Incorporating Token Importance in Multi-Vector Retrieval

Archish S[*†], **Ankit Garg, Kirankumar Shiragur, Neeraj Kayal**

Microsoft Research, India
archish17@gmail.com,{garga, kshiragur, neeraka}@microsoft.com

Abstract

ColBERT introduced a late interaction mechanism that independently encodes queries and documents using BERT, and computes similarity via fine-grained interactions over token-level vector representations. This design enables expressive matching while allowing efficient computation of scores, as the multi-vector document representations could be pre-computed offline. ColBERT models distance using a Chamfer-style function: for each query token, it selects the closest document token and sums these distances across all query tokens.

In our work, we explore enhancements to the Chamfer distance function by computing a weighted sum over query token contributions, where weights reflect the token importance. Empirically, we show that this simple extension, requiring only token-weight training while keeping the multi-vector representations fixed, further enhances the expressiveness of late interaction multi-vector mechanism. In particular, on the BEIR benchmark, our method achieves an average improvement of 1.28% in Recall@10 in the zero-shot setting using IDF-based weights, and 3.66% through few-shot fine-tuning.

Code — https://github.com/kayalneeraj/weighted_chamfer

Introduction

Recent advances in natural language understanding have highlighted the power of deep language models for dense retrieval and document ranking. Building on this progress, (Khattab and Zaharia 2020) introduced ColBERT, a late interaction architecture that leverages BERT (Devlin et al. 2019) to independently encode queries and documents. ColBERT enables efficient query-time scoring by precomputing token-level multi-vector representations for documents, while still capturing the rich semantics of deep models. ColBERT computes the relevance between a query and a document by comparing two sets of token-level vectors: for each query token, it identifies the most similar document vector using ℓ_2 distance, and aggregates these distances across all query tokens. This approach has proven both efficient and effective (Khattab and Zaharia 2020), outperforming earlier BERT-based and traditional retrieval models.

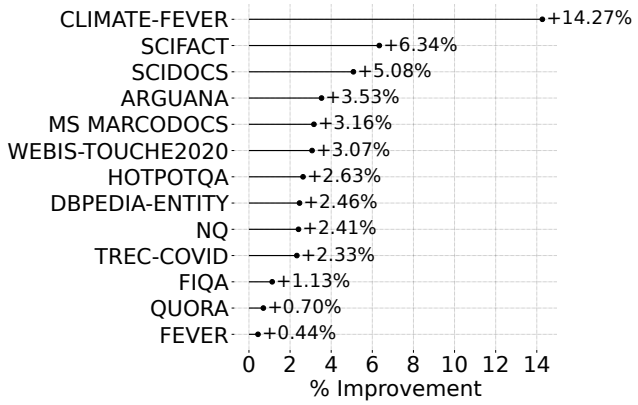

Figure 1: Effectiveness of Weighted Chamfer over Chamfer (ColBERTv2) on Recall@10 evaluated across the BEIR benchmark

Traditional dense retrieval methods (Devlin et al. 2019; Nogueira and Cho 2020), although popular for their ability to capture semantic information, yet often fall short in capturing fine-grained nuances, as they compress entire queries and documents into a single vector embedding. In contrast, classical retrieval systems like BM25 (Robertson et al. 1995) excel at modeling term-level granularity through bag-of-words representations, though they lack semantic depth. Multi-vector retrieval systems aim to combine the strengths of both paradigms by representing each token with its own vector. However, current multi-vector approaches typically overlook term importance; a key factor that has contributed significantly to the success of BM25 and related models.

In this work, we build upon the multi-vector retrieval framework and propose a novel extension to the Chamfer distance function. Rather than aggregating query token contributions uniformly, we introduce a weighted summation mechanism where each token's contribution is scaled by a learned weight associated with its token ID. This weighing scheme captures the relative importance of tokens in determining query-document similarity, thereby enhancing the expressiveness of the model and improving ranking performance. This extension is particularly simple to incorporate into existing multi-vector pipelines and does not add any

[*]Corresponding Author.
[†]Work done while at Microsoft Research.

additional latency cost during the inference stage.

In ColBERT-style architectures, token vectors are typically normalized to unit norm prior to computing Chamfer distance (Khattab and Zaharia 2020; Santhanam et al. 2022b). Under this constraint, incorporating explicit token weights becomes especially important, as vector norms can no longer encode importance. While one might consider using unnormalized vectors to allow the model to implicitly encode token importance, e.g., by assigning lower norms to less relevant tokens, it remains unclear whether such behavior emerges reliably in practice. Although prior work has explored connections between embedding norms and word importance in static embeddings (Oyama, Yokoi, and Shimodaira 2022), similar evidence for contextual embeddings remains limited.

