

On the Value of Behavioral Representations for Dense Retrieval

Nan Jiang*
Purdue University
IN, USA
jiang631@purdue.edu

Dhivya Eswaran*
Amazon Search
CA, USA
deswaran@amazon.com

Choon Hui Teo
Amazon Search
CA, USA
choonhui@amazon.com

Yexiang Xue
Purdue University
IN, USA
yexiang@purdue.edu

Yesh Dattatreya
Amazon Search
CA, USA
ydatta@amazon.com

Sujay Sanghavi
Amazon Search
CA, USA
sujayrs@amazon.com

Vishy Vishwanathan
Amazon Search
CA, USA
vishy@amazon.com

ABSTRACT

We consider text retrieval within dense representational space in real-world settings such as e-commerce search where (a) document popularity and (b) diversity of queries associated with a document have a skewed distribution. Most of contemporary dense retrieval literature—which typically focuses on MSMARCO and TREC benchmark datasets—present two shortcomings in these settings. (1) They learn an *almost equal number of representations per document*, agnostic to the fact that a few ‘head’ documents are disproportionately more critical to achieving a good retrieval performance. (ii) They learn *purely semantic document representations* inferred from intrinsic document characteristics (e.g. tokenized text) which may not contain adequate information to determine the queries for which the document is relevant—especially when the document is short.

We propose to overcome these limitations by *augmenting semantic document representations learned by bi-encoders with behavioral document representations* learned by our proposed approach MVG. To do so, MVG (1) determines how to divide the total budget for behavioral representations by drawing a connection to Pitman-Yor process, and (2) simply clusters the queries related to a given document (based on user behavior) within the representational space learned by a base bi-encoder, and treats the cluster centers as its behavioral representations. Our central contribution is the finding *such a simple intuitive light-weight approach leads to substantial gains in key first-stage retrieval metrics (e.g. recall) by incurring only a marginal memory overhead*. We establish this via extensive experiments over three large public datasets comparing to several single-vector (e.g. SentenceBERT) and multi-vector (e.g. ColBERT)

*Both authors contributed equally to the paper

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

xxxx 'xx, xxx 00 – 99, 0000, xx, xx

© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/22/04...\$15.00
<https://doi.org/10.1145/1122445.1122456>

bi-encoders, a proprietary e-commerce search dataset comparing to production-quality bi-encoder, and an A/B test. We hope that the next generation of dense retrieval approaches more carefully considers the dual questions of representational budget distribution, and of jointly learning semantic and behavioral representations.

ACM Reference Format:

Nan Jiang, Dhivya Eswaran, Choon Hui Teo, Yexiang Xue, Yesh Dattatreya, Sujay Sanghavi, and Vishy Vishwanathan. 2022. On the Value of Behavioral Representations for Dense Retrieval. In *xxx*. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/1122445.1122456>

1 INTRODUCTION

Dense retrieval (DR) is a powerful workhorse for large-scale language systems which rely on text retrieval as the first step, e.g. web search [24, 43], e-commerce shopping [31] and multi-label text categorization [29]. The main idea in DR is to embed queries and documents into a common continuous representation space, such that the relevance of a document to a query is captured by the proximity of their representations in this space. The representations are robust and fully learnable from data; thus DR greatly enhances **generalization** (e.g. robustness in the face of synonyms, misspellings, morphological variants) compared to sparse retrieval approaches which rely on lexical overlap, e.g. BM25 [37]. During serving, the document representations are pre-computed and stored within an efficient nearest-neighbor (NN) index for fast lookup [15].

Our motivating observation is that *in many real-world retrieval settings, document popularity follows a skewed (e.g. power-law) distribution [7]*. As shown in Figure 1 (left), most user queries result in clicks for only a small set of ‘head’ documents whereas there is a long ‘tail’ of documents with very few associated queries. Thus **memorization** of associations between these *head* documents and queries associated with them in the past (as well as generalizing these associations to new query variants) tends to be disproportionately more important to maintaining a good retrieval performance in real-world settings, compared to the retrieval of *tail* documents.

Typical DR approaches [11, 13, 28, 31, 35, 43] are ‘single-vector’ i.e. they learn a single representation per document, agnostic to whether the document is head or tail. However, *learning only a*

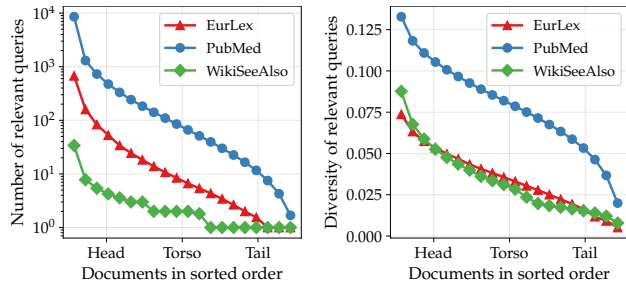


Figure 1: Document popularity distributions in a few common datasets: A few documents are seen to be disproportionately popular (left) and relevant to a diverse (right) set of queries. This motivates us to consider non-uniform number of extra (behavioral) representations for such documents within the nearest-neighbor (NN) index of dense retrieval.

single representation per head document is not adequate for memorization of its historical query associations. Figure 1 (right) compares the diversity of queries used to access head and tail documents, and shows that head documents are typically accessed through a disproportionately diverse set of queries compared to tail documents¹. However, such diverse and disparate reasons for accessing a head document cannot be shoehorned into a single representation: triangle inequality (a typical prerequisite for the representation space to be well-defined) disallows a document to be close to two queries, when the queries are themselves well-separated in this space.

Recent DR approaches [19, 24, 38, 39] consider learning multiple representations per document. Still, a prevailing limitation is that the learned document representations are purely semantic. That is, once the ‘behavioral’ graph of query-document associations is used to train the bi-encoder, the graph is discarded, and only the documents’ intrinsic features (typically, tokenized text) are used to infer their representations. This is especially problematic for short documents where the tokenized text does not contain adequate information to determine the queries for which it is relevant. Graph neural networks [8, 22, 44] is an attractive framework to better leverage graph information, but suffers from high online retrieval latency [21].

To address the above challenges, we propose to *augment semantic document representations learned by bi-encoder models with behavioral document representations learned by our proposed approach: MVG (Multi-View Geometric Index)*. MVG and its resulting behavioral representations have several attractive properties: **(a) Variable number of representations per document:** The number of behavioral representations learned per document can be controlled via a hyperparameter to be as skewed as is appropriate for a given dataset. **(b) Improved memorization for head documents:** The behavioral representations are learned directly from the behavioral graph (hence the name) by simply clustering the queries connected to the document within the representational space given by the bi-encoder. Thus, behavioral representations are not tied

¹Here, diversity of a set of queries is measured by the volume of the smallest bounding box enclosing all their representations in a given space (we use bert-base-uncased model from <https://huggingface.co>). Concretely, if l_j is the length of this box along dimension $j \in \{1, \dots, r\}$, we compute diversity as $(l_1 l_2 \dots l_r)^{1/r}$. But other notions of diversity (e.g. smallest bounding ball instead of box) led to a similar observation.

Table 1: MVG-based Dense Retrieval with steps 1-4 performed offline and step 5 conducted online. Only step 3 (focus of this paper) is new compared to standard dense retrieval pipeline.

