Revisiting BM25 Feedback Models using HyDE

Nour Jedidi
University of Waterloo
Waterloo, ON, Canada
njedidi@uwaterloo.ca

Jimmy Lin
University of Waterloo
Waterloo, ON, Canada
jimmylin@uwaterloo.ca

Abstract

Recent approaches that leverage large language models (LLMs) for pseudo-relevance feedback (PRF) have generally not utilized well-established feedback models like Rocchio and RM3 when expanding queries for BM25. Instead, they often opt for a simple string concatenation of the query and LLM-generated expansion content. In this paper, we revisit and systematically evaluate traditional BM25 feedback models in the context of HyDE, a popular method that enriches query representations using LLM-generated hypothetical answer documents. Experiments demonstrate a mutually beneficial relationship between BM25 feedback models and HyDE. Feedback models are more effective when provided LLM-generated documents versus top-ranked documents retrieved by BM25, while HyDE benefits from feedback algorithms that select and weight expansion terms. In fact, by incorporating BM25 feedback models within the HyDE setup, we can further narrow the gap between BM25 and a strong "single-shot" dense retriever, without incurring the costs associated with such embedding-based retrieval methods. Ultimately, our results demonstrate that traditional BM25 feedback models can play an important role within modern LLM PRF methods and provide a simple approach to further enhance the accuracy of HyDE. Our code is available at https://github.com/nourj98/hyde-feedback.

CCS Concepts

• **Information systems** → **Retrieval models and ranking**.

Keywords

Pseudo-Relevance Feedback, Query Representation, LLMs

ACM Reference Format:
Nour Jedidi and Jimmy Lin. 2026. Revisiting BM25 Feedback Models using HyDE. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3805712.3809889

1 Introduction

Hypothetical Document Embeddings (HyDE) [10] represent a simple, yet effective method to improve the accuracy of "bag of words" retrieval methods like BM25 by utilizing large language models (LLMs) to expand queries with hypothetical answer documents. The core concept underlying HyDE is that the knowledge contained within the parameters of LLMs can be leveraged as a form of *relevance feedback*, helping bridge the vocabulary gap between query representations and those of its corresponding relevant documents. This builds upon decades of research in information retrieval (IR) on *pseudo*-relevance feedback (PRF), which traditionally assumed top-ranked documents from a first-stage retriever (like BM25) contain useful expansion terms which can be used for enriching the query representation. Other than obvious distinctions that are simply due to advancements in modeling, the core difference between traditional PRF techniques and HyDE is primarily the *source* of the feedback terms: traditional PRF methods extract feedback terms from top-ranked retrieved documents, whereas HyDE's feedback terms are generated directly by the LLM.

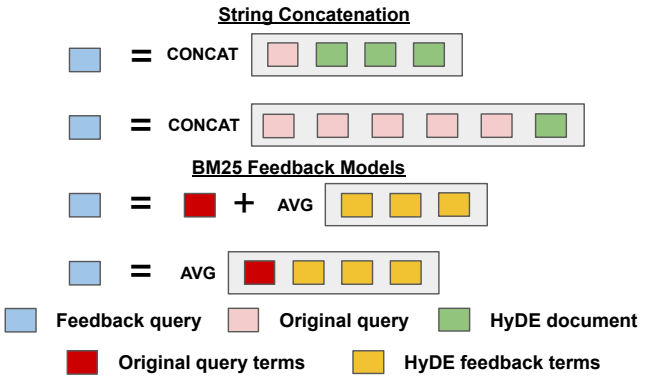

Figure 1: Conceptual illustration of HyDE with feedback based on string concatenation (naive concatenation and Query2Doc-style), contrasted against HyDE using BM25 feedback models (Rocchio and average vector).