One of the notable strengths of the ColBERT model is its impressive out-of-distribution generalization. Although trained solely on the MS MARCO dataset, ColBERT demonstrates strong zero-shot performance on the BEIR benchmark. In this work, we show that our simple weighted extension can give a significant improvement over ColBERT. In particular, a simple static weighting scheme based on inverse document frequency (IDF) computed from the document collection, yields an average gain of 1.28% and maximum gain of 3.16% in Recall@10 across BEIR datasets over ColBERTv2. Furthermore, by fine-tuning only the token weights (rather than the full model parameters) using limited human-labeled data, we observe an average gain of 3.66% and maximum gain of 14.27% over ColBERTv2.

Our empirical results thus demonstrate the adaptability and effectiveness of our approach. These results underscore the potential of parameterized similarity functions in enhancing multi-vector retrieval, especially in zero-shot and low-resource settings. Given the limited availability of human-labeled relevance data and high cost in generating LLM based synthetic training data (an approach that has gained traction, e.g., (Dai et al. 2022; Wang et al. 2023)), our method offers a lightweight alternative approach offering superior performance with limited resources. We also provide theoretical proofs under standard assumptions and techniques from statistical learning theory arguing about the sample complexity needed to learn the weights and also establish generalization bounds.

To learn the token weights, we employ a contrastive loss function, which tries to minimize the distance between the queries and positive documents relative to the negative ones. Notably, when the set of negative examples is fixed, the loss becomes a convex function with respect to the token weights, offering theoretical advantages and simplifying optimization. However, we find that training is most effective when hard negatives are selected iteratively, adapting to the current set of learned weights. A distinctive aspect of our training procedure is the use of a convex combination of two contrastive loss functions, each differing in the number of negatives per query, and we have found that this approach works best for finding the best set of token weights. This technique may prove valuable in other contrastive learning settings as well.

Related Work

Initial neural information retrieval (IR) approaches based on pre-trained transformers leverage BERT (Devlin et al. 2019; Nogueira and Cho 2020; MacAvaney et al. 2019) to model query-document relevance. These approaches also referred to as cross-encoders compute a similarity score by concatenating the query document pair and jointly encoding them using BERT. While effective, these methods are computationally expensive, as they require re-encoding each query-document pair during inference. Methods like SPLADE (Formal, Piwowarski, and Clinchant 2021; Formal et al. 2021) reformulate the sparse retrieval paradigm by explicitly invoking sparse regularization to improve efficiency.

In contrast to cross encoders, bi-encoders (Karpukhin et al. 2020; Macdonald, Tonellotto, and Ounis 2021; Qu et al. 2020) encode queries and documents independently into high-dimensional vectors using separate BERT-based encoders and compute similarity through ℓ_2 distance. This design allows for fast retrieval through pre-computed document embeddings and approximate nearest neighbor (ANN) search techniques (Johnson, Douze, and Jégou 2019; Malkov and Yashunin 2018; Subramanya et al. 2019).

ColBERT (Khattab and Zaharia 2020) addresses the lower expressivity of bi-encoders by representing queries and documents as sets of token-level embeddings and computing similarity using a late-interaction mechanism. This approach enables the pre-computation of document representations. Building on this, ColBERTv2 (Santhanam et al. 2022b) introduces improved training strategies and indexing optimizations to support retrieval over billion-scale corpora. To further reduce latency, PLAID (Santhanam et al. 2022a) incorporates centroid interaction and pruning mechanisms, which eliminate low-scoring documents early in the search process, significantly improving efficiency without sacrificing retrieval quality.

MUVERA (Dhulipala et al. 2024) seeks to bridge the gap between single-vector and multi-vector retrieval by approximating multi-vector similarity with fixed-dimension single vector encodings. While this design improves compatibility with highly optimized Maximum Inner Product Search (MIPS) solvers, it introduces a trade-off: its reliance on large high-dimensional vectors increases memory consumption, impacting storage efficiency.

More recent work such as XTR (Lee et al. 2024) refines ColBERT's scoring by selecting and ranking only a subset of tokens through a modified training objective, thereby reducing computational cost. In parallel, ConstBERT (MacAvaney, Mallia, and Tonellotto 2025) introduces a fixed-size representation for each document by pooling token embeddings into a smaller set of learned vectors, decoupled from individual input tokens. This improves storage efficiency and simplifies late interaction, making it more practical for downstream re-ranking. Other research on improving the latency and performance of multi-vector retrieval systems includes (Humeau et al. 2019; Gao, Dai, and Callan 2021).

(Qian et al. 2022) introduce an alignment framework to generalize ColBERT's late interaction mechanism. There are similarities to our approach but major differences in terms of parametrization and training objectives.

Preliminaries

While prior work on improving ColBERT has largely focused on reducing latency by pruning or compressing document vectors, our approach aims to enhance the expressiveness of the late interaction itself by reweighting query tokens. We employ the ColBERT model to embed the queries and the documents into multi-vector representations at the token-level.