Steps in MVG-based Dense Retrieval	New?
1. Learn a bi-encoder model \mathcal{M}	✗
2. Infer semantic document representations from \mathcal{M}	✗
3. Infer behavioral document representations from MVG	✓
4. Build an NN-index of all document representations	✗
5. Query the NN-index online	✗

to intrinsic document features, and are more flexible in memorizing query-document associations. **(c) Light-weight practical approach:** MVG can be implemented with a light-weight step (step 3) in the standard DR pipeline as shown in Table 1, with no change in bi-encoder training or inference procedure (steps 1-2), the NN-index building step (step 4), or the online retrieval logic (step 5). These properties, together, set MVG apart from past dense retrieval literature as we illustrate in Table 2 and detail in Section 2.

Concretely, our contributions are four-fold:

- **Theoretical Connection:** We leverage Pitman-Yor Process [34] (PYP) to explain the skewed distribution of query diversity from Figure 1. Under PYP, the expected number of query clusters m for a document with n queries scales as $m = \mathcal{O}(n^\beta)$ for some $\beta \in (0, 1)$.
- **Light-Weight Algorithm:** We propose the MVG algorithm to improve the recall of any bi-encoder by incurring a marginal overhead in NN-index size. The core algorithmic questions here are: (i) how many behavioral representations to learn per document, and (ii) how to learn them. MVG provides principled answers for both, by leveraging the connection to PYP for (i) and formulating (ii) as a constrained clustering problem on a unit sphere. MVG is also easy to deploy within any industrial infrastructure that supports bi-encoders since the only change is in the set of document representations to be indexed for NN-search (Table 1).
- **Experiments on Public Datasets:** We evaluate MVG as an approach for first-stage retrieval where recall and mean-average precision (MAP) are the key metrics. When applied on top of state-of-the-art (SOTA) single-vector approaches [31, 35] in three large diverse datasets, MVG consistently delivers 12 – 28% recall and 4 – 33% MAP gains (computed over $k = 100$ results) by incurring only 1.3× increase in index size. In comparison with the prominent SOTA ‘multi-vector’ ColBERT [19], MVG achieves gains of 3% – 15% recall and 1% – 17% MAP by requiring 4 – 6× smaller index space. All gains are in absolute percentages, and statistically significant [45].
- **Experiments within E-Commerce Search Engine:** MVG improves a production-quality bi-encoder by 13% recall and 4% MAP (at $k = 100$) in terms of offline metrics by increasing the index size by only 1.2% (a large fraction of tail documents were absent from the behavioral graph as is typical in an industrial setting; such documents were not allocated any behavioral representations by MVG). In an A/B test, MVG significantly increased customer engagement metrics without degrading customer-perceived latency.

2 RELATED WORK

We review prior dense retrieval (DR) literature, emphasizing the axes which set our work apart via Table 2. We categorize prior DR

methods into two based on whether they learn a single or multiple representation(s) per query and document.

Single-Vector Bi-encoders learn a single vector per query and document; such methods are widely used industrially, e.g. Youtube [5], Airbnb [11], Amazon [31]. Early approaches include DSSM [13] and DESM [28] which relied on simple text embedding approaches (e.g. FastText [16]) and worked well for short-text documents. Recently, the emphasis has shifted to transformer [6] based bi-encoders such as S.BERT [35] to better capture semantic similarity for long-text documents by considering the entire context. Several works try to improve negative mining for single-vector bi-encoders, e.g. ADORE [47], DPR [18] and most recently, ANCE [43] which relies on globally hard negatives. Our proposed MVG is complementary to any work in this category, and can be applied on top of any single-vector bi-encoder to improve retrieval performance.

Multi-Vector Bi-encoders are becoming increasingly common as a single vector may not suffice to adequately represent a document it [32, 42]. We further subdivide these works in two. **(1) Fixed count:** These methods learn m vectors for each document by either directly using the first m BERT token embeddings (ME-BERT [24], or after clustering all token embeddings into m centroids (PQR [39]), or by having m [CLS]-like tokens instead of one (MVR [48]). These methods typically employ a different relevance scoring function during inference than the one used in training. **(2) Variable count:** Here the number of representations learned per document may vary, e.g. based on document length. The most prominent approach here is ColBERT [19] which achieves state-of-the-art performance by preserving all (contextualized) token embeddings from queries and documents and using late-stage interaction in an end-to-end differentiable manner. Several works consider how to optimize memory usage of ColBERT while minimally impacting retrieval performance. MAIZE [38] uses a residual compression mechanism with a denoised supervision strategy; CQ [46] uses contextual quantization of token embeddings; COIL [9] leverages contextualized token representations stored in inverted lists; ColBERTer [12] removes document tokens which minimally impact final scores. Other methods are PRF [41] which uses pseudo-relevance feedback and MVA [49] which uses multiple vector attention mechanism. However, a common limitation here is that the learned document representations are all semantic, which is limited by what can be inferred from intrinsic document features (e.g. tokenized text)—especially for short-text documents. Moreover, the proposed MVG is complementary to these DR methods and can be used in combination to create behavioral representations to augment the learned semantic representations.

Exploiting graph structure: Our method uses query-document relevance graph to compute behavioral representations; this is broadly related to the relational learning [10] and graph neural networks [8, 22, 44] but they are typically expensive to serve online at web-scale. The closest work here is simultaneous work [21] which uses graph information within a bi-encoder model. But it does not consider variable number of representations per document.

Choice of Baselines: Given the rapid pace of DR research, and our focus on retrieval settings which are uncommon for DR, a core experimental challenge is select a set of representative baselines which (a) allows us to test all key hypotheses but also (b) is small in number so that we can retrain and tune performance on all three

public datasets that we consider. We shortlist two single-vector baselines: classic DSSM and transformer-based S.BERT which are better suited for short- and long- text applications respectively. We choose ColBERT as the competitive multi-vector baseline. Fortunately, we do not have to consider follow-up works of ColBERT which trade-off memory footprint for retrieval performance as we already observe superior performance w.r.t ColBERT in Table 5.

3 PRELIMINARIES

We provide a brief background on dense retrieval and use this to formally state the problem we tackle in the rest of the paper.

3.1 Dense Retrieval

Consider historical relevance data (Q, \mathcal{D}, w) over a set of queries Q and documents \mathcal{D} . For $q \in Q, d \in \mathcal{D}$, the relevance score (e.g. based on past clicks) is noted as $w_{qd} \geq 0$. Let \mathcal{M} be a bi-encoder model with query encoder \mathcal{M}_Q and document encoder \mathcal{M}_D , where the parameters for the two encoders can be fully or partially shared.

Single-Vector Bi-encoders. Here, we obtain a single representation per query or document. Denote these as $\mathbf{x}_q := \mathcal{M}_Q(q)$ and $\mathbf{x}_d := \mathcal{M}_D(d)$. Typically, $\|\mathbf{x}_q\|_2 = \|\mathbf{x}_d\|_2 = 1$, i.e. unit vectors in some r -dimensional space, and their relevance score is computed via their dot product or cosine similarity:

$$\text{rel}(q, d) := \mathbf{x}_q^\top \mathbf{x}_d \quad (1)$$

As the query and document vectors can be inferred independently, bi-encoders allow us to precompute document vectors and index them in an efficient data-structure, e.g. HNSW [25] so that time complexity of nearest-neighbor search to retrieve relevant documents grows only logarithmically in the size of the document corpus.