In addition, despite this seemingly small difference, another distinction surprisingly lies in the exact mechanism used to update the query representation for BM25. Traditional PRF techniques typically run two phases after an initial set of candidate feedback documents are returned [4]. The first phase consists of selecting the terms to be included in the expanded query, q_{new}. The second phase consists of properly weighting these selected terms, along with those in the original query, for q_{new}. On the other hand, HyDE and its variants [18, 23] have generally opted for string concatenation strategies that directly feed the LLM-generated text and the original query into BM25 with minimal processing.

These differences raise two important questions: Can traditional BM25 feedback models benefit from HyDE? And, conversely, can HyDE benefit from such feedback models?

To answer the first question, we revisit feedback models for selecting and weighting expansion terms generated by HyDE, focusing primarily on those which remain widely used today, such as Rocchio [15] and RM3 [1]. Across 14 retrieval datasets, our results demonstrate that simply substituting top-ranked BM25-retrieved

SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3809889

documents with HyDE-generated documents consistently improves the effectiveness of BM25 feedback models.

To answer the second question, we compare HyDE using BM25 feedback models against the standard string concatenation strategy. Our results demonstrate that leveraging BM25 feedback models improves the effectiveness of HyDE. For example, by utilizing the Rocchio feedback algorithm, HyDE's effectiveness can improve by as much as 1.4 points (4.2%) on average versus a top-performing string concatenation approach. The improvement is even more pronounced on the diverse retrieval tasks in BEIR [16], with BM25 feedback models improving HyDE's effectiveness by as much as 2.2 points (6%).

Our contributions are as follows. First, we show that BM25 feedback models can benefit from recent advancements in leveraging LLMs as a tool for relevance feedback. Second, we run a comprehensive evaluation of HyDE with different feedback mechanisms. While recent work has primarily examined how to improve the expansions the LLM produces, relatively little effort has been devoted to evaluating different ways LLM-generated feedback can be leveraged to update the query representation for BM25. To the best of our knowledge, our study represents the first to properly compare different feedback mechanisms, spanning new and traditional techniques, in the context of LLM-generated feedback. Finally, we demonstrate that the vast literature on BM25 feedback models, which has been mostly overlooked by newer LLM PRF approaches, remains highly relevant and can provide substantial improvements in the effectiveness of HyDE.

2 Methods

Given a query q, HyDE first prompts an LLM to generate a hypothetical answer document and samples N variations, $e = \{\hat{d}_1, \ldots, \hat{d}_N\}$. The terms in these hypothetical documents form the basis of what is subsequently used for generating q_{new}. In this section, we first describe how traditional BM25 feedback models can be leveraged in the HyDE setup. We then overview different string concatenation strategies used in the LLM PRF literature that we compare BM25 feedback models against.

2.1 BM25 Feedback Models

BM25 feedback models typically undergo two phases before generating the updated query q_{new}. The first phase selects the set of feedback terms from e for inclusion in q_{new}. Then, in the second phase, weights are assigned to the selected terms to form the updated query representation.

2.1.1 Selecting Feedback Terms. Once HyDE has generated e, we generate a term-frequency vector, $f(\hat{d}_i)$, for each of the feedback documents in e. We then directly follow the process implemented in Anserini [22], which, at a high level, first filters out common corpus terms from each feedback vector, keeping only the terms which occur in less than 10% of the corpus documents. From these filtered vectors, $\tilde{f}(\hat{d}_i)$, candidate expansion terms are then ranked by the sum of their normalized term frequencies across each of $\tilde{f}(\hat{d}_i)$ and, from this, $\tilde{f}(\hat{d}_i)$ is further pruned to only include the top-k ranked feedback terms [4].[1]

[1]The specifics of this selection procedure can be different based on the feedback model.

Now that we have a set of feedback terms for each of the HyDE-generated documents, we next discuss the different feedback models considered in our evaluation for generating q_{new}.