The ColBERT architecture comprises of a) query encoder b) document encoder, and c) the late-interaction mechanism. Let q denote a query, which is a collection of tokens $\{q_i\}_{i=1}^{\text{len}(q)}$ with $q_i \in [T] \forall i$, where T denotes the total number of tokens in the universe. The ColBERT's query encoder encodes the query q into a bag of fixed size unit-norm embeddings $E(q) = \{E_q(q_i)\}_{i=1}^{\text{len}(q)} \subseteq \mathbb{R}^d$ while the document encoder encodes every document $d = \{d_j\}_{j=1}^{\text{len(d)}}$ in the document corpus $\mathcal{D}$ into another bag of unit-norm embeddings $E(d) = \{E_d(d_j)\}_{j=1}^{\text{len}(d)}$ based on the tokens in q or d respectively. Throughout this work, the query and the document encoders are unchanged and used as is from ColBERTv2.

The late-interaction through Chamfer Distance (eq. (1)) allows us for fast computation of relevance scores by enabling pre-computation of document token embeddings, while requiring only query token embeddings to be computed during the inference stage. The late-interaction mechanism involves matching every token q_i in the query against the tokens in the document via the smallest ℓ_2 distance between $E_q(q_i)$ and $E_d(d_j)$ vectors. We now formally define the Chamfer Distance.

Definition 1 (Chamfer Distance (Khattab and Zaharia 2020))**.** *Given a query-document pair* (q, d)*, the Chamfer Distance is defined as*

$$\text{Chamfer} = \eta(q, d) \coloneqq \frac{1}{\text{len}(q)} \sum_{i=1}^{\text{len}(q)} \min_{j \in [\text{len}(d)]} \|E_q(q_i) - E_d(d_j)\|_2 . \tag{1}$$

We replace the MaxSim operator introduced in ColBERT with the Euclidean distance instead to measure the document irrelevancy for elegance, aliased MinDist. This modification essentially keeps the performance unchanged.

Our Contribution

In this work, we present Weighted Chamfer, a modified Chamfer Distance (eq. (2)) that improves performance on out-of-domain datasets, at virtually no increase in latency. This variant reflects the *importance* of the query token q_i in q, analogous to the word importance measures used in BM25 and related systems. The formal definition of Weighted Chamfer distance is as follows.

Definition 2 (Weighted Chamfer Distance)**.** *Given a query-document pair (q, d), the Weighted Chamfer distance is defined as*

$$\text{WeightedChamfer} = \eta_w(q, d) \coloneqq \frac{1}{\text{len}(q)} \sum_{i=1}^{\text{len}(q)} w_{q_i} \cdot \min_{j \in [\text{len}(d)]} \|E_q(q_i) - E_d(d_j)\|_2 \tag{2}$$

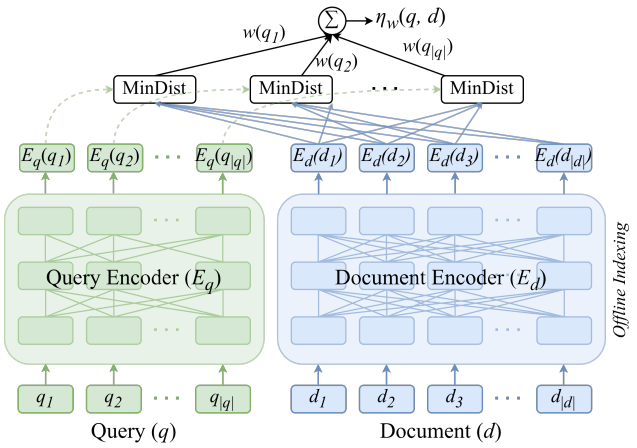

Figure 2: ColBERT Architecture with WeightedChamfer

where w_{q_i} *are the weights associated with token* q_i*.*

Note that Chamfer distance is a special case of Weighted Chamfer where all the token weights are set to 1.

We explore two different weighting techniques a) zero-shot setting (unsupervised) with no access to any relevance data[1], and b) few-shot setting (supervised) with limited access to relevance data.

Zero-shot Weighted Chamfer In this setting, we choose the weights $w \in \mathbb{R}^T$ using statistics derived solely from the document corpus $\mathcal{D}$. Specifically, we use the widely used Inverse Document Frequency (IDF) as the weight for each token t. Formally, the IDF weight of a token t is defined as:

$$\text{IDF}(t) = \log\left(\frac{N - n(t) + 0.5}{n(t) + 0.5} + 1\right) \tag{3}$$

where N is the size of the corpus $\mathcal{D}$ and $n(t)$ is the number of documents that contain the token[2] t. Since these token weights are computed solely from the token frequency in the document corpus, they do not require any labeled relevance data. Moreover, the computation of IDF weights has linear complexity in the total number of tokens in the corpus and can be further accelerated by sub-sampling documents, resulting in an approximate but efficient estimation of the weights.

