mathrm{softmax}(h_g(q))$. Query distillation acts on these base distributions:

$$\mathcal{L}_{\mathrm{queryKD}} = \mathbb{E}_{q,g}\,\mathrm{mean}\left\{D_{\mathrm{KL}}(r_{i_g}(\cdot \mid q)\|\pi_g^f), D_{\mathrm{KL}}(r_{j_g}(\cdot \mid q)\|\pi_g^h)\right\}.$$

In particular, its second-token target supervises h_g before the conditional table is added. The joint distribution and its marginals are used by the address and retrieval objectives.

The dense teacher's candidate distribution w_q supervises local addresses through weighted likelihood:

$$\mathcal{L}_{\mathrm{native}} = -\mathbb{E}_{q,g} \sum_{d \in \mathcal{T}(q)} w_q(d) \log p_{\psi,g}(a_g(d) \mid q).$$

Unlike the earlier query-model loss, this term does not normalize student scores over the candidate documents. For posting cost, let $L_g(a) = |I_g(a)|$, let $L_g^{\max}$ be the largest posting length, and define $\chi(z) = \mathrm{softplus}(\log z)$ for positive z. Then

$$c_g(d) = \frac{\chi(L_g(a_g(d)))}{\chi(L_g^{\max})}, \qquad \widehat{w}_{q,g}(d) = \frac{p_{\psi,g}(a_g(d) \mid q)}{\sum_{d' \in \mathcal{T}(q)} p_{\psi,g}(a_g(d') \mid q)},$$

$$\mathcal{L}_{\mathrm{cost}} = \mathbb{E}_{q,g} \sum_{d \in \mathcal{T}(q)} \widehat{w}_{q,g}(d) c_g(d).$$

The cost is logarithmically normalized and evaluated over the teacher's sampled document entries. Repeated addresses in that sample retain their multiplicity. These terms give

$$\mathcal{L}_{\mathrm{aux}} = \lambda_q \mathcal{L}_{\mathrm{queryKD}} + \lambda_t \mathcal{L}_{\mathrm{native}} + \lambda_p \mathcal{L}_{\mathrm{cost}}.$$

The initial local head learns from $\mathcal{L}_{\mathrm{addr}}$ and $\mathcal{L}_{\mathrm{aux}}$. Its subsequent continuation adds the access and retention terms below.

Aligning predictions with index decisions

A relevant address must cross a probe boundary before its document can be considered. Let $\tau_g(q)$ be the current log probability of the B_{train}-th highest-probability nonempty address. The threshold is recomputed but detached from differentiation. For a positive document, define

$$\Delta_g(q, d^+) = \log p_{\psi,g}(a_g(d^+) \mid q) - \mathrm{sg}(\tau_g(q)),$$

$$M_{\mathrm{access}}(q, d^+) = T_a \left[\log \sum_{g \in \mathcal{G}} \exp(\Delta_g / T_a) - \log |\mathcal{G}|\right].$$

This normalized log-sum-exp is a conservative surrogate for crossing a probe boundary. Let $\Delta_* = \max_{g \in \mathcal{G}} \Delta_g$. For positive T_a,

$$\Delta_* - T_a \log |\mathcal{G}| \;\le\; M_{\mathrm{access}}(q, d^+) \;\le\; \Delta_*.$$

To obtain the bounds, factor $\exp(\Delta_*/T_a)$ out of the sum: the remaining sum lies between unity and $|\mathcal{G}|$. Thus the surrogate approaches the hard maximum as the temperature vanishes. At finite temperature, a positive M_{access} implies that a positive address clears a probe boundary, but clearing a boundary in only one group need not make M_{access} positive. Equality at a boundary is subject to tie breaking.

The subtraction of $T_a \log |\mathcal{G}|$ removes the positive group-count offset of an unnormalized log-sum-exp when all margins lie at the boundary. At fixed T_a and $\mathcal{G}$, this shift preserves the margin's derivatives with respect to the group margins. It nevertheless changes the value and gradient scale of the following softplus loss:

$$\mathcal{L}_{\mathrm{access}} = \mathbb{E}_q \mathbb{E}_{d^+ \in \mathcal{P}(q)} T_a\, \mathrm{softplus}(-M_{\mathrm{access}}(q, d^+)/T_a).$$

The normalization is a modeling choice rather than an exact differentiable encoding of the hard any-group access event; the reported experiments do not isolate its effect against alternative access surrogates.