Multi-Vector Bi-encoders. Here, we obtain one or more representations per query or document. Denote them as $\{\mathbf{x}_{qi}\}_{i \in [1, m_q]} = \mathcal{M}_Q(q)$ and $\{\mathbf{x}_{di}\}_{i \in [1, m_d]} = \mathcal{M}_D(d)$. Many scenarios are possible as shown in Table 2, e.g. multiple vectors for documents only or for queries only or for both; count of multiple vectors fixed or variable across documents and queries. The most common formulation of multi-vector relevance score [12, 19, 24, 46], based on **max-sim** operator over document vectors for a given query vector, is:

$$\text{rel}(q, d) := \sum_{i \in [1, m_q]} \max_{j \in [1, m_d]} \mathbf{x}_{qi}^\top \mathbf{x}_{dj} \quad (2)$$

where again $\|\mathbf{x}_{qi}\|_2 = \|\mathbf{x}_{dj}\|_2 = 1 \forall q, i, d, j$. Typically, queries have short text; thus setting $m_q = 1 \forall q$ helps achieve the best online latency. In this case, retrieval according to Equation (2) is simply the following: build an NN-index of all representations for all documents and conduct NN-search identical to single-vector case.

3.2 Problem Formulation

Our goal is to augment semantic document representations learned by a given bi-encoder with behavioral document representations so as to improve the memorization performance of DR approaches. For ease of exposition in the rest of the paper, we assume that bi-encoder under consideration is a single-vector approach; but our formulation and method generalizes to multi-vector bi-encoders in a straightforward manner. Overall, our problem can be stated as:

Table 2: MVG vs prior DR approaches on (a) size of nearest-neighbor (NN) index (e.g. HNSW [25]), (b) latency or time required for NN search (for the same embedding dimensionality). Let D be the number of documents, t_d (t_q) be average number of tokens per document (query). Let m be a fixed hyper-parameter, and δ be the relative increase in number of document representations due to MVG. For conservative values such as $D = 1M$, $t_q = 8$, $t_d = 128$, $m = 4$, $\delta = 0.3$, MVG achieves 7 – 90% less latency while requiring 3 – 12× smaller index size than other representative multi-vector approaches, e.g. ColBERT, ME-BERT. Thus, MVG-based dense retrieval is faster and requires lower memory; as we later show in Section 6, MVG also achieves better retrieval performance.

Methods	Document representations		Index Size	Online Latency
	Nature	Count		
Single-vector methods, e.g. DSSM [13], DESM [28], S.BERT [35], ANCE [43]	semantic only	only one per document	$O(D)$	$O(\log D)$
ME-BERT [24], MVR [48], PQR [39]	semantic only	multiple, fixed hyperparameter m	$O(mD)$	$O(\log(mD))$
ColBERT [19], MAIZE [38], COIL [9]	semantic only	multiple, tied to document length t_d	$O(t_d D)$	$O(t_q \log(t_d D))$
MVG applied on any single-vector approach (this paper)	semantic and behavioral	multiple, variable per document based on hyperparameter β , $1 + \delta$ on average	$O((1 + \delta)D)$	$O(\log((1 + \delta)D))$

PROBLEM 1 (BEHAVIORAL REPRESENTATION LEARNING). *Given query-document relevance dataset (Q, \mathcal{D}, w) , semantic representations of queries and documents $\{\mathbf{x}_q\}_{q \in Q}$, $\{\mathbf{x}_d\}_{d \in \mathcal{D}}$ from a bi-encoder and a budget M on the total number of behavioral representations across documents, **learn** behavioral document representations $\{\mathbf{v}_{dj} \mid j \in [1, m_d]\}_{d \in \mathcal{D}}$ under the given budget so as to **maximize** the relevance of historically associated query-document pairs.*

$$\text{maximize } \sum_{q \in Q, d \in \mathcal{D}} w_{qd} \text{rel}(q, d) \quad \text{s.t.} \quad \sum_{d \in \mathcal{D}} m_d \leq M \quad (3)$$

Problem 1 aims to maximize the relevance score of past query-document associations within a given memory budget, essentially seeking better memorization of historical relevance data within the bi-encoder representational space. This sets the stage for an improved recall, which is crucial for DR approaches used for first-stage retrieval. The above maximization objective can be further expanded using the max-sim relevance score from Equation (2) as

$$\sum_{q \in Q, d \in \mathcal{D}} w_{qd} \text{rel}(q, d) = \sum_{q \in Q, d \in \mathcal{D}} w_{qd} \max_{j \in [0, m_d]} \mathbf{x}_q^\top \mathbf{v}_{dj} \quad (4)$$

where $\mathbf{v}_{d0} := \mathbf{x}_d$ is fixed, $\|\mathbf{v}_{dj}\|_2 = \|\mathbf{x}_q\|_2 = 1 \forall q, d, j$, and queries are assumed to be represented as single vector each.

Jointly solving Problem 1 for behavioral document representations $\{\mathbf{v}_{dj}\}_{j \in [1, m_d], d \in \mathcal{D}}$ and their count $\{m_d\}_{d \in \mathcal{D}}$ only permits local search algorithms [26] which have no quality guarantees and are also computationally expensive. Therefore, we instead seek a two-step approach which can determine $\{m_d\}_{d \in \mathcal{D}}$ in a heuristic manner, and then solve the simplified Problem 1 exactly to compute the optimal behavioral representations $\{\mathbf{v}_{dj}\}_{j \in [1, m_d], d \in \mathcal{D}}$.

4 THEORETICAL CONNECTION

We leverage Pitman-Yor Process [34] (PYP) to explain the skewed distribution of query diversity from Figure 1 (right). In this section, we provide background on PYP and draw the connection to document retrieval; later in Section 5, we show how to use it for behavioral budget distribution among documents.

Pitman-Yor Process (PYP) [34] is a discrete-time stochastic process that generalizes the popular Chinese Restaurant Process [1] to accommodate power-law tails. In a PYP, customers arrive sequentially to be seated in a restaurant with infinite tables, each with

infinite capacity. Let z_n be the choice of table for n^{th} customer. After the arrival of n^{th} customer, let $T_{n,k}$ be the number of customers at table k and let T_n denote the number of occupied tables. The table z_{n+1} picked by the $n + 1^{\text{th}}$ customer is distributed as

$$P(z_{n+1} | z_1, \dots, z_n) = \begin{cases} \frac{T_{n,k} - \beta}{n + \alpha}, & 1 \leq k \leq T_n \quad \text{occupied table} \\ \frac{\alpha + \beta T_n}{n + \alpha}, & k = T_n + 1 \quad \text{new table} \end{cases} \quad (5)$$

where $\alpha > 0$ and $\beta \in (0, 1)$ are concentration parameters: for low values of α and β , customers tend to crowd around fewer tables. When choosing to sit at one of the occupied tables, observe that customers prefer to sit at popular tables having higher values of $T_{n,k}$; thus PYP naturally captures the ‘rich get richer’ phenomenon.

Connection to Document Retrieval: We posit that relevant queries for a document have a *clustered distribution*, where each *query cluster* encodes a latent *intent* for accessing the document (e.g. kitchen and camping are two different intents to buy the same gas lighter product). The number of query clusters is not known in advance, and can grow as more queries are observed. There is also a notion of ‘rich get richer’ as the search engine typically uses mechanisms such as query auto-complete or auto-correct to ensure that new queries are similar to common queries in the past. All these characteristics make PYP a natural fit to model queries used to retrieve a document via search engine. Concretely, each document has an associated PYP according to which its associated queries (customers) arrive and map to query clusters or intents (tables).

Query Diversity under PYP: The consequence of the above connection is our ability to leverage classical results from discrete-time stochastic processes to distribute the total budget M of behavioral representations among documents in a principled manner. Lemma 1 guarantees that the number of tables T_n in a PYP grows sublinearly as $O(n^\beta)$ in terms of the number of customers n .