2.1.2 Average Vector Feedback. The first feedback model we consider is the average vector feedback, which is a reformulation of HyDE's proposed query update – originally designed for dense vectors – to the bag-of-words vector space. The average vector update takes a simple average of the query representation and the representations of the hypothetical answer documents. In particular, it considers the query as an additional feedback document in e, i.e., $e_{\text{HyDE}} = \{q, \hat{d}_1, \ldots, \hat{d}_N\}$. Adapted to the bag-of-words space, the weight of a term $w_{t,q_{\text{new}}}$ in q_{new} is computed as follows:

$$w_{t,q_{\text{new}}} = \frac{1}{N+1} \sum_{d_i \in e_{\text{HyDE}}} \tilde{f}(d_i)[t] \tag{1}$$

Here, $\tilde{f}(d_i)[t]$ is the normalized frequency of t in $\tilde{f}(d_i)$. Note that $\tilde{f}$ is computed as described in Section 2.1.1, however, for q, we do not filter out common corpus terms. At a high level, this represents a term-weight as its average across the query and HyDE-generated documents.

2.1.3 Traditional Feedback Models: Rocchio and RM3. We next consider Rocchio's algorithm, which is a more generalized version of the HyDE vector update that instead controls the weight of the query vector and feedback vectors using parameters, α and β, rather than an equal weighting:

$$w_{t,q_{\text{new}}} = \alpha \cdot f(q)[t] + \frac{\beta}{N} \sum_{\hat{d}_i \in e} \tilde{f}(\hat{d}_i)[t] \tag{2}$$

Here, $f(q)$ is the normalized term-frequency vector of q, and $f(q)[t]$ is the normalized frequency of t in $f(q)$.

Lastly, we consider the RM3 feedback approach [1]. Given feedback documents, RM3 estimates a relevance model that computes the probability that a term is observed in a document relevant to the query [13]. In our case, this probability is estimated based on the hypothetical documents in e. This relevance model is then linearly interpolated with the probability of the term given the original query. Using our above framework, the term-weight would be computed as follows:

$$w_{t,q_{\text{new}}} = \lambda \cdot P(t|q) + (1-\lambda) \sum_{\hat{d}_i \in e} P(t|\hat{d}_i) \tag{3}$$

The exact mechanism for how to compute $P(t|\hat{d}_i)$ is beyond the scope of this paper. We refer readers to Abdul-Jaleel et al. [1] and to the RM3 implementation in Anserini [22], which we adopted in our experiments.

Once the weights for each feedback term in q_{new} have been computed, we construct custom Lucene queries for Anserini in which each term's boost is set to $w_{t,q_{\text{new}}}$.

2.2 String Concatenation Baselines

While we aim to understand if HyDE can benefit BM25 feedback models, it is also critical to evaluate whether feedback models can improve HyDE itself, relative to existing approaches which integrate LLM-generated hypothetical documents with BM25.

To do so, we compare BM25 feedback models against three string concatenation strategies to generate q_{new}. It is worth noting that, unlike the feedback models described, string concatenation approaches primarily focus on boosting the query term-weights, while leaving the generated hypothetical documents "as-is".

2.2.1 Query2Doc. Our first baseline is Query2Doc [18], which was one of the first to propose the use of LLM query expansion via the string concatenation operation. To generate q_{new}, Query2Doc repeats the query a constant r times before appending it to a generated hypothetical document $\hat{d}$, i.e., $q_{\text{new}} = \text{Concat}(q \times r, \hat{d})$. We note that one crucial difference between Query2Doc and HyDE, however, is that Query2Doc only generates one hypothetical document rather than N. As we will detail in Section 3, to control for any differences in effectiveness that may arise from HyDE generating N hypothetical documents – versus Query2Doc only generating one – we limit the number of top-k feedback terms used by HyDE with BM25 feedback models to match the length of Query2Doc's generated hypothetical document. In turn, this comparison allows us to directly study how implementing HyDE with feedback models compares to a strong string concatenation baseline under an approximately equal budget of feedback terms.

Our next two baselines directly leverage all N samples of the HyDE-generated documents in e.