Candidate retention addresses the next decision: keeping a reached document when code scores truncate the union. A stored training retrieval snapshot supplies active stages $\mathcal{S}_q$, position masks $J_{q,s}$, and boundary competitors $\mathcal{N}_{q,s}$. The stages correspond to the prefilter and complete-code selection. Using the current smoothed score from Appendix D, define

$$M_{\mathrm{keep}}(q, d^+, s) = S_{J_{q,s}}(q, d^+) - \max_{d^- \in \mathcal{N}_{q,s}} S_{J_{q,s}}(q, d^-),$$

$$\mathcal{L}_{\mathrm{retain}} = \mathbb{E}_q \mathbb{E}_{s \in \mathcal{S}_q} \mathbb{E}_{d^+ \in \mathcal{P}(q)} \operatorname{softplus}(-M_{\mathrm{keep}}).$$

Queries without a valid stored stage contribute no retention loss. Competitor identities and position masks are fixed, but their scores and the positive scores are recomputed, with gradients through both. Codes identical to any positive code are excluded as competitors. Positives missed by a stored retrieval run may still supervise training; evaluation never inserts them into the candidate union.

The final local objective is the weighted sum of positive-address likelihood, auxiliary supervision, access, and retention given in the main text. Only the local head is updated. These losses optimize smooth score margins around discrete decisions; they do not differentiate through posting lookup or hard candidate selection.

D Retrieval through the generative interfaces

Retrieval converts decoder predictions into document candidates through a fixed semantic index. The procedure has a common structure across interfaces: construct addresses, gather their postings, and compare complete codes within the resulting union. This section describes those operations in execution order.

A fixed index over shared document codes

Each document contributes its identifier to $I_g(a_g(d))$ for every position pair g, and a separate table stores its complete token sequence. Pairing is fixed before query-model continuations. Local generation uses adjacent positions; complete-code generation uses a learned perfect matching over the same position set. Its offline selection considers prediction quality, coverage, posting selectivity, and code stability. The document codes are shared even though the pairing can differ.

Lookup preserves every document attached to a selected key. When a document occurs in several accessed postings, union construction deduplicates its identifier. Code collisions likewise retain the colliding documents rather than assigning the code to an arbitrary representative.

Turning generated predictions into lookups

For complete-code generation, let $\mathcal{R}(q)$ be the retained completed beam and ℓ_r its stored log score. Its normalized beam weights, token marginals, and address sets are

$$\omega_r = \frac{\exp(\ell_r)}{\sum_{r' \in \mathcal{R}(q)} \exp(\ell_{r'})}, \qquad p_j(v \mid q) = \sum_{r \in \mathcal{R}(q)} \omega_r \mathbb{I}\{t_j^{(r)} = v\},$$

$$A_g(q) = \{(t_{i_g}^{(r)}, t_{j_g}^{(r)}) : r \in \mathcal{R}(q)\}.$$

Only pairs observed together in a completed sequence become lookup keys. Marginals are used for scoring, not to create additional keys by a Cartesian product. The weights normalize scores within the retained beam; they are not an exact posterior over every possible code.

For local generation, let $\mathcal{A}_g^\circ$ be the nonempty addresses in the index. The model directly selects

$$A_g(q) = \mathrm{Top}_B\{\log p_{\psi,g}(a \mid q) : a \in \mathcal{A}_g^\circ\},$$

$$p_{i_g}(a \mid q) = \sum_{b \in \mathcal{V}} p_{\psi,g}(a, b \mid q), \qquad p_{j_g}(b \mid q) = \sum_{a \in \mathcal{V}} p_{\psi,g}(a, b \mid q).$$

If fewer nonempty addresses are available, all are retained. Under the conditional factorization, the first-token marginal equals $\mathrm{softmax}(f_g(q))$, whereas the second-token marginal depends on the conditional table C_g. The table therefore affects the selected keys and the second-token evidence used to compare candidates.