LEMMA 1 (PITMAN [33] §3.3). *The expected number of tables (T_n) occupied by n customers under Pitman-Yor Process (Equation (5)) is:*

$$\mathbb{E}(T_n | \alpha, \beta) = \frac{\alpha}{\beta} \left\{ \prod_{j=1}^n \frac{\alpha + \beta + j - 1}{\alpha + j - 1} - 1 \right\} \asymp \frac{\Gamma(\alpha + 1)}{\beta \Gamma(\alpha + \beta)} n^\beta$$

where Γ is the gamma function and \asymp indicates asymptotic equality.

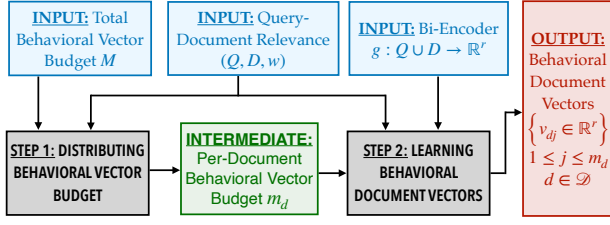


Figure 2: Overview of the proposed method: MVG (1) splits a total behavioral vector budget into per-document vector budgets, and (2) uses a Bi-encoder to learn additional document vectors to be indexed for nearest-neighbor search. Both steps utilize the query-document relevance data.

Algorithm 1 MVG Algorithm (Step 2) for Learning Behavioral Document Representations

Input: Semantic document vector \mathbf{x}_d from bi-encoder; query vectors and relevance weights $\{\mathbf{x}_q, w_{qd}\}_{q \in Q}$; budget m_d of behavioral vectors.

- 1: **procedure** MVG($\mathbf{x}_d, \{\mathbf{x}_q, w_{qd}\}_{q \in Q}, m_d$)
 - 2: Cluster assignment matrix $S_{qj} := 0, \forall q \in Q, j \in [0, m_d]$
 - 3: **for** $q \in Q$ **do** ▷ random initialization
 - 4: Pick $j' \sim \{0, \dots, m_d\}$ uniformly at random;
 - 5: Set $S_{qj'} := 1$.
 - 6: Fix one cluster center $\mathbf{v}_{d0} := \mathbf{x}_d$.
 - 7: **repeat**
 - 8: **for** $q \in Q$ and $j = 0, \dots, m_d$ **do** ▷ update assignments
 - 9: $S_{qj} = 1[j = \arg \max_{0 \leq j' \leq m_d} \mathbf{x}_q^\top \mathbf{v}_{dj'}]$
 - 10: **for** $j = 1$ **to** m_d **do** ▷ update cluster centers
 - 11: $\mathbf{v}_{dj} = \sum_q S_{qj} w_{qd} \mathbf{x}_q / \sum_q w_{qd} S_{qj}$
 - 12: $\mathbf{v}_{dj} = \mathbf{v}_{dj} / \|\mathbf{v}_{dj}\|_2$
 - 13: **until** matrix S does not change.
 - 14: **return** behavioral document representations $\{\mathbf{v}_{dj}\}_{j=1}^{m_d}$.
-

Thus, if all documents follow identical and independent Pitman-Yor Processes, the resulting distribution of query diversity across documents (measured in terms of the number of query clusters) also exhibits similar behavior as stated in Corollary 1. Our proposed approach leverages this result as we discuss next.

COROLLARY 1. *For a given document d under the Pitman-Yor Process Equation (5), the expected number of query clusters (\tilde{m}_d) after observing n_d queries is: $\mathbb{E}(\tilde{m}_d | \alpha, \beta) \asymp \frac{\Gamma(\alpha+1)}{\beta \Gamma(\alpha+\beta)} (n_d)^\beta$.*

5 OUR APPROACH

Our proposed approach, called Multi-View Geometric Index (MVG), works in two steps as shown in Figure 2. First, it determines how to distribute the total budget M of behavioral representations across all documents by leveraging the above connection to PYP (Section 5.1). Then, it computes the behavioral representations of a document by clustering its associated queries in the representational space learned by the Bi-encoder (Section 5.2). After describing these steps, Section 5.3 reviews how MVG fits within the dense retrieval pipeline, and also highlights its desirable properties for use in practice.

5.1 Step 1: Budget Distribution

How many behavioral representations m_d out of a total budget M should we allocate to a document d ? MVG leverages the connection to PYP from Section 4 to provide a principled answer: *the number of behavioral representations allocated to a document should be proportional to the number of observed query clusters for the document.* If a document d has $n_d = \sum_{q \in Q} 1[w_{qd} > 0]$ relevant queries in the training set, then:

$$m_d \propto n_d^\beta \quad (6)$$

where $\beta \in (0, 1)$ is now an experimental hyperparameter that allows us to smoothly interpolate between uniform distribution ($\beta = 0$) and popularity distribution ($\beta = 1$). Due to its ability to model a rich set of budget distribution schemes, the functional form in Equation (6) is widely used in other applications as well [14, 27].

5.2 Step 2: Representation Learning

Given the count m_d of behavioral representations to learn for a document d , our approach for learning them is to directly optimize the objective in Problem 1 using max-sim formulation in Equation (2). When rel is dot product, it is easy to show Equation (3) is the weighted k -means clustering objective with $k = m_d + 1$ and one constrained cluster center $\mathbf{v}_{d0} := \mathbf{x}_d$. Algorithm 1 simply adapts Llyod’s algorithm [23] for this setting.

5.3 MVG in Practice

We highlight that MVG is a light-weight approach that is easy for dense retrieval practitioners to use. As shown in Table 1, MVG can be implemented as a single step (step 3) in the standard DR pipeline, with no change in Bi-encoder training or inference procedure (steps 1-2), the nearest-neighbor (NN) index building step (step 4), or the online retrieval logic (step 5). The only impact of MVG is an increase in the number of document representations to be indexed for NN search. But due to efficient NN index data structures, this incurs only a sublinear increase in online retrieval latency [2].

An additional implementation detail when using MVG (or any approach learning multiple representations for documents such as ColBERT [19]) is that the nearest neighbor documents obtained online for a query can contain duplicates as a single document can be retrieved in multiple times due to its many representations. Thus additional deduplication is necessary. However, we find that the deduplication overhead is negligible as online retrieval latency is heavily dominated by that of NN-search.

6 EXPERIMENTS

We conduct an extensive evaluation of MVG to answer the following questions: **(Q1)** Applied on propriety e-commerce product search dataset (which is our motivating problem setting), does MVG outperform production dense retrieval baseline in offline metrics? **(Q2)** When A/B tested, does MVG improve customer engagement within acceptable latency regression? **(Q3)** Does MVG improve over state-of-the-art dense retrieval approaches across a diverse set of public datasets? **(Q4)** What is the behavior of MVG with respect to important hyperparameters, e.g. budget for behavioral representations and embedding dimensionality of base bi-encoder? **(Q5)** How can we understand the specific examples where MVG

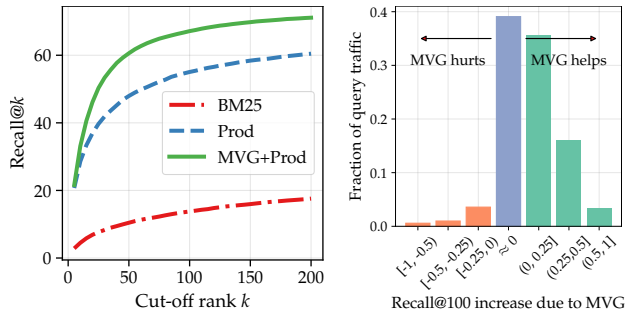


Figure 3: On ProdSearch Dataset, (Left) MVG improves over PROD on Recall@ k for all $k \in [1, 200]$. (Right) MVG improves Recall@100 for >55% of query traffic (green), while hurting only < 5% of query traffic (orange).

leads to substantial improvements? (Q6) How does MVG behave on datasets where its motivating factors (e.g. skewed distributions in Figure 1) are absent? Our findings for these questions are detailed in order in Section 6.1 through Section 6.6.