2.2.2 Naive Concat. Our second baseline is a naive concatenation approach which simply concatenates the query to the HyDE-generated documents, i.e., $q_{\text{new}} = \text{Concat}\,(q, e)$. We note that this, by itself, would return similar rankings by BM25 to the average vector feedback described in 2.1.2, assuming no selection of feedback terms is performed (i.e., if we bypass the term-selection step detailed in 2.1.1).[2] As such, this feedback method allows us to evaluate how useful it is to select feedback terms from HyDE-generated documents.

2.2.3 MuGI Concat. Our last baseline is based on the adaptive query reweighting approach proposed in MuGI [23], which we refer to as *MuGI Concat*. MuGI Concat serves as our upper bound on string concatenation effectiveness, as it leverages all N of HyDE's generated documents in its updated query and leverages an adaptive query repeat, rather than a single (e.g., Naive Concat) or constant repeat (e.g., Query2Doc). MuGI adaptively repeats the query γ times before concatenating it with the HyDE-generated documents in e. To compute γ, MuGI uses the following equation:

$$\gamma = \frac{\text{len}(\hat{d}_1) + \text{len}(\hat{d}_2) + \cdots + \text{len}(\hat{d}_N)}{\text{len}(q) \cdot \phi} \tag{4}$$

As such, $q_{\text{new}} = \text{Concat}(q \times \gamma, e)$.

3 Experimental Setup

The aim of our experiments is to (1) evaluate whether BM25 feedback models benefit from leveraging HyDE documents and (2) compare HyDE with BM25 feedback models against the string concatenation baselines described in Section 2.2.

[2]This assumes that BM25 creates the query vector using a term-frequency vector, as described in Ge et al. [11], which would only cause BM25 similarity scores to differ by a multiplicative constant due to using a sum rather than a mean term-weight.

Our HyDE implementation follows the approach described in Gao et al. [10]. We sample eight hypothetical documents of up to 512 tokens. When implementing HyDE, we consider three LLMs: Qwen2.5-7B-Instruct [21], Qwen3-14B [20], and gpt-oss-20b [2].

For the average vector, Rocchio, and RM3 feedback models, we stick to the default hyperparameters used in the literature [14]. Namely, for Rocchio we set $\alpha = 1$ and $\beta = 0.75$. For RM3, we set the query weight λ to 0.5. All feedback models select up to $k = 128$ feedback terms to match the number of feedback terms used by Query2Doc [18], one of our string concatenation baselines. We explore varying the number of feedback terms and generated hypothetical documents in Section 5.

To isolate the effect of HyDE on BM25 feedback models, we first compare against the average vector, Rocchio, and RM3 feedback models when using top-ranked BM25-retrieved documents rather than HyDE documents. We directly leverage the same hyperparameters described above for Rocchio and RM3, and use eight BM25 feedback documents to match the number of HyDE-generated documents. We also consider BM25 without any query expansion and include BGE-base-en-v1.5 [19] as a strong dense retrieval baseline.

To evaluate whether HyDE benefits from BM25 feedback models, we compare HyDE with feedback models against the Query2Doc, Naive Concat, and MuGI Concat baselines described in Section 2.2. For Query2Doc, we follow the implementation in Wang et al. [18]: we repeat the query a constant of five times and generate a single hypothetical document of up to 128 tokens. For MuGI Concat, we set $\phi = 5$, following [23].

We evaluate all methods on five web-search datasets from MS MARCO v1 (TREC DL19 and TREC DL20) [3, 7, 8], and from MS MARCO v2 (TREC DL21, TREC DL22, and TREC DL23) [5, 6, 9]. We also evaluate on nine retrieval datasets from BEIR [16]. The tasks include news retrieval (TREC-News, Robust04), financial question answering (FiQA), entity retrieval (DBpedia), biomedical IR (TREC-COVID, NFCorpus), fact checking (SciFact), citation prediction (SCIDOCS), and argument retrieval (ArguAna). For metrics, we report Recall@20 to reflect a common use of HyDE within retrieval-augmented generation systems, e.g., top-20 documents retrieved are fed into the LLM for answer generation [17].