SELECTING CANDIDATES FROM THE POSTING UNION

The common union $U(q)$ contains precisely the documents reached through these keys. For a token distribution p_j, let α and $\bar{\alpha}$ be complementary mixture weights, with α assigned to the uniform distribution. For any position subset J, define

$$\widetilde{p}_j(v \mid q) = \bar{\alpha} p_j(v \mid q) + \alpha/V, \qquad S_J(q,d) = \sum_{j \in J} \log \widetilde{p}_j(t_{d,j} \mid q).$$

Smoothing bounds the penalty for an unlikely token. It is applied after token probabilities are constructed, so even a factorized BSQ teacher need not induce a binary inner-product score after smoothing.

Although the local marginals incorporate conditional-table predictions, summing their log probabilities discards explicit within-pair association. This score is therefore distinct from a sum of joint address log probabilities. We use marginal scoring to provide a common complete-code ranking interface for both query routes after their different address proposals; this does not imply that it is an optimal ranking rule. Table 8 compares scoring rules on a fixed candidate pool from complete-code generation on NQ. That comparison concerns scoring within the fixed pool, not an ablation of the final local-address decoder.

A coarse prefilter first selects informative positions. Let $v_j(d) = \log \widetilde{p}_j(t_{d,j} \mid q)$ and let $n_j(q)$ count its distinct values over $U(q)$. The position utility combines confidence, score variation, and value diversity:

$$\rho_j(q) = \frac{\log V - H(p_j)}{\log V} \sqrt{\mathrm{Var}_{d \in U(q)}\, v_j(d)}\, \frac{\chi(n_j(q))}{\chi(V)}, \qquad \chi(z) = \mathrm{softplus}(\log z).$$

The r highest-utility positions form $J_r(q)$, with deterministic position-based tie breaking. Their partial score reduces the union before the complete-code score is applied:

$$\widehat{U}(q) = \mathrm{Top}_{K_{\mathrm{pre}}}\{S_{J_r(q)}(q,d) : d \in U(q)\}, \qquad \mathcal{C}_K(q) = \mathrm{Top}_K\{S_{\mathcal{J}}(q,d) : d \in \widehat{U}(q)\}.$$

Smaller sets are retained in full, and an empty union returns no candidates. Computing position utility inspects code values across the union; r describes the partial scoring step, not all memory accesses.

WORK AND SUBSEQUENT RANKING

Address count and candidate count describe different aspects of execution. The posting work is

$$W(q) = \sum_{g \in \mathcal{G}} \sum_{a \in A_g(q)} |I_g(a)|.$$

It counts entries read across posting lists, including repeated accesses to a document before deduplication. Different address budgets can therefore produce different work even when $|\mathcal{C}_K(q)|$ is capped at the same value.

Within the generative interface, dense similarity and a text scorer can order the documents in $\mathcal{C}_K(q)$. The index-only experiments preserve this candidate set. The complete-system evaluation additionally considers complementary retrieval, as summarized in Appendix E.

E EXPERIMENTAL SETTINGS

This section defines the benchmarks, metrics, and common settings used to evaluate BSI's representation quality, query interfaces, transfer, and resource use.

E.1 DATASETS AND EVALUATION

NQ uses an NQ320K-derived collection of 109,650 documents, with 15,324 calibration queries and a separate 7,830-query holdout. MS MARCO Passage contains 8,841,823 passages; our grouped development evaluation has 6,980 queries. BEIR includes all thirteen datasets shown in Table 3. The additional evaluation on MAIR (Massive Instructed Retrieval Benchmark) (Sun et al., 2024) covers six tasks and 419 queries.

BEIR dataset abbreviations. AR: ArguAna; CF: Climate-FEVER; CV: TREC-COVID; DB: DBPedia; FE: FEVER; FQ: FiQA; HQ: HotpotQA; NF: NFCorpus; NQ: Natural Questions; QU: Quora; SD: SCIDOCS; SF: SciFact; TO: Touché.

Unless marked as document recall, Recall@K is the fraction of queries with a relevant document among the top K results:

$$\text{Recall@}K = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{I}\{\mathcal{R}_K(q) \cap \mathcal{P}(q) \neq \varnothing\}.$$

Document Recall@K instead averages $|\mathcal{R}_K(q) \cap \mathcal{P}(q)|/|\mathcal{P}(q)|$. The two coincide in our single-positive NQ setting. MRR@10 uses the reciprocal rank of the first relevant result, with zero assigned when none appears in the first ten. BEIR and MAIR use linear-gain nDCG@10, normalized by the ideal ordering for each query, and macro-average equally across datasets. All reported scores are percentages. The code-length comparison uses 25,197 MS calibration queries, and the address-budget study uses all 6,980 development queries.