MVG Implementation: We use query-document pairs with positive relevance scores in the training data as input to MVG, treating all queries for a document as equally relevant. By default, we use $\beta = 0.5$ and learn $M_{\text{avg}} = 0.3$ additional vectors per document on average. We apply MVG as a wrapper over multiple baseline bi-encoder models described later.

Evaluation Metrics: We use Faiss [15] to retrieve documents that have the highest cosine similarity with the test queries and deduplicate the retrieved document list when evaluating MVG. We adopt two of the most commonly used information retrieval metrics, namely, *Recall* and *Average Precision* to evaluate the quality of models in retrieving relevant documents. More specifically, Recall measures the fraction of relevant documents within the retrieved document list, whereas Average Precision measures the goodness of relevant documents by their ranks within the retrieved document list. Due to high variability in the number of relevant documents across queries, we limit the retrieval to top $k \in \{10, 100\}$ documents with the highest cosine similarity. We report Recall@ k and MAP@ k (mean Average Precision at k) as the average of corresponding per-query metrics over all queries, in percentage.

6.1 Results on Proprietary ProdSearch Dataset

We took a uniform sample from fourteen months of anonymized aggregated search logs from an e-commerce search engine for this experiment. We used data recorded in the first twelve months for training and validation, setting aside the last two months for testing. The document corpus consists of about 60 million products. The baseline dense retrieval model (PROD) follows the network architecture from [13]. Both PROD and MVG were trained on 1M queries, and tested on 100K queries. For comparison, we also evaluate the standard BM25 model [37], which relies solely on the statistics of overlapping terms between queries and documents.

The overall columns in Table 3 show the metrics computed on all test queries. We see that MVG applied to PROD (MVG +PROD)

consistently outperforms PROD alone by improving Recall@ k between 4.6%–12.4%, and MAP@ k between 0.4%–4%. These gains are statistically significant according to two-sided macro-sign test [45], and also generalize to $1 \leq k \leq 200$ as shown in Figure 3 (left). In contrast, BM25 underperforms, indicating its inability to retrieve relevant documents by pure lexical matching from a large corpus.

The **memorization** and **generalization** columns in Table 3 show the break-down of metrics by queries that appeared in and were absent from the training set, respectively. Even though MVG primarily targets memorization of query-document pairs in the training set, we noticed that it also generalized to new queries. These results showcase the effectiveness of memorization in dense representation space: memorized queries for a document can also help new similar queries discover the relevant document.

Figure 3 (right) shows the fraction of query traffic in which MVG has gained or lost in Recall@100 compared to PROD alone. We see that the Recall@100 of MVG increased in over 55% of query traffic, matched with PROD in about 40% of query traffic, but lost only in less than 5% query traffic. This analysis shows that MVG is effective and safe to use even though the additional queries may also, in theory, invite more irrelevant documents into retrieval results.

6.2 A/B Testing in E-Commerce Product Search

An e-commerce service typically collects first-stage retrieval results from several sources like lexical matchers (inverted indices), semantic matchers (dense retrieval methods), upstream machine learning models, and advertised products. To evaluate MVG online, we replace only the production dense retrieval method (X) with its MVG-augmented version (X+MVG) and measure improvements in (a) **customer engagement**: number of products purchased (*Units*) and the amount of product sales (*PS*), (b) **relevance quality**: percentage of exact or substitute matches (*E+S*) within the top-16 products, (c) **sparse results**: percentage of queries with less than 16 products retrieved (*SR*), and (d) **latency P90**: value of latency at 90th percentile traffic. Table 4 tabulates the results. We observe that MVG brings statistically significant improvements in customer engagement metrics and sparse results without impacting the overall latency. However, there is a slight degradation in relevance quality metrics, which we discuss in Section 7 and leave for future work.

6.3 Results on Publicly Available Datasets

We evaluate MVG on publicly available datasets which have the characteristics that motivated our approach: skewed distribution in frequency and diversity of queries across documents from Figure 1.

Dataset Description: (1) **EurLex** [4]² for labeling legal documents with one or more EUROVOC concepts (e.g. international affairs, data processing). We use document titles as queries and concepts as documents for dense retrieval. The dataset consists of 4271 documents and 45K, 6K, and 6K queries in the training, validation, and test sets, respectively. (2) **PubMed**³ [17] for classification of biomedical articles into one or more NIH Medical Subject Headings (MeSH) (e.g. fibrosis, humans, and mutation). We use article titles as queries, and MeSH labels as documents. After preprocessing and splitting, the dataset consists of 27K documents and 1.2M, 1K,

²<https://huggingface.co/datasets/eurlax>

³<https://www.kaggle.com/bonhart/pubmed-abstracts>

Table 3: Results on proprietary ProdSearch dataset: Metrics are reported over all test queries (“overall”), test queries seen during training (“memorization”), and those not seen during training (“generalization”). Per-query metric values are aggregated by accounting for the traffic contribution of queries. Bold and underline typefaces follow the same convention as in Table 5.

Method	Recall@ k						MAP@ k						Index Size (GB)
	Overall		Memorization		Generalization		Overall		Memorization		Generalization		
	$k = 10$	$k = 100$	$k = 10$	$k = 100$	$k = 10$	$k = 100$	$k = 10$	$k = 100$	$k = 10$	$k = 100$	$k = 10$	$k = 100$	
BM25	4.50	13.75	4.13	13.25	13.23	25.47	1.92	2.64	1.72	2.45	6.60	7.13	N/A
PROD	28.72	55.39	28.50	55.49	33.86	53.00	17.13	21.48	17.04	21.54	19.31	20.19	30.0
MVG + PROD	<u>33.33</u>	<u>67.77</u>	<u>33.19</u>	<u>68.14</u>	<u>36.76</u>	<u>59.04</u>	<u>17.55</u>	<u>25.54</u>	<u>17.47</u>	<u>25.74</u>	<u>19.51</u>	<u>20.73</u>	30.8

Table 4: A/B test results showing % improvements from MVG over production system. Underline indicates statistical significance while arrows indicate whether higher (\uparrow) or lower (\downarrow) is better on a metric. All metrics are defined in Section 6.2.

Units (\uparrow)	PS (\uparrow)	E+S (\uparrow)	SR (\downarrow)	Latency P90 (\downarrow)
<u>+0.24%</u>	<u>+0.06%</u>	<u>-0.18%</u>	<u>-1.21%</u>	-3ms

and 100K queries for the training, validation, and test sets, respectively. (3) **WikiSeeAlso**⁴ [20] for predicting the titles of related Wiki pages from a given webpage title. The dataset consists of 300K documents and 500K, 138K, and 177K queries in the training, validation, and test sets, respectively.

Baselines: We compare MVG to the following: (i) **BM25** [37] as before. (ii) **Popularity**, which ranks documents based on the count of unique queries associated in training set. Even though the same document list is applied to all test queries, this baseline remains competitive on datasets with skewed distribution of association between queries and documents. (iii) **Deep Structured Semantic Model** (DSSM) [13] using SentencePiece Byte-Pair Encoding [3] to tokenize text. The tokenizers are trained on both queries and documents to learn a vocabulary of size 40K for EurLex, 300K for PubMed, and 100K for WikiSeeAlso. The vocabulary sizes were set to roughly half the number of unique words in the respective datasets. (iv) **SentenceBERT** (S.BERT) [35] initialized using domain-specific pretrained BERT: LegalBERT⁵ for EurLex, PubMedBERT⁶ for PubMed, and BERTBase⁷ for WikiSeeAlso and fine-tuned using default hyper-parameter settings. (v) **ColBERT** [19] using the default configuration. This is the state-of-the-art dense retrieval method learning multiple representations per document.