All retrieval experiments were conducted using the BM25 implementation from Anserini [22] via the Pyserini [14] Python wrapper; default BM25 parameters were used. Pyserini provides convenient APIs for accessing index statistics used to compute feedback document vectors, $\tilde{f}(\hat{d}_i)$, via its IndexReader API, as well as for constructing custom Lucene queries through its Query Builder API. LLM inference was performed using vLLM [12].

4 Results

Experimental results on the MS MARCO and BEIR datasets are shown in Table 1. Below we highlight the key findings:

BM25 feedback models are more effective with HyDE-generated documents. When controlling for the feedback model (Avg Vector, RM3, Rocchio), we find that applying relevance feedback over HyDE documents consistently improves Recall@20 relative to applying the same feedback model over BM25-retrieved documents. For example, feedback with Avg Vector, RM3, and Rocchio over HyDE (gpt-oss-20b) documents yields improvements of

Table 1: Main results (Recall@20) on MS MARCO and BEIR. Bold denotes best feedback approach for a given LLM.

	MS MARCO					BEIR										
	DL19	DL20	DL21	DL22	DL23	COVID	News	SciFact	FiQA	DBpedia	NFCorpus	Robust04	SCIDOCS	ArguAna	BEIR (Avg.)	All (Avg.)
BM25	27.0	32.6	14.7	4.4	12.5	3.0	20.8	85.2	37.1	28.3	18.0	20.0	20.6	79.1	34.7	28.8
BGE-base-en-v1.5	37.6	49.5	-	-	-	4.0	26.7	92.4	55.7	33.2	22.1	18.9	30.0	95.3	42.0	-
Feedback Models																
BM25 + Avg Vector	30.1	35.2	17.4	5.0	12.5	3.0	24.3	86.5	31.5	28.3	21.7	19.3	21.6	79.7	35.1	29.7
BM25 + RM3	28.9	36.3	17.5	4.9	12.6	3.3	24.9	86.9	35.4	28.7	22.0	21.0	22.0	85.5	36.6	30.7
BM25 + Rocchio	28.7	35.3	17.3	5.1	12.9	3.1	25.2	87.4	33.7	29.1	22.3	20.8	22.0	81.9	36.2	30.4
Qwen2.5-7B																
String Concatenation																
Query2Doc	34.1	44.5	18.2	10.6	**17.1**	3.6	24.2	**90.5**	37.7	31.3	21.3	21.9	21.3	80.8	37.0	32.6
HyDE + Naive Concat	29.5	36.9	17.6	11.1	14.6	3.6	22.4	88.9	29.2	27.4	20.2	17.6	21.0	69.9	33.3	29.3
HyDE + MuGI Concat	**37.3**	**46.4**	**20.3**	12.2	16.5	**4.0**	26.4	89.4	36.6	33.6	21.3	21.0	22.0	74.0	36.5	32.9
Feedback Models																
HyDE + Avg Vector	33.3	41.9	18.8	**12.3**	15.1	3.4	26.8	89.3	35.8	29.2	21.7	23.0	**22.6**	78.1	36.6	32.2
HyDE + RM3	32.8	40.1	19.5	9.1	15.1	3.7	26.4	89.3	**40.6**	32.3	21.2	23.3	22.2	82.4	37.9	32.7
HyDE + Rocchio	36.9	45.5	20.3	11.7	16.5	3.9	**27.0**	90.5	39.9	**33.9**	**21.8**	**23.7**	22.4	**82.9**	**38.4**	**34.0**
Qwen3-14B																
String Concatenation																
Query2Doc	32.5	44.2	20.9	10.1	15.5	3.7	26.5	90.9	38.3	32.5	21.5	22.1	22.2	81.1	37.7	33.0
HyDE + Naive Concat	29.2	38.7	16.9	11.4	14.8	3.4	24.8	90.1	31.5	27.3	21.2	18.0	21.1	74.8	34.7	30.2
HyDE + MuGI Concat	37.3	46.0	22.5	**12.4**	**16.4**	**3.9**	25.9	90.9	36.9	**34.0**	21.6	21.2	22.8	76.4	37.1	33.4
Feedback Models																
HyDE + Avg Vector	30.7	42.5	18.1	11.8	15.3	3.3	27.8	**91.1**	36.3	29.2	**22.3**	23.1	**23.7**	79.8	37.4	32.5
HyDE + RM3	33.0	40.5	20.3	8.8	14.8	3.8	27.4	90.8	41.1	32.8	21.4	24.0	22.4	83.3	38.5	33.2
HyDE + Rocchio	**37.4**	**47.3**	**22.9**	11.8	16.2	3.8	**27.8**	90.9	**41.4**	33.7	22.1	**24.0**	23.4	**83.8**	**39.0**	**34.7**
gpt-oss-20b																
String Concatenation																
Query2Doc	37.5	44.0	20.6	9.9	**15.5**	3.8	24.8	87.7	39.0	31.2	20.6	21.4	21.1	80.9	36.7	32.7
HyDE + Naive Concat	28.9	38.8	18.9	8.8	14.0	3.1	24.5	91.8	31.3	26.5	20.6	18.2	14.7	72.4	33.7	29.5
HyDE + MuGI Concat	**38.4**	45.3	23.0	12.1	14.9	3.8	26.0	90.6	39.4	33.6	21.1	21.9	20.9	74.8	36.9	33.3
Feedback Models																
HyDE + Avg Vector	32.9	42.0	19.5	10.3	13.9	3.2	28.7	91.5	37.7	28.9	**22.5**	24.1	21.8	78.6	37.4	32.5
HyDE + RM3	31.8	41.4	20.4	9.2	14.9	**3.9**	27.0	91.0	**42.1**	32.4	21.6	24.5	22.2	82.5	38.6	33.2
HyDE + Rocchio	36.9	**45.8**	**23.5**	**12.3**	15.4	3.8	**28.1**	**91.9**	42.0	**33.8**	22.1	**24.6**	**22.7**	**82.5**	**39.1**	**34.7**