Scope of the main results. The NQ main score uses calibration queries, and the MS MARCO score uses development queries; these judgments also inform model selection. Published NQ results use their original test splits, so the table provides context rather than a matched test-set ranking. BEIR reports the best available configuration per dataset, selected using target judgments. Its model-size labels refer to the encoder; both sizes can use external text models up to 8B. These conventions apply throughout the experiments.

Statistical scope. The principal comparisons report observed scores without repeated-seed estimates or confidence intervals. Close results, including BSQ–RQ and BSI–Dense, are numerical observations rather than tests of equivalence or non-inferiority.

E.2 Common representation and query models

The NQ encoder and generators are initialized from Qwen3-0.6B. The encoder uses the last non-padding state, a LayerNorm–linear readout, and normalization to produce a 1,024-dimensional representation. BSQ produces 256 bits, packed losslessly into 32 tokens with 256 values per position. Training first learns the encoder, then the quantizer with the encoder frozen, and finally both components jointly. Document codes remain fixed during generator training.

The representation is trained on 742,946 pairs with train-derived document-to-query augmentation. Generator training uses 246,220 queries. Tables 4 and 5 summarize the training schedule. The complete-code decoder combines sequence likelihood with soft BSQ targets. The local decoder learns query-token evidence and address distributions, followed by the index-aware continuation described in Appendix C.

Table 4: NQ representation training on 742,946 pairs. Batch sizes are global.

Stage	Epochs	Updates	Batch	Learning rate
Encoder	8	1,448	4,096	Backbone 3×10^{-5}; readout 10^{-3}
Frozen-encoder BSQ	8	360	16,384	BSQ 10^{-3}
Joint encoder–BSQ	1	181	4,096	Encoder 3×10^{-6}; BSQ 10^{-4}

The contrastive temperature is 0.02. Quantized relevance, entropy, and decorrelation losses have weights 1, 0.05, and 0.05; joint training adds continuous relevance supervision with weight 1. Complete-code distillation uses weight 0.1. Local query training combines teacher distillation, positive-token likelihood, document-level distillation, and an anchor objective. Local-head training adds address likelihood, posting cost, and access and retention objectives. Their roles and formulas are given in the method supplement.

Table 5: NQ decoder training on 246,220 queries. Batch sizes are global.

Stage	Updates	Batch	Learning rate	Exposures
Query model (local branch)	7,700	128	10^{-5}	985,600
Local address head	2,000	256	10^{-4}	512,000
Local access/retention	1,374	256	10^{-5}	351,744
Complete-code continuation	1,924	128	10^{-5}	246,272

The query-model counts combine 3,850 initial and 3,850 continuation updates. Local-head stages use that frozen query model; retention continues the initial head with 30% replay from a 100,000-query snapshot. Complete-code training starts from a separate selected Soft-SID checkpoint, and its exposures include padding. These counts exclude earlier training of the source checkpoints.

E.3 Index access and complete-system evaluation

Both generative interfaces query position-aware token-pair indexes and retain up to 2,000 documents. The common settings are summarized in Table 6. MS MARCO uses the same 256-bit construction, with up to 512 local addresses per pair; direct access predicts index keys from the encoded query. The transfer experiments use MS-trained representations, with Qwen3-0.6B or an 8B encoder as indicated by the table panel.

Table 6: NQ inference budgets for the two generative interfaces.

Setting	Local address	Complete code
Query input limit	64 tokens	
Position pairs	16 adjacent	16 learned
Beam width	—	2,048
Lookup keys per pair	256 nonempty	$\leq$2,048 observed
Shared candidate selection		
Smoothing α	0.01	
Prefilter positions	8 of 32	
Prefilter document cap	70,000	
Complete-code candidate cap	2,000	
Candidate-restricted ranking		
Native shortlist / text depth	100	200
Score-fusion weights	1 : 3 : 1	1 : 3

Complete-code keys are distinct pairs observed in completed beams. Fusion weights apply to standardized scores: native:NQ-adapted:public for local generation, and native:public for complete-code generation.