MVG vs Single-Vector Bi-encoders: We apply MVG as a wrapper on top of DSSM and S.BERT. We do not compare with solutions built for the datasets that are not of bi-encoder architecture because they are not compatible with MVG. Table 5 summarizes the performance of all approaches on all the public datasets. Popularity is more competitive than BM25 on EurLex and PubMed which exhibit skewed query-document distribution but not on WikiSeeAlso that has the least skewed distribution among the three datasets as shown in Figure 1. The dense retrieval approaches DSSM and S.BERT achieve higher metrics, with S.BERT yielding higher metrics on PubMed dataset where semantic understanding of documents

is more crucial. Importantly, MVG consistently outperforms both these baselines, delivering upto 27.27% gain in Recall@100 and 22.95% gain in MAP@100 – at the cost of only 1.3 \times increase in index size. Thus MVG can bring value in a wide variety of settings across diverse datasets and underlying bi-encoders.

MVG vs Multi-Vector Bi-Encoder: Importantly, MVG also improves over the state-of-the-art multi-vector ColBERT baseline which learns multiple semantic (but no behavioral) representations per document (and query). Both MVG and ColBERT can be viewed as improvements over the same single-vector S.BERT bi-encoder to incorporate multiple behavioral representations and multiple semantic representations respectively. However, as seen in Table 5, on datasets with motivating skewed distributions, behavioral representations turn out to be much more effective. Thus, MVG + S.BERT outperforms ColBERT by a statistically significant 3 – 15% recall and 1 – 17% MAP by requiring 4 – 6 \times smaller space as measured by number of million floating points stored within the NN-index. Our results suggest that improvements from MVG will likely persist over recent dense retrieval methods which optimize the memory overhead of ColBERT by sacrificing some retrieval performance Gao et al. [9]. Finally, Figure 5 illustrates that wide spectrum of memory overhead that MVG offers a practitioner to play with, compared to ColBERT which is a fixed point in the figure; this makes MVG better suited to practical settings with stricter latency and memory requirements.

6.4 Effect of Hyper-parameters and Base Model

We study how the performance depends on choice of hyperparameters for MVG and the capacity and quality of base bi-encoder.

Budget Hyper-parameters: We vary total budget M by varying the average number of behavioral vectors per document $M_{\text{avg}} \in [0.01, 3]$. Figure 4 (left) illustrates how the relative budget allocated to head, torso, tail documents varies with β (these are the top, middle, bottom 33%^{1e} documents based on number of queries n_d). As $\beta \rightarrow 0$, all documents get equal budget, and when $\beta \rightarrow 1$, 80% budget is allocated to head documents which account for most unique queries. Intermediate values provide more reasonable budget distributions, and thus we use them to chart the variation of Recall@100 with M_{avg} in Figure 4 (right). For a given β , as we increase M_{avg} , the recall steeply improves due to the improved memorization. The best recall is achieved for $\beta = 0.3$, $M_{\text{avg}} = 0.5$. The recall falls gradually as M_{avg} is increased beyond its optimal value however, suggesting the MVG overfits to training data and hurts generalization. Larger values of β overfit more severely to the head documents, and hence show a steeper decrease in this regime. Importantly, observe that for

⁴<http://manikvarma.org/downloads/XC/XMLRepository.html>

⁵<https://huggingface.co/nlpauub/legal-bert-base-uncased>

⁶<https://huggingface.co/microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract>

⁷<https://huggingface.co/bert-base-uncased>

Table 5: MVG outperforms baselines on publicly available datasets in Recall@k and MAP@k: Bold indicates best method under each metric within each row group, while underline signifies statistically significant differences ($p < 0.001$) with respect to the second best method in the same row group according to a two-sided macro-sign test [45].

Method	Recall@100			MAP@100			Index Size (# Floats in Millions)		
	EurLex	PubMed	WikiSeeAlso	EurLex	PubMed	WikiSeeAlso	EurLex	PubMed	WikiSeeAlso
BM25	18.95	20.55	25.90	7.26	11.22	9.55	N/A		
Popularity	46.26	33.65	0.20	8.35	10.35	0.12	N/A		
DSSM [13]	67.20	20.24	35.44	31.83	6.58	11.42	1.84	7.52	78.95
MVG + DSSM	81.62	49.01	37.90	51.28	16.70	13.73	2.40	9.77	102.63
S.BERT [35]	50.58	35.49	29.38	14.75	6.87	9.66	5.53	22.56	236.86
MVG + S.BERT	78.15	59.62	42.12	47.87	17.73	13.78	7.19	29.32	307.92
ColBERT [19]	73.47	44.88	39.01	31.24	14.35	12.70	30.66	170.22	1748.16

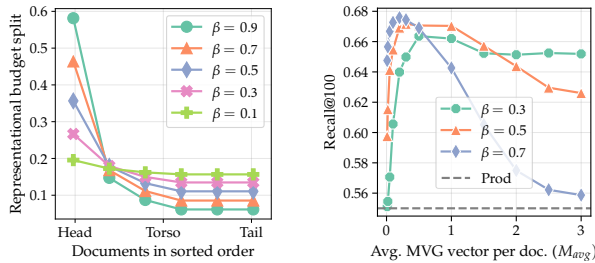


Figure 4: On ProdSearch dataset, (left) distribution of representational budget among documents based on β . (right) MVG outperforms PROD for all hyper-parameter settings.

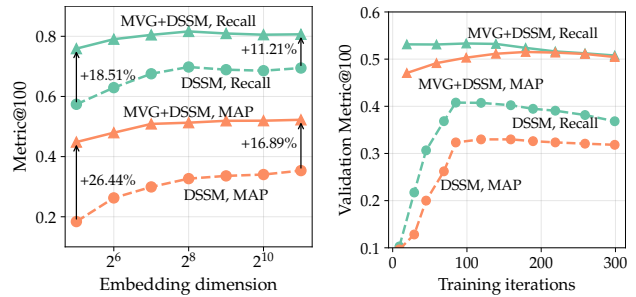


Figure 6: On Eulex dataset, (Left) MVG improvement with variation of embedding dimensions (Right) MVG improvement throughout the training iterations.

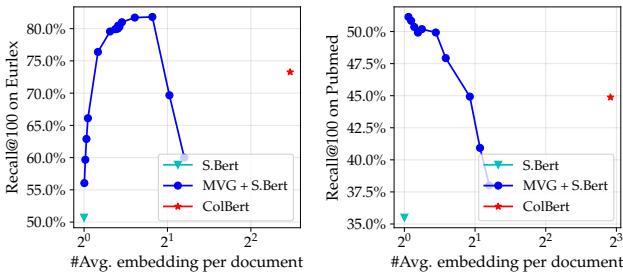


Figure 5: For a given bi-encoder (here, S.BERT), ColBERT incurs a fixed memory overhead which depends on document length but is out of practitioner’s control. In contrast, MVG +S.BERT exposes a wider spectrum of memory overhead that is better suited to practical settings having strict memory or latency limits.

all hyper-parameters, MVG consistently matches or outperforms the PROD baseline (gray dashed line).