Few-shot Weighted Chamfer In this setting, we prescribe a technique to learn the set of token weights using relevance data. We consider the widely used ranking loss, defined in eq. (4), which tries to reduce the distance to the positive document $d^+ \in \mathcal{D}_q^+$ relative to the negative documents $\mathcal{D}_q^-$ for a query q.

[1]Relevance data consists of query-document pairs, each annotated with a label indicating their relevance.

[2]For the tokens t' that do not appear in the corpus (but seen in the queries, during inference), we assign weights to 0. For the special tokens (viz. `[MASK]`, `[PAD]`, `[QUERY]`, `[DOCUMENT]`, `[CLS]` tokens), we set weights to 0 or 1 based on the performance on the validation slice.

Definition 3 (Cross-Entropy Loss)**.** *Given a query q and the set of positive (relevant) documents $\mathcal{D}_q^+ \subset \mathcal{D}$ and negative documents $\mathcal{D}_q^- \subseteq \mathcal{D} \backslash \mathcal{D}_q^+$ let*

$$\mathrm{CE}_w(q;\, \mathcal{D}_q^+, \mathcal{D}_q^-) = \tag{4}$$
$$-\sum_{d \in \mathcal{D}_q^+} \log \left(\frac{\exp\left(-\eta_w(q,d)\right)}{\sum\limits_{d' \in \{\mathcal{D}_q^+ \cup \mathcal{D}_q^-\}} \exp\left(-\eta_w(q,d')\right)} \right).$$

The cross-entropy loss (ranking-loss) is defined as,

$$\sum_q \mathrm{CE}_w(q;\, \mathcal{D}_q^+, \mathcal{D}_q^-)$$

We remark here that the cross-entropy loss defined above is convex. We formally state this result next.

Lemma 4. *For any query q and sets $\mathcal{D}_q^+$ and $\mathcal{D}_q^-$, the function $\mathrm{CE}_w(q;\, \mathcal{D}_q^+, \mathcal{D}_q^-)$ is convex in variable w. Consequently, the ranking-loss function $\sum_q \mathrm{CE}_w(q;\, \mathcal{D}_q^+, \mathcal{D}_q^-)$ is also convex in w.*

Since most datasets lack explicit annotations for irrelevant documents, the negative set is often very large. A common practice is to sample a subset of negatives (typically mined using a retriever) to make training computationally feasible. In our experiments, we found that combining losses (eq. (5)) computed over two sets of negatives, $\Lambda_q^{(1)} \subseteq \Lambda_q^{(2)} \subseteq \mathcal{D}_q^-$, leads to improved performance. We believe that this approach may also be beneficial in other contrastive learning applications.

$$\sum_q \mathcal{L}_w(q), \text{ where} \tag{5}$$
$$\mathcal{L}_w(q) = \alpha \cdot \mathrm{CE}_w(q;\, \mathcal{D}_q^+, \Lambda_q^{(1)}) + (1-\alpha) \cdot \mathrm{CE}_w(q;\, \mathcal{D}_q^+, \Lambda_q^{(2)})$$

By Lemma 4, it follows immediately that the loss function defined above is convex. We formalize this result next.

Corollary 5. *For any query q and sets $\mathcal{D}_q^+$, $\Lambda_q^{(1)}$ and $\Lambda_q^{(2)}$, the function $\mathcal{L}_w(q)$ is a convex in variable w. Consequently, the function $\sum_q \mathcal{L}_w(q)$ is also convex in w.*

In our experiments, we find that instead of optimizing the loss function for a fixed set of negatives per query, choosing the closest (i.e., lowest-distance) negatives based on the current weights in an iterative fashion (to further encourage the model by forcing it to assign lower distances to positive documents by focusing on the hardest negatives) works best. Hence, in practice, we don't optimize the convex function $\sum_q \mathrm{CE}_w(q;\, \mathcal{D}_q^+, \mathcal{D}_q^-)$ but take gradient steps according to a series of convex functions which iteratively change according to the set of negatives. We provide complete details on our empirical performance in the results section.

In addition to our empirical results, we also provide theoretical bounds that characterize the limited amount of relevance data required to learn the weights. The theoretical models that we study don't exactly correspond to the empirical setting we are interested in. Instead they serve as reference points for understanding the sample complexity for learning token weights. Our theoretical contributions are twofold.

First, we study the problem of recovering the true hidden weights in a setting where query–document pairs are sampled i.i.d. from an underlying distribution, and their relevance score is given by the Weighted Chamfer score for some unknown true weights. In this setting, using the matrix Chernoff bound, we establish an upper bound on the sample complexity required to precisely recover the true hidden weights. This result is formal stated in the following theorem.