For the complete-system scores, retrieved documents are ordered with embedding and text relevance scores from the Qwen model family (Zhang et al., 2025b). NQ combines an adapted 8B text model with a public 4B model; its best system additionally admits at most two documents from complementary dense retrieval into the final results. MS MARCO's BSI result uses binary-index candidates. BEIR selects among MS-trained and public relevance models, with task-appropriate prompts and score combinations. The main text uses the common name BSI; the interface analysis below separates the binary-index results from the complete NQ system.

Published references. The NQ table retains 25 methods from the existing comparison, spanning sparse, dense, and generative retrieval. The main reporting sources are GenRet, TSGen, DOGR, and the recent Few-Shot GR, HyPE, and PDMR studies (Sun et al., 2023; Zhang et al., 2024; Lu et al., 2025; Askari et al., 2026; Lee et al., 2026; Oussaidene & Boughanem, 2026). MS MARCO uses full Passage references, including LTRGR, DGR, RIPOR, and PAG (Li et al., 2024a;b; Zeng et al., 2024a;b). Smaller MS collections are not mixed into that table. BEIR baselines follow the

published SSR comparison (Guo et al., 2026). The main comparison and its additional baselines use all thirteen datasets; the scoring study in Table 13 uses twelve, excluding Touché.

F SUPPLEMENTARY EXPERIMENTS

The supplementary studies examine the retrieval information retained by binary codes, how query interfaces access it, and the training and search choices that affect both.

F.1 QUERY INTERFACES AND HELD-OUT RETRIEVAL

Table 7: Query-interface results (%) on 15,324 NQ calibration and 6,980 MS development queries. R denotes query-level Recall. R@2000 measures binary-index coverage; complete NQ systems also admit complementary retrieval. Dashes denote unreported or inapplicable results.

NQ interface	R@2000	R@10
Complete code	95.41	92.40
Local address	**96.47**	92.95
Complete system		
Complete code	—	94.15
Local address	—	**94.28**

MS interface	R@2000	R@10	MRR@10
Local	93.21	69.86	41.98
Global	—	71.67	41.23
Direct BSQ	**97.21**	**72.95**	**43.36**

Table 7 shows that both NQ generators reach high coverage through the binary index. Local generation achieves 96.47 Recall@2000, compared with 95.41 for complete-code generation. Restricting final results to index-generated documents gives 92.95 and 92.40 Recall@10, respectively; the complete NQ systems reach 94.28 and 94.15. On MS MARCO, direct query encoding reaches 97.21 Recall@2000, 72.95 Recall@10, and 43.36 MRR@10. Local generation reaches 41.98 MRR@10, while Global generation reaches 71.67 Recall@10 and 41.23 MRR@10. These results show that the same binary document space remains useful across direct and generative query interfaces.

On the independent NQ holdout, the local interface reaches 96.17 Recall@2000 and 90.22 Recall@10 with embedding-based ordering. Exact complete-code decoding at beam size 100 reaches 81.58 Recall@10 and 84.79 Recall@100. The best complete-system NQ evaluation remains the calibration result reported in the main text.

Using evidence across the document code. Table 8 changes only the ordering of the complete-code generator's fixed 2,000 candidates. Token marginals yield 82.21 Recall@10 and 90.06 Recall@100, compared with 76.50 and 87.24 for address probabilities; pair-joint scoring reaches 81.73 Recall@10. With Recall@2000 fixed at 95.41, this supports using code-wide probability evidence for ranking within the candidate set. It does not isolate candidate retention or the final Local decoder.

Table 8: Ordering the same 2,000 candidates from complete-code generation on 15,324 NQ calibration queries (%). Candidate Recall@2000 is fixed at 95.4124 for every row. No embedding or text scoring is applied.

Score	Recall@10	Recall@100	MRR@10
Address probability	76.5009	87.2357	67.6560
Pair joint	81.7280	88.3973	69.6570
Token marginal	**82.2109**	**90.0613**	**69.7459**

F.2 QUALITY OF THE DISCRETE REPRESENTATION

BSQ retains competitive retrieval quality while providing directly indexable binary structure. Table 9 compares representations on 15,324 NQ calibration queries. At 256 bits, BSQ obtains 84.36 Recall@10, close to RQ's 84.38, and 66.48 MRR@10, compared with RQ's 65.11. OPQ retains an

Table 9: Static representation quality on NQ. R denotes Recall; bold marks the best result using unrounded values.