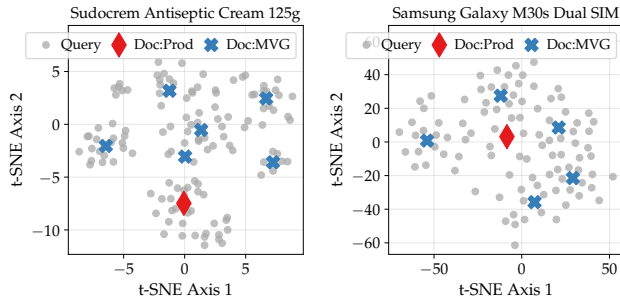
Embedding Dimension of Base Bi-Encoder: In Figure 6 (left), we vary the embedding dimensionality r of DSSM, and compare it to r -dimensional DSSM +MVG with $1.3\times$ memory overhead. We see that the recall of DSSM increases steadily with larger dimensions. This behavior is expected based on [36]; and MVG +DSSM follows this pattern too. In particular, at $r = 32$, MVG brings a 18% recall improvement (from 58% to 76%) compared to DSSM with just $1.3\times$

memory overhead. However, even with 100% memory overhead (doubling dimensionality), DSSM with $r = 64$ attains a recall of just 63%. This shows that index size increase by allocating non-uniform representational budget (e.g. more vectors to head documents) performs better than uniformly increasing representational budget across all documents (e.g. dimensionality increase).

Base Model Quality: To effectively produce base model DSSM of varying quality, we set a limit on the training iterations (or training set size) for base model. Figure 6(right) shows the metrics on EurLex dataset for base model DSSM and MVG (applied to DSSM) at various training iterations. The Recall@100 and MAP@100 of DSSM model gradually increase as training progresses before showing sign of model overfitting towards the end where metrics decreased. In the early stage of training, MVG improves over DSSM with 42.8% for Recall@100 and 37.4% on MAP@100, which indicates that MVG can improve on DSSM even when it has not fully converged. At the stage where DSSM attains the highest Recall@100, MVG shows 12.5% and 19.2% improvement over DSSM on Recall@100 and MAP@100 respectively. We also evaluate the impact of MVG over pretrained BERT models with no task-specific finetuning on EurLex dataset: BERTBase pretrained on general data source, LegalBERT pretrained on European legislation documents. As shown in Table 6, MVG brings large recall and MAP improvements, suggesting that applying MVG over pretrained bi-encoders can potentially serve as a light-weight alternative to expensive fine-tuning when memorization is the primary objective.

Table 6: On Eurlex dataset, MVG improves over BERTBASE and LEGALBERT models on two metrics.

Method	Recall@ k		MAP@ k	
	$k = 10$	$k = 100$	$k = 10$	$k = 100$
BERTBASE	0.66	3.42	0.23	0.31
MVG + BERTBASE	55.84	76.20	43.46	46.58
LEGALBERT	3.74	11.53	1.92	2.26
MVG + LEGALBERT	56.15	77.15	43.53	46.90

**Figure 7: On ProdSearch dataset, t-SNE [40] visualization of vectors query vectors (grey dots), document vectors (red diamond). MVG captures the relevance of far-away queries by having several auxiliary vectors (blue cross).**

6.5 Case Studies

Figure 7 depicts case studies from ProdSearch dataset to help understand where and how MVG helps compared to the underlying bi-encoder. For each document, we display its semantic vector learned by PROD as a red diamond, behavioral vectors learned by MVG as blue crosses, and queries associated with the document in the training data as grey dots—after projecting all vectors into two dimensions using t-SNE [40]. While L_2 distances in the projected space only approximately preserve cosine similarities in the original space, these plots provide an insight into the relative distances among all vectors of interest in the high-dimensional space. After examining such plots for many documents, we discover two types of scenarios where MVG proves most useful over PROD bi-encoder.

Figure 7 (left) depicts the first success scenario: the PROD’s semantic document vector captures only a single group of relevant queries (red diamond is near one grey cluster only), and MVG learns behavioral vectors (blue crosses) to capture other groups of relevant queries. Observe that the blue crosses are not only situated close to tight grey clusters (likely queries sharing the same intent) but are also far away from the red diamond (semantic document vector). This visually confirms that MVG learns behavioral vectors which complement the semantic PROD vector of the document, and thus representation power is not wasted trying to capture the same set of queries more than once. Figure 7 (right) shows the second success scenario: the PROD document vector is unable to capture any single group of relevant queries sufficiently well (red diamond is in a sparse central region away from most grey points). In this case,

Table 8: On MSMarco dataset [30] with flat document popularity distribution, MVG does not improve over single-vector or multi-vector baselines; notably, it does not hurt either.

Recall@ k	ANCE [43]	ANCE+MVG	ColBERT [19]
$k = 10$	58.35	58.35	60.83
$k = 100$	85.24	85.24	85.82
$k = 1000$	95.87	95.66	96.18

the PROD document vector is essentially wasted, while MVG successfully learns multiple behavioral vectors which better represent the dense clouds of grey points.

Table 7 provides examples of query-document pairs from ProdSearch training dataset that were missed by the top-100 retrieved documents of the PROD model, but were correctly captured by MVG. For these cases, a single vector for a document was insufficient; as MVG boosts the relevance scores of positive query-document pairs much more than negative ones (Figure 8), MVG is able to rectify these memorization gaps.

6.6 Results on Datasets w/o Skewed Distribution

How does MVG work on datasets where motivating characteristics from Figure 1 are absent? To answer this, we use the popular MS-Marco [30] question-answering dataset which has flat document popularity distribution, likely due to synthetic curation and heavy preprocessing. Table 8 compares the performance of state-of-the-art single-vector (ANCE [43]) and multi-vector (ColBERT [19]) methods to that of MVG + ANCE. We see that MVG does not bring gains over ANCE as expected; in this case, the improvements from multi-vector ColBERT method are marginal too. Importantly, even when motivating dataset characteristics are absent, MVG does not significantly decrease performance compared to underlying bi-encoder.

7 CONCLUSION AND FUTURE WORK

We presented a simple yet effective approach called MVG to improve any given bi-encoder model for first-stage retrieval by incurring only a marginal memory overhead. MVG was motivated by the skewed distributions in document popularity and query diversity in real-world retrieval settings, which have thus far largely been overlooked in the dense retrieval literature. To correct for this, our main ideas were to (1) carefully consider the distribution of representational budget among documents based on document properties (e.g. popularity); (2) learn behavioral representations for documents which can more flexibly memorize past query-document associations compared to the typical semantic representations which are limited by what can be inferred from intrinsic document features (e.g. tokenized text that is short). Extensive experiments over three large public datasets, a proprietary e-commerce search dataset, and an A/B test consistently demonstrated the merits of our ideas and the specific way in which MVG incorporates them—as long as the dataset characteristics align with our motivations. When dataset characteristics deviate, however, (e.g. MS Marco) we saw that MVG does not bring gains; but it does not significantly hurt either.

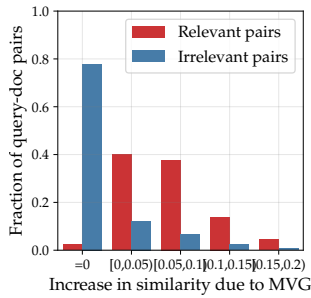


Figure 8: MVG selectively boosts similarity scores of relevant query-document pairs.

We hope that our work paves the way for next generation of dense retrieval approaches which more carefully consider the dual questions of representational budget distribution, and of jointly learning semantic and behavioral representations. Some concrete directions for future work are: (a) improving robustness to noise and overfitting (Table 4, Figure 5) by using a discriminative approach instead of clustering, (b) hyperparameter-free automatic way to infer the number of behavioral representations per document.