2.8, 2.5, and 4.3 points, respectively, compared to feedback with BM25-retrieved documents across all datasets. This trend is stable regardless of the LLM: when running HyDE using Qwen2.5-7B, we find similar improvements (2.5, 2.0, and 3.6 points on average, respectively). Interestingly, we note that while RM3 is less effective than Rocchio when applied to HyDE documents, this is not the case when applied to top-ranked BM25 documents.

Filtering out noisy terms from HyDE-generated documents improves effectiveness. Comparing HyDE + Avg Vector to HyDE + Naive Concat, we find that the average vector approach is consistently more effective, yielding improvements of 2.9, 2.3, and 3.0 points on average for Qwen2.5-7B, Qwen3-14B, and gpt-oss-20b, respectively. As the main difference between HyDE + Avg Vector and HyDE + Naive Concat lies in the term selection step, this shows that it is helpful to filter out noisy terms from HyDE documents.

Rocchio feedback demonstrates the strongest effectiveness with HyDE. Across all LLMs, HyDE + Rocchio consistently outperforms the other feedback models and string concatenation methods. Compared to HyDE + Avg Vector, which operates in a similar vector space, the crucial difference is that the Rocchio algorithm gives more weight to the query terms. This suggests that the HyDE feedback mechanism (i.e., average vector) proposed in Gao et al. [10] *underemphasizes* query terms. Additionally, one advantage of the Rocchio algorithm compared to the adaptive query reweighting of MuGI Concat is that the query weight can be adapted linearly with a single parameter rather than being dependent on other factors such as feedback document length. An interesting area for future work would be to investigate how to adaptively reweight the α and β parameters for Rocchio using a scheme like that of MuGI.