Representation	Bits/doc.	R@10	R@2000	MRR@10
FP32	32,768	**91.28**	98.88	76.66
FP16	16,384	91.27	**98.88**	**76.66**
BSQ128	128	75.23	96.44	55.41
PQ	256	79.06	98.39	59.99
OPQ	256	87.45	98.53	69.55
RQ	256	84.38	98.66	65.11
BSQ256	256	84.36	97.70	66.48

advantage in this nearest-neighbor evaluation, reaching 87.45 Recall@10 and 69.55 MRR@10. The comparison also involves different query representations: PQ, OPQ, and RQ score reconstructed document vectors against continuous queries, whereas BSQ operates on binary query and document codes. Under this binary constraint, BSQ preserves useful retrieval information and supplies the discrete units required for BSI's shared generative and indexed access. This is the basis for our choice of representation. The NQ 128- and 256-bit models differ in their training stages; Figure 3(a) isolates code length with a fixed encoder.

F.3 Model scale and generator learning

Table 10: NQ training controls: encoder scale at a fixed update budget (top) and generator objectives (bottom).

Encoder	R@10	R@2000	MRR@10	GPU-hours
0.6B	85.08	98.20	66.18	**2.35**
4B	89.92	98.89	71.92	6.32
8B	**90.17**	**98.96**	**72.80**	9.43

Generator objective	Updates	R@10	R@64	MRR@10
Parent checkpoint	0	80.91	83.30	69.28
Cross-entropy	5,000	80.66	83.14	69.28
Listwise	5,000	80.68	82.81	69.30
Cross-entropy + soft targets	5,000	**81.36**	**83.69**	**69.72**

The scale study in Table 10 holds training fixed at 512 updates and 524,288 pair exposures. Increasing the backbone from 0.6B to 4B improves NQ Recall@10 from 85.08 to 89.92, while 8B reaches 90.17. Most of the observed gain comes from the first increase in capacity. The table reports training time alongside retrieval quality to characterize this tradeoff.

Figure 5 examines the data used to learn document-code prediction. Diverse query views outperform repeated real queries in the first continuation, and the next multiview stage improves over replay of its parent data. Returning the mature model to a narrow query distribution reverses that benefit: continued multiview replay remains stronger. The result points to the importance of covering multiple ways to express a document's relevance.

The objective comparison in Table 10 evaluates 5,000-update continuations on a separate 6,144-query development set. Adding soft BSQ targets reaches 81.36 Recall@10, compared with 80.66 for cross-entropy alone and 80.68 for the listwise variant. The modest improvement supports using the representation's geometry to inform decoder supervision.

F.4 Decoding and query-access sensitivity

Table 11 studies how query effort affects access to a fixed document space. Increasing the complete-code beam from 16 to 256 raises exact-code recall from 79.27 to 88.05. For direct queries, increasing

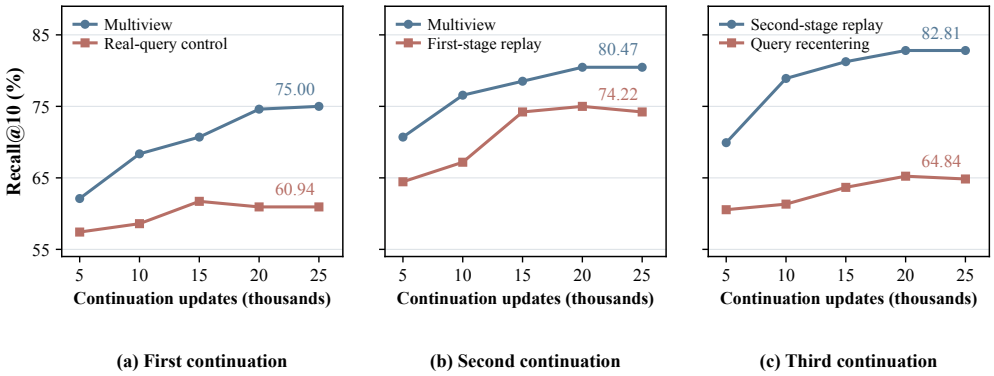


Figure 5: Generator learning on 256 NQ calibration queries. Each panel compares matched training durations within a curriculum stage; scores are exact-code Recall@10.