REFERENCES

[1] David J Aldous. 1985. Exchangeability and related topics. In *École d’Été de Probabilités de Saint-Flour XIII—1983*. Springer, 1–198.

[2] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. 2018. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*. World Scientific, 3287–3318.

[3] Kaj Bostrom and Greg Durrett. 2020. Byte Pair Encoding is Suboptimal for Language Model Pretraining. In *EMNLP (Findings)*, Vol. EMNLP 2020. 4617–4624.

[4] Ilias Chalkidis, Emmanouil Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2019. Extreme Multi-Label Legal Text Classification: A Case Study in EU Legislation. In *Proceedings of the Natural Language Language Processing Workshop*. Association for Computational Linguistics, 78–87.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. ACM, 191–198.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[7] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 1999. On Power-law Relationships of the Internet Topology. In *SIGCOMM*. ACM, 251–262.

[8] Lu Fan, Qimai Li, Bo Liu, Xiao-Ming Wu, Xiaotong Zhang, Fuyi Lv, Guli Lin, Sen Li, Taiwei Jin, and Keping Yang. 2022. Modeling User Behavior with Graph Convolution for Personalized Product Search. In *WWW*. ACM, 203–212.

[9] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *NAACL-HLT*. Association for Computational Linguistics, 3030–3042.

[10] Lise Getoor and Lilyana Mihalkova. 2011. Learning statistical models from relational data. In *SIGMOD Conference*. ACM, 1195–1198.

[11] Malay Haldar, Prashant Ramanathan, Tyler Sax, Mustafa Abdool, Lanbo Zhang, Aamir Mansawala, Shulin Yang, Bradley C. Turnbull, and Junshuo Liao. 2020. Improving Deep Learning for Airbnb Search. In *KDD*. ACM, 2822–2830.

[12] Sebastian Hofstätter, Omar Khattab, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Introducing Neural Bag of Whole-Words with ColBERTer: Contextualized Late Interactions using Enhanced Reduction. *CoRR* abs/2203.13088 (2022).

[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. ACM, 2333–2338.

[14] Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dube. 2016. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. In *ICLR*. OpenReview.net.

[15] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 3 (2021), 535–547.

Table 7: Examples from ProdSearch training dataset demonstrating gaps in memorization of Prod model: the similarity scores are low, leading to ranks worse than 100. MVG rectifies these gaps, leading to improved similarity scores and ranks.

Query	Document (Product)	Prod		MVG + Prod	
		Sim	Rank	Sim	Rank
automatic water plant	DIY Automatic Drip Irrigation Kit	0.65	>500	0.72	74
art for toddlers	Doodle Dog Arts and Crafts	0.63	265	0.75	24
gym gloves	Adidas Performance Gloves	0.64	>500	0.73	81
label manager	Brother Easy Portable Label Maker	0.66	112	0.75	8
arm mobile holder	Adidas Sports Armband for iPhone X	0.70	>500	0.80	19

[16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *EACL*. 427–431.

[17] Hager Kammoun, Imen Gabsi, and Ikram Amous. 2022. MeSH-Based Semantic Indexing Approach to Enhance Biomedical Information Retrieval. *Comput. J.* 65, 3 (2022), 516–536.

[18] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP*. 6769–6781.

[19] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*. ACM, 39–48.

[20] Sahiti Labhishetty, Ayesha Siddiqua, Rajivteja Nagipogu, and Sutanu Chakraborti. 2017. WikiSeeAlso: Suggesting Tangentially Related Concepts (See also links) for Wikipedia Articles. In *MIKE*, Vol. 10682. Springer, 274–286.

[21] Jiduan Liu, Jiahao Liu, Yang Yang, Jingang Wang, Wei Wu, Dongyan Zhao, and Rui Yan. 2022. GNN-encoder: Learning a Dual-encoder Architecture via Graph Neural Networks for Passage Retrieval. *CoRR* abs/2204.08241 (2022).

[22] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. Is a Single Vector Enough?: Exploring Node Polysemy for Network Embedding. In *KDD*. ACM, 932–940.

[23] Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28, 2 (1982), 129–136.

[24] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Trans. Assoc. Comput. Linguistics* 9 (2021), 329–345.

[25] Yury A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[26] W. Michiels, E. H. L. Aarts, and J. Korst. 2018. *Theory of Local Search*. Springer International Publishing, Cham, 299–339. https://doi.org/10.1007/978-3-319-07124-4_6

[27] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[28] Bhaskar Mitra, Eric T. Nalisnick, Nick Craswell, and Rich Caruana. 2016. A Dual Embedding Space Model for Document Ranking. *CoRR* abs/1602.01137 (2016).

[29] Anshul Mittal, Kunal Dahiya, Sheshansh Agrawal, Deepak Saini, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021. DECAF: Deep Extreme Classification with Label Features. In *WSDM*. ACM, 49–57.

[30] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. In *CoCo@NIPS (CEUR Workshop Proceedings, Vol. 1773)*. CEUR-WS.org, 1–10.

[31] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Allen Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic Product Search. In *KDD*. ACM, 2876–2885.

[32] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest. In *KDD*. ACM, 2311–2320.

[33] Jim Pitman. 2006. *Combinatorial stochastic processes: Ecole d’été de probabilités de saint-flour xxxii-2002*. Springer.

[34] Jim Pitman and Marc Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Ann. Probab.* 25, 2 (1997), 855–900.

[35] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP/IJCNLP*. Association for Computational Linguistics, 3980–3990.

- [36] Nils Reimers and Iryna Gurevych. 2021. The Curse of Dense Low-Dimensional Information Retrieval for Large Index Sizes. In *ACL/IJCNLP*. 605–611.
- [37] Stephen E. Robertson and Karen Spärck Jones. 1976. Relevance weighting of search terms. *J. Am. Soc. Inf. Sci.* 27, 3 (1976), 129–146.
- [38] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Maize: Effective and Efficient Retrieval via Lightweight Late Interaction. In *NAACL-HLT*. Association for Computational Linguistics, 1–12.
- [39] Hongyin Tang, Xingwu Sun, Beihong Jin, Jingang Wang, Fuzheng Zhang, and Wei Wu. 2021. Improving Document Representations by Generating Pseudo Query Embeddings for Dense Retrieval. In *ACL/IJCNLP (1)*. Association for Computational Linguistics, 5054–5064.
- [40] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605.
- [41] Xiao Wang, Craig Macdonald, Nicola Tonello, and Iadh Ounis. 2021. Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval. In *ICTIR*. ACM, 297–306.
- [42] Jason Weston, Ron J. Weiss, and Hector Yee. 2013. Nonlinear latent factorization by embedding multiple user interests. In *RecSys*. ACM, 65–68.
- [43] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *ICLR*. OpenReview.net.
- [44] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Charles Rosenberg, and Jure Leskovec. 2020. MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In *KDD*. ACM, 2434–2443.
- [45] Yiming Yang and Xin Liu. 1999. A Re-Examination of Text Categorization Methods. In *SIGIR*. ACM, 42–49.
- [46] Yingrui Yang, Yifan Qiao, and Tao Yang. 2022. Compact Token Representations with Contextual Quantization for Efficient Document Re-ranking. In *ACL (1)*. Association for Computational Linguistics, 695–707.
- [47] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *SIGIR*. ACM, 1503–1512.
- [48] Shunyu Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Multi-View Document Representation Learning for Open-Domain Dense Retrieval. In *ACL (1)*. Association for Computational Linguistics, 5990–6000.
- [49] Giulio Zhou and Jacob Devlin. 2021. Multi-Vector Attention Models for Deep Re-ranking. In *EMNLP*. Association for Computational Linguistics, 5452–5456.