Theorem 6. *Given query-document pairs (q,d) sampled i.i.d from a distribution $\mathfrak{D}$ and access to the weighted chamfer score from an underlying hidden weights. Let $A = \mathbb{E}_{(q,d)\sim\mathfrak{D}}\left[\frac{1}{l} x_{(q,d)} x_{(q,d)}^\top\right]$, where $l = \mathrm{len}(q)$ and $x_{(q,d)}(t) = \max\limits_{t' \in \mathrm{len}(d)} E_q(q_t)^\top E_d(d_{t'})$ if $t \in q$ and 0 otherwise, then there exists an algorithm that takes*

$$n \geq \Omega\left(\frac{1}{\lambda_{\min}(A)} \log\left(\frac{T}{\delta}\right)\right)$$

samples and with probability at least $1-\delta$, learns the true hidden weights.

Second, we relax the strong assumption of having access to Weighted Chamfer scores and instead assume access only to binary relevance labels for query–document pairs. We consider a hypothesis class consisting of threshold classifiers based on the Weighted Chamfer score, where the threshold τ is fixed and shared across all queries[3]. Under this setting, we derive generalization bounds by reducing the problem of learning the weights to that of learning linear threshold functions. Using standard statistical learning theory techniques involving the VC dimension, we establish a bound on the sample complexity required to learn a Weighted Chamfer score–based classifier from this hypothesis class whose expected risk is within an additive ϵ of the optimum achievable by any function in the class. We defer the formal theoretical statements and their proofs to the appendix[4].

Results

We evaluate the performance of Weighted Chamfer in the re-ranking setting. For each query q, we employ BM25 (with Lucene parameters $k_1 = 1.5$ and $b = 0.75$) as the retriever, followed by Weighted Chamfer for re-ranking a small set of k' documents ($k' = 1000$). Since k' is small, we exhaustively score all the documents against the query q.

We use the ColBERTv2 checkpoint, trained on the MS MARCO Passage Ranking dataset (Bajaj et al. 2018), with its parameters frozen for both zero-shot and few-shot evaluation settings. To assess performance, we evaluate the Weighted Chamfer and naive Chamfer[5] methods on a subset of the

[3] In the ranking setting, each query could, in principle, have its own relevance threshold.

[4] Our theoretical results do not exactly model our empirical studies but instead provide a guiding direction for understanding their behavior.

[5] Note that ColBERTv2 uses Chamfer distance.

BEIR benchmark (Thakur et al. 2021) (13 datasets not including MS MARCO), excluding datasets that are not publicly accessible. Re-ranking quality is primarily reported using Recall@10, MRR@10 and nDCG@10. Since ColBERTv2 is trained exclusively on MS MARCO, the results, on the other BEIR datasets, reflect out-of-domain performance.

Dataset	α	$\lvert\Lambda_1\rvert$	$\lvert\Lambda_2\rvert$
ARGUANA	0	-	500
CLIMATE-FEVER	0	-	1000
DBPEDIA-ENTITY	0.25	5	100
FEVER	0.1	10	100
FIQA	0.1	10	100
HOTPOTQA	0.1	100	1000
MSMARCODOCS	0.1	100	1000
NQ	0.75	5	100
QUORA	0.1	10	100
SCIDOCS	0.5	100	250
SCIFACT	0.1	50	250
TREC-COVID	0	-	100
WEBIS-TOUCHE2020	0	-	100

Table 1: Hyper-parameters used for Few-shot training of Weighted Chamfer

We use BM25 as retriever due to its low latency and demonstrate the performance of Weighted Chamfer over Chamfer as a re-ranker. We believe the advantages presented here would generalize to other retrievers as well.

Training Details As discussed, we learn only the token weights, while keeping the underlying ColBERTv2 model frozen. We use the Adam optimizer (Kingma and Ba 2017), and across all datasets, we use a learning rate of 10^{-4} and cosine scheduler to lower the learning rate to 10^{-8} over 100 iterations. The token weights are initialized uniformly, and after each iteration, we constrain their sum to 1.

Due to the absence of predefined training splits[6] in some BEIR datasets, we adopt a random train–validation–test split to evaluate the effectiveness of our approach. We identify the optimal hyper-parameters, specifically, the sizes of the negative sets per query ($|\Lambda_1|$, $|\Lambda_2|$) and the interpolation factor $\alpha \in [0, 1]$ – based on the performance on the validation split. The learned weights are then evaluated on the validation set, and we select the best-performing weights between IDF and learnt. If the learnt weights perform better, we retrain the model on the union of the training and validation sets to further enhance performance. We remark here that not all tokens are 'seen' during the training procedure. To further improve performance, we assign IDF-based weights to tokens that are unseen in the training slice. While doing so, we ensure that the total weight assigned to seen tokens remains the same as in the IDF-weighted case. Essentially, the training procedure only re-weights the relative importance among the seen tokens, while assigning IDF-based weights to the unseen ones.