Table 11: NQ sensitivity to query-access and decoding budgets (%). Each block fixes its model: direct anchors use 1,024 calibration queries; the beam sweep uses 15,324 calibration queries; temperature uses the 7,830-query holdout and beam 100. Exact-code recall is measured at the indicated beam size.

Direct-query anchors	**1**	**2**	**4**	**8**	**16**
Recall@2000	83.20	94.73	98.63	99.02	**99.12**
Recall@10	80.27	89.26	91.50	**91.60**	**91.60**

Beam	**16**	**32**	**128**	**256**
Exact-code recall	79.27	81.79	86.00	**88.05**

Temperature	**1.00**	**1.15**
Recall@10	80.66	**80.95**
MRR@10	69.32	**69.47**

the number of token anchors from one to four raises Recall@2000 from 83.20 to 98.63; further expansion brings smaller gains. These trends agree with the local-address study in the main text: broader access helps, but its marginal return declines.

Temperature has a smaller effect in the mature generator. With the checkpoint and beam fixed, changing the temperature from 1 to 1.15 raises holdout Recall@10 from 80.66 to 80.95. The beam and address studies show that query effort controls coverage; this temperature comparison suggests a finer adjustment within a fixed decoding budget.

G Additional transfer results

Table 12 adds published baselines to the BSI comparison in Table 3 on the same thirteen datasets. The MAIR study then compares binary and dense access under shared scoring across different document types.

G.1 Shared and dataset-specific scoring on BEIR

Table 13 fixes the 8B encoder and binary-index candidates. Native embedding scores yield 64.7084 macro nDCG@10; shared text scoring raises it to 67.4252, and dataset-specific selection reaches 69.7178. Shared scoring improves eleven datasets, with a small decrease on NFCorpus. Uniform execution does not establish selection without target judgments. These means exclude Touché and are distinct from the BEIR-13 main comparison.

Table 12: Additional BEIR-13 baselines from Guo et al. (2026) (nDCG@10, %), complementing Table 3. Means use all thirteen datasets; bold marks the best score within each panel. Abbreviations are defined in Appendix E.

(a) Additional token and sparse retrieval baselines

Method	AR	CF	CV	DB	FE	FQ	HQ
ColBERT	23.60	18.10	67.90	39.20	76.90	31.70	59.20
COIL	29.50	**21.70**	66.80	39.90	**83.80**	31.30	**71.30**
PLAID	46.10	17.40	**73.70**	**44.70**	78.80	**36.30**	66.70
AligneR	28.90	18.20	68.80	41.60	72.40	33.50	61.50
CITADEL	**51.10**	18.30	71.50	42.20	76.60	33.00	66.30
XTR	40.70	20.80	73.60	40.80	73.70	34.80	64.70

Method	NF	NQ	QU	SD	SF	TO	Avg.
ColBERT	30.70	51.80	85.70	14.60	66.80	20.50	45.13
COIL	33.20	52.10	83.80	15.50	70.70	28.10	48.28
PLAID	33.80	**56.40**	85.00	15.40	69.20	26.30	49.98
AligneR	**34.00**	52.40	82.30	14.30	70.30	**34.80**	47.15
CITADEL	33.70	54.00	85.30	**15.90**	69.00	34.20	**50.08**
XTR	**34.00**	52.80	**85.90**	14.60	**71.10**	31.30	49.14

(b) Additional embedding baselines

Method	AR	CF	CV	DB	FE	FQ	HQ
SFR-Embed	67.20	28.30	**87.60**	27.30	86.60	60.40	73.80
Linq-Embed	**69.60**	30.30	87.10	27.80	87.60	61.20	**75.40**
E5-Mistral	61.70	28.50	87.00	34.60	87.00	56.80	73.20
GTE-Qwen2	54.60	**32.20**	80.40	**47.30**	**90.60**	**62.00**	75.00
BGE-large	64.50	27.30	74.70	41.90	87.30	45.00	75.00