String concatenation approaches are less effective on diverse retrieval tasks in BEIR. Compared to feedback models, Query2Doc and HyDE + MuGI Concat are generally competitive on the MS MARCO datasets, but are consistently less effective on BEIR. Taking HyDE + RM3 using gpt-oss-20b as a representative case, Query2Doc and HyDE + MuGI Concat beat HyDE + RM3 on all MS MARCO datasets. However, on BEIR, RM3 beats Query2Doc on all 9 datasets, and MuGI Concat on 8 of the 9. This suggests that updating the query representation using feedback models is generally more *robust* to different query variations. In fact, BM25 + RM3 or BM25 + Rocchio with no LLM is competitive with Query2Doc and HyDE + MuGI Concat – and more effective than HyDE + Naive Concat – on the BEIR datasets.

HyDE with feedback models brings BM25 closer to a strong "single-shot" dense retriever. Lastly, our results demonstrate

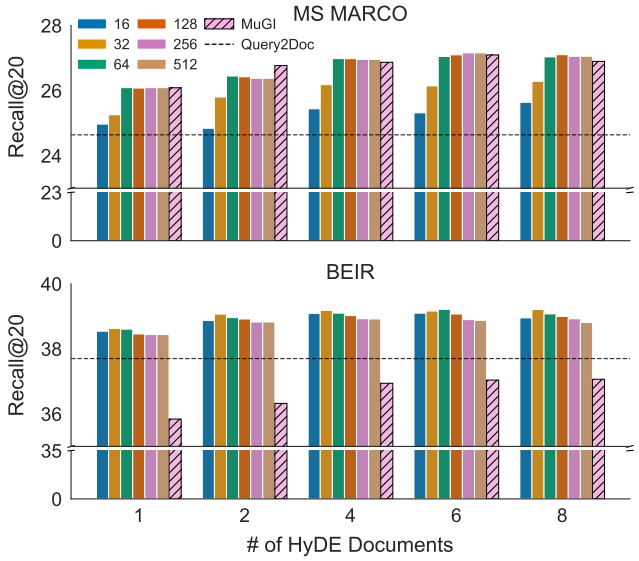

Figure 2: Varying the number of feedback terms and generated HyDE documents on HyDE + Rocchio accuracy. Recall@20 represents an average over the datasets in Table 1.

that using HyDE with better feedback mechanisms can push a BM25 retrieval pipeline closer to the effectiveness of BGE on BEIR, without requiring any of the costs associated with dense retrievers. We highlight that this was achieved through simply *adapting* well-established BM25 feedback models to the HyDE framework. This naturally leads to the question of how much further the gap can be reduced by developing feedback models which are better optimized for HyDE documents.

5 Analysis of Feedback Size

In this section, we investigate two hyperparameters which may influence the accuracy of HyDE with traditional feedback models: the number of feedback terms and the number of HyDE-generated documents. We run HyDE + Rocchio with a varying number of (1) feedback terms and (2) generated HyDE documents over the MS MARCO and BEIR datasets. We also evaluate MuGI under varying numbers of generated HyDE documents and include Query2Doc as a baseline. Qwen3-14B is the LLM considered for this experiment. The results are in Figure 2.

On MS MARCO, increasing the number of feedback terms for HyDE + Rocchio results in a consistent improvement in accuracy up until 64 terms. Beyond 64 feedback terms, however, effectiveness is largely stable. We observe a similar trend with respect to the number of HyDE documents: effectiveness improves until reaching four generated feedback documents.

On BEIR, HyDE + Rocchio exhibits more stability across different numbers of feedback terms and HyDE documents, suggesting that increasing these parameters is not necessarily more beneficial on the different BEIR search tasks. This further demonstrates – at least on BEIR – that properly selecting *good* feedback terms (e.g., by filtering out noisy terms as described in Section 2.1.1) is a valuable step. In fact, when using *only* 16 feedback terms with one HyDE document, HyDE + Rocchio outperforms the best MuGI baseline despite leveraging far fewer feedback terms.