Table 1 enumerates the choice of hyper-parameters used. Based on empirical results, when optimizing for Recall@k, we recommend setting the sizes of Λ_1 and Λ_2 to be slightly larger than k and $10k$, respectively, and $\alpha = 0.1$ in most cases. Through our experiments, we observe that datasets with limited supervision are particularly sensitive to the choice of hyper-parameters. Only in those cases, we do a hyper-parameter search over and choose the optimal values, but other wise resort to $|\Lambda_1| = 10$ and $|\Lambda_2| = 100$ with $\alpha = 0.1$.

Weighted Chamfer Table 2 and Table 3 displays the Recall@10, MRR@10 and nDCG@10 of Chamfer and Weighted Chamfer post retrieval using BM25 and using the end-to-end ColBERTv2 pipeline respectively. In the zero-shot setting, the weights are computed solely from the document corpus and therefore do not require any relevance data. This is particularly useful in scenarios where human-labeled data is scarce or unavailable. Moreover, the static nature of these weights ensures low computational overhead. For datasets HOTPOTQA, MS MARCODOCS, SCIFACT, and WEBIS-TOUCHE2020, the reported results use zero weights for special tokens. For all the remaining datasets, a weight of 1 is assigned to special tokens (prior to normalization).

In the few-shot setting, with our prescribed training routine, we see a significant gain over the zero-shot setting, even when trained on a very small amount of relevance data[7]. This demonstrates that Weighted Chamfer can effectively utilize limited supervision to substantially improve re-ranking quality. Moreover, since it maintains only a few additional parameters (per dataset) while sharing the underlying model, this approach is storage-friendly compared to maintaining multiple fine-tuned models. Our experiments suggest that Weighted Chamfer performs particularly well on datasets where the relevance signals are clear but differ significantly from those seen during training. This indicates that token weights are effective in adapting to dataset-specific importance.

Discussion

In this work, we introduced Weighted Chamfer, an enhancement to the ColBERT re-ranking framework. By integrating token-level importance weights into the late-interaction mechanism, we improved retrieval performance without altering the core model architecture.

In both zero-shot and few-shot settings, our proposed methods enhance re-ranking performance. In the zero-shot setting, weights are derived from corpus-level statistics without requiring any relevance supervision, while in the few-shot setting, our training routine learns optimal weights using limited relevance data. Empirically, our few-shot approach consistently outperforms the Chamfer baseline across all evaluation metrics; Recall@10, MRR@10, and nDCG@10. Importantly, Weighted Chamfer is simple to deploy within existing re-ranking pipelines, making it particularly suitable

[6]For datasets that include a train-dev split, we randomly sample from the training data to create train-validation subset. For the remaining datasets, we construct train-validation-dev splits by randomly sampling from the original dev set.

[7]Note that TREC-COVID and WEBIS-TOUCHE2020 contain fewer than 50 queries.

Dataset	Recall@10				MRR@10				nDCG@10			
	Chamfer	Weighted Chamfer		%Δ	Chamfer	Weighted Chamfer		%Δ	Chamfer	Weighted Chamfer		%Δ
		Zero-Shot	Few-Shot			Zero-Shot	Few-Shot			Zero-Shot	Few-Shot	
Passage Retrieval												
MS MARCODOCS	0.6984	0.7205	0.7205	3.16%	0.8926	0.9049	0.9049	1.38%	0.7285	0.7504	0.7504	3.01%
Average Performance				**+3.16%**				**+1.38%**				**+3.01%**
Question Answering												
NQ	0.7622	0.7628	0.7806	2.41%	0.5296	0.5172	0.5351	1.04%	0.5707	0.5617	0.5787	1.40%
HOTPOTQA	0.6963	0.6998	0.7146	2.63%	0.8922	0.8872	0.9019	1.09%	0.6954	0.6952	0.7113	2.29%
FIQA	0.4412	0.4367	0.4462	1.13%	0.4334	0.4267	0.4375	0.95%	0.3630	0.3589	0.3672	1.16%
Average Performance				**+2.06%**				**+1.03%**				**+1.62%**
Entity Retrieval												
DBPEDIA-ENTITY	0.6716	0.6881	0.6881	2.46%	0.9320	0.9234(0	0.9320	0.00%	0.5417	0.5383	0.5417	0.00%
Average Performance				**+2.46%**				**0.00%**				**0.00%**
Argument Retrieval												
ARGUANA	0.6600	0.6600	0.6833	3.53%	0.2162	0.2131	0.2269	4.95%	0.3234	0.3206	0.3372	4.27%
WEBIS-TOUCHE2020	0.3327	0.3429	0.3429	3.07%	0.6174	0.5407	0.6174	0.00%	0.3591	0.3519	0.3577	0.39%
Average Performance				**+3.29%**				**+2.46%**				**+2.33%**
Duplicate Question Retrieval												
QUORA	0.9319	0.9384	0.9384	0.70%	0.8405	0.8502	0.8502	1.15%	0.8488	0.8576	0.8576	1.04%
Average Performance				**+0.70%**				**+1.15%**				**+1.04%**
Citation Prediction												
SCIDOCS	0.1575	0.1575	0.1655	5.08%	0.2860	0.3041	0.3041	6.33%	0.1558	0.1597	0.1635	4.94%
Average Performance				**+5.08%**				**+6.33%**				**+4.94%**
Fact Checking												
FEVER	0.9128	0.9128	0.9168	0.44%	0.8507	0.8466	0.8590	0.98%	0.8416	0.8389	0.8485	0.82%
CLIMATE-FEVER	0.2804	0.2886	0.3204	14.27%	0.3044	0.2959	0.3385	11.20%	0.2391	0.2359	0.2675	11.88%
SCIFACT	0.8136	0.8336	0.8652	6.34%	0.6714	0.6900	0.7194	7.15%	0.6994	0.7202	0.7485	7.02%
Average Performance				**+7.02%**				**+6.44%**				**+6.57%**
Bio-Medical IR												
TREC-COVID	0.9440	0.9660	0.9660	2.33%	0.9767	1.0000	1.0000	2.39%	0.7032	0.7169	0.7169	1.95%
Average Performance				**+2.33%**				**+2.39%**				**+1.95%**
Average Performance				**+3.66%**				**+2.97%**				**+3.01%**