Method	NF	NQ	QU	SD	SF	TO	Avg.
SFR-Embed	40.30	34.80	88.90	19.90	76.20	55.20	57.42
Linq-Embed	**41.10**	41.90	**89.20**	21.90	76.40	54.50	58.77
E5-Mistral	33.40	**66.40**	88.50	16.30	75.10	55.40	58.76
GTE-Qwen2	40.40	65.60	87.90	**23.50**	**79.50**	**59.20**	**61.40**
BGE-large	37.10	51.10	88.90	22.60	73.60	47.10	56.62

Table 13: BEIR-12 nDCG@10 (%) with a fixed 8B encoder and binary-index candidates. Native uses embedding scores; Uniform applies the same text-scoring configuration across datasets; Selected uses a configuration chosen for each dataset with target judgments. Touché is excluded from every mean in this table.

Scoring	AR	CF	CV	DB	FE	FQ
Native	63.05	35.64	86.66	49.53	91.73	59.92
Uniform	75.00	41.43	91.21	**52.94**	93.32	60.08
Selected	**83.23**	**43.65**	**93.96**	**52.94**	**94.44**	**62.83**

Scoring	HQ	NF	NQ	QU	SD	SF	Mean
Native	81.48	42.25	73.57	88.88	26.18	77.61	64.71
Uniform	83.83	41.26	75.04	89.62	26.82	78.56	67.43
Selected	**84.94**	**43.83**	**76.33**	**90.78**	**29.82**	**79.87**	**69.72**

Table 14: MAIR nDCG@10 (%) with the same encoder for BSI and Dense, before and after shared text scoring. Columns: AILA2019-Case, FinanceBench, News21, LitSearch, CodeSearchNet, and ClinicalTrials_2023 (50, 100, 32, 100, 100, and 37 queries). BM25 is an independent lexical reference. Bold marks the best score per column within each panel.

(a) Native embedding scores

Method	AILA	Finance	News	LitSearch	Code	Clinical	Mean
Dense	**5.33**	**21.08**	18.34	15.77	**42.00**	**12.12**	**19.11**
BSI (Direct)	5.30	20.38	**19.23**	**15.82**	41.98	11.64	19.06

(b) Shared text scoring and lexical reference

Method	AILA	Finance	News	LitSearch	Code	Clinical	Mean
BM25	**13.62**	17.37	17.65	**77.93**	50.95	**38.25**	35.96
Dense	5.97	61.21	27.05	46.12	**68.62**	24.47	38.91
BSI (Direct)	5.97	**62.79**	**27.16**	46.14	67.91	23.55	**38.92**

G.2 Preserving dense retrieval quality on MAIR

Binary and dense access give close scores before and after text scoring. With a shared 0.6B encoder, their native macro nDCG@10 scores are 19.0562 and 19.1077 (Table 14); shared 8B text scoring yields 38.9202 and 38.9078. BM25 averages 35.96 and remains strong on specialized tasks. These observations cover six MAIR subsets and 419 queries.

H Resource analysis

Index storage and query work describe distinct resource requirements.

Table 15: Measured logical storage for 8,841,823 MS MARCO passages, including metadata. Model weights and training caches are additional.

Stored component	GiB	Bytes / document
Code table with document IDs	0.329	40.00
Posting index	1.655	200.95
Combined access structure	**1.984**	**240.95**
FP32 embedding cache	134.982	16,392.11

Table 15 reports 1.98 GiB for codes, document IDs, and postings over 8.84 million MS MARCO passages. Complete-system memory additionally includes the embedding cache, model weights, and working memory; index compactness alone does not determine this total or end-to-end latency.

For the mature MS local model, increasing the address budget from 256 to 512 raises posting reads from 0.66 to 1.29 million per query and Recall@2000 from 91.59 to 93.21. Reads count repeated accesses before deduplication.

An MS local-candidate job uses 3.29 allocated GPU-hours for 6,980 queries. Such job-level measurements include startup where recorded and do not establish online latency or throughput under matched hardware.

I Scope and limitations

This study focuses on fixed document collections and separately trained query interfaces. Extending BSI to evolving collections and jointly optimized query mechanisms is a natural next step. Posting-list length and overlap could further guide query generation, allowing access effort to adapt to collection structure while preserving the shared document representation.