6 Conclusion

In this work, we revisit traditional BM25 feedback models for *pseudo*-relevance feedback in the context of HyDE. Our experiments across 14 retrieval datasets demonstrate that BM25 feedback models and HyDE are mutually beneficial: by simply swapping top-ranked documents retrieved by BM25 with HyDE-generated documents, Rocchio and RM3 feedback see notable improvements in effectiveness. Conversely, by incorporating BM25 feedback models into the HyDE setup, HyDE's effectiveness increases by up to 1.4 points (4.2%) – with a 2.2-point (6%) improvement on BEIR – relative to the recent standard practice of integrating LLM-generated documents with BM25 via string concatenation.

Given these gains, an obvious follow-up question is: *Why does HyDE with BM25 feedback models outperform HyDE with string concatenation*? Our results suggest that improvements come from two sources. The first is that feedback models implicitly filter out noisy terms, keeping only terms that are meaningful within the corpus and assigned high weight within HyDE's generations. The second improvement, we hypothesize, comes from a more stable and linear weighting of query terms and terms in the HyDE-generated documents via simple parameters such as α and λ, rather than through string repetition. This may help explain why the average vector, RM3, and Rocchio feedback models demonstrated stronger robustness on the diverse range of queries present across the BEIR retrieval tasks.

Our findings highlight that BM25 feedback models should not be overlooked when leveraging methods like HyDE for relevance feedback. In fact, they represent a simple way to better utilize the expansions generated by LLMs.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Additional funding was provided by Snowflake and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project). We would like to thank Basit Ali, Yijun Ge, Kevin Wang, and Felix Labelle for their thoughtful feedback and contributions.

References

[1] Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *TREC*.

[2] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. 2025. GPT-OSS-120B & GPT-OSS-20B Model card. *arXiv:2508.10925* (2025).

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268* (2016).

[4] Claudio Carpineto, Giovanni Romano, and Vittorio Giannini. 2002. Improving Retrieval Feedback with Multiple Term-Ranking Function Combination. *ACM Transactions on Information Systems (TOIS)* 20, 3 (2002), 259–290.

[5] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Jimmy Lin. 2021. Overview of the TREC 2021 Deep Learning Track. In *TREC*.

[6] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2022. Overview of the TREC 2022 Deep Learning Track. In *TREC*.

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 Deep Learning Track. *arXiv:2003.07820* (2020).

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2021. Overview of the TREC 2020 Deep Learning Track. *arXiv:2102.07662* (2021).

[9] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Hossein A. Rahmani, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2023. Overview of the TREC 2023 Deep Learning Track. In *TREC*.

[10] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1762–1777.

[11] Yijun Ge, Sahel Sharifymoghaddam, and Jimmy Lin. 2025. Lighting the Way for BRIGHT: Reproducible Baselines with Anserini, Pyserini, and RankLLM. *arXiv:2509.02558* (2025).

[12] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.

[13] Victor Lavrenko and W. Bruce Croft. 2017. Relevance-Based Language Models. In *ACM SIGIR Forum*, Vol. 51. 260–267.

[14] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2356–2362.

[15] Joseph John Rocchio Jr. 1971. Relevance Feedback in Information Retrieval. *The SMART Retrieval System: Experiments in Automatic Document Processing* (1971).

[16] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-Shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[17] Shivani Upadhyay, Ronak Pradeep, Nandan Thakur, Daniel Campos, Nick Craswell, Ian Soboroff, and Jimmy Lin. 2025. A Large-Scale Study of Relevance Assessments with Large Language Models Using UMBRELA. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval*. 358–368.

[18] Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query Expansion with Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 9414–9423.

[19] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 641–649.

[20] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. *arXiv:2505.09388* (2025).

[21] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2.5 Technical Report. *arXiv:2412.15115* (2024).

[22] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1253–1256.

[23] Le Zhang, Yihong Wu, Qian Yang, and Jian-Yun Nie. 2024. Exploring the Best Practices of Query Expansion with Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 1872–1883.