Table 2: Recall@10, MRR@10, and nDCG@10 are reported after re-ranking top-1000 retrieved by BM25 on the BEIR benchmark.

Dataset	Recall@10				MRR@10				nDCG@10			
	Chamfer	Weighted Chamfer		%Δ	Chamfer	Weighted Chamfer		%Δ	Chamfer	Weighted Chamfer		%Δ
		Zero-Shot	Few-Shot			Zero-Shot	Few-Shot			Zero-Shot	Few-Shot	
Passage Retrieval												
MS MARCODOCS	0.3486	0.3512	0.3450	0.75%	0.8992	0.8992	0.9054	0.69%	0.4217	0.4236	0.4292	1.78%
Average Performance				**+0.75%**				**+0.69%**				**+1.78%**
Question Answering												
NQ	0.7989	0.7861	0.7989	0.00%	0.5460	0.5324	0.5460	0.00%	0.5917	0.5782	0.5917	0.00%
HOTPOTQA	0.6938	0.7040	0.7040	1.47%	0.8941	0.9009	0.9009	0.76%	0.6937	0.7043	0.7043	1.53%
FIQA	0.4312	0.4283	0.4356	1.02%	0.4306	0.4232	0.4306	0.00%	0.3588	0.3551	0.3588	0.00%
Average Performance				**+0.83%**				**+0.25%**				**+0.51%**
Entity Retrieval												
DBPEDIA-ENTITY	0.6716	0.6881	0.6761	2.46%	0.9320	0.9234	0.9320	0.00%	0.5417	0.5383	0.5417	0.00%
Average Performance				**+2.46%**				**0.00%**				**0.08%**
Argument Retrieval												
ARGUANA	0.6567	0.6567	0.6967	6.09%	0.2152	0.2112	0.2361	9.71%	0.3211	0.3183	0.3479	8.35%
WEBIS-TOUCHE2020	0.3306	0.3327	0.3327	0.64%	0.6163	0.5850	0.6183	0.32%	0.3570	0.3462	0.3573	0.08%
Average Performance				**+3.36%**				**+5.02%**				**+4.22%**
Duplicate Question Retrieval												
QUORA	0.9321	0.9381	0.9381	0.64%	0.8406	0.8504	0.8504	1.17%	0.8488	0.8576	0.8576	1.04%
Average Performance				**+0.64%**				**+1.17%**				**+1.04%**
Citation Prediction												
SCIDOCS	0.1575	0.1575	0.1655	5.08%	0.2860	0.3041	0.2962	6.33%	0.1558	0.1597	0.1635	4.94%
Average Performance				**+5.08%**				**+6.33%**				**+4.94%**
Fact Checking												
FEVER	0.9142	0.9120	0.9142	0.00%	0.8561	0.8515	0.8561	0.00%	0.8454	0.8418	0.8454	0.00%
CLIMATE-FEVER	0.2728	0.2929	0.2929	7.37%	0.2945	0.2890	0.2945	0.00%	0.2337	0.2328	0.2337	0.00%
SCIFACT	0.8036	0.8202	0.8522	6.05%	0.6696	0.6849	0.7145	6.71%	0.6957	0.7133	0.7401	6.38%
Average Performance				**+4.47%**				**+2.24%**				**+2.13%**
Bio-Medical IR												
TREC-COVID	0.9200	0.9600	0.9600	4.35%	0.9800	1.0000	1.0000	2.04%	0.9272	0.9651	0.9651	4.09%
Average Performance				**+4.35%**				**+2.04%**				**+4.09%**
Average Performance				**+3.04%**				**+2.13%**				**+2.24%**

Table 3: Recall@10, MRR@10, and nDCG@10 are reported after end-to-end retrieval using ColBERTv2 on the BEIR benchmark.

for scenarios where relevance data is limited or supervision is costly.

Our findings indicate that modifying only the late-interaction mechanism can substantially enhance the expressiveness of the underlying embedding model, opening the door to more sophisticated distance functions. For instance, the distance function can be extended to promote matching between co-occurring query and document tokens, capturing multi-token phrases (e.g., "white house") while preserving lexical structure during retrieval. This direction is particularly promising, as it can be seamlessly integrated into existing re-ranking pipelines, even in low-supervision or resource-constrained scenarios.

References

Bajaj, P.; Campos, D.; Craswell, N.; Deng, L.; Gao, J.; Liu, X.; Majumder, R.; McNamara, A.; Mitra, B.; Nguyen, T.; Rosenberg, M.; Song, X.; Stoica, A.; Tiwary, S.; and Wang, T. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv:1611.09268.

Dai, Z.; Zhao, V. Y.; Ma, J.; Luan, Y.; Ni, J.; Lu, J.; Bakalov, A.; Guu, K.; Hall, K. B.; and Chang, M.-W. 2022. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Dhulipala, L.; Hadian, M.; Jayaram, R.; Lee, J.; and Mirrokni, V. 2024. MUVERA: Multi-Vector Retrieval via Fixed Dimensional Encodings. arXiv:2405.19504.

Formal, T.; Lassance, C.; Piwowarski, B.; and Clinchant, S. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. arXiv:2109.10086.

Formal, T.; Piwowarski, B.; and Clinchant, S. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. arXiv:2107.05720.

Gao, L.; Dai, Z.; and Callan, J. 2021. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. *arXiv preprint arXiv:2104.07186*.

Humeau, S.; Shuster, K.; Lachaux, M.-A.; and Weston, J. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*.

Johnson, J.; Douze, M.; and Jégou, H. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3): 535–547.

Karpukhin, V.; Oğuz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and tau Yih, W. 2020. Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2004.04906.

Khattab, O.; and Zaharia, M. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. arXiv:2004.12832.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Lee, J.; Dai, Z.; Duddu, S. M. K.; Lei, T.; Naim, I.; Chang, M.-W.; and Zhao, V. Y. 2024. Rethinking the Role of Token Retrieval in Multi-Vector Retrieval. arXiv:2304.01982.

MacAvaney, S.; Mallia, A.; and Tonellotto, N. 2025. Efficient Constant-Space Multi-Vector Retrieval. arXiv:2504.01818.

MacAvaney, S.; Yates, A.; Cohan, A.; and Goharian, N. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 1101–1104.

Macdonald, C.; Tonellotto, N.; and Ounis, I. 2021. On single and multiple representations in dense passage retrieval. *arXiv preprint arXiv:2108.06279*.

Malkov, Y. A.; and Yashunin, D. A. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4): 824–836.

Nogueira, R.; and Cho, K. 2020. Passage Re-ranking with BERT. arXiv:1901.04085.

Oyama, M.; Yokoi, S.; and Shimodaira, H. 2022. Norm of word embedding encodes information gain. *arXiv preprint arXiv:2212.09663*.

Qian, Y.; Lee, J.; Duddu, S. M. K.; Dai, Z.; Brahma, S.; Naim, I.; Lei, T.; and Zhao, V. Y. 2022. Multi-vector retrieval as sparse alignment. *arXiv preprint arXiv:2211.01267*.

Qu, Y.; Ding, Y.; Liu, J.; Liu, K.; Ren, R.; Zhao, W. X.; Dong, D.; Wu, H.; and Wang, H. 2020. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*.

Robertson, S.; Walker, S.; Jones, S.; Hancock-Beaulieu, M. M.; and Gatford, M. 1995. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, 109–126. Gaithersburg, MD: NIST.

Santhanam, K.; Khattab, O.; Potts, C.; and Zaharia, M. 2022a. PLAID: An Efficient Engine for Late Interaction Retrieval. arXiv:2205.09707.

Santhanam, K.; Khattab, O.; Saad-Falcon, J.; Potts, C.; and Zaharia, M. 2022b. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. arXiv:2112.01488.

Subramanya, S. J. S.; Devvrit; Kadekodi, R.; Krishnaswamy, R.; and Simhadri, H. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *NeurIPS 2019*.

Thakur, N.; Reimers, N.; Rücklé, A.; Srivastava, A.; and Gurevych, I. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663.

Wang, L.; Yang, N.; Huang, X.; Yang, L.; Majumder, R.; and Wei, F. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.