Predicting RAG Performance for Text Completion

Oz Huly
ozhuly@campus.technion.ac.il
Technion
Haifa, Israel

David Carmel
david.carmel@tii.ae
TII and Technion
Haifa, Israel

Oren Kurland
kurland@technion.ac.il
Technion
Haifa, Israel

Abstract

We address the challenge of predicting the performance of using retrieval augmented generation (RAG) in large language models (LLMs) for the task of text completion; specifically, we predict the perplexity gain attained by applying RAG. We present novel supervised post-retrieval prediction methods that utilize the specific characteristics of the text completion setting. Our predictors substantially outperform a wide variety of prediction methods originally proposed for ad hoc document retrieval. We then show that integrating our post-retrieval predictors with recently proposed post-generation predictors — i.e., those analyzing the next-token distribution — is of much merit: the resultant prediction quality is statistically significantly better than that of using the post-generation predictors alone. Finally, we show that our post-retrieval predictors are as effective as post-generation predictors for selective application of RAG. This finding is of utmost importance in terms of efficiency of selective RAG.

CCS Concepts

• **Information systems** → **Information retrieval**.

Keywords

LLM, RAG, Text completion, Performance prediction

ACM Reference Format:
Oz Huly, David Carmel, and Oren Kurland. 2025. Predicting RAG Performance for Text Completion. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3730062

1 Introduction

Large language models (LLMs) have demonstrated remarkable accomplishments in various NLP tasks [4]. Retrieval Augmented Generation (RAG) [12, 19, 28, 36] is a widely accepted methodology to ground the generation process by incorporating relevant, accurate, and often fresh evidence from external sources to improve the quality of the LLM response. It has been shown that RAG contributes to making the LLM more up-to-date, as well as reducing hallucinations in the generated content [41], and making the pre-trained LLM more adaptable to a variety of domains [52].

SIGIR '25, Padua, Italy.

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730062

RAG has mostly been studied for open domain question answering and for conversational dialog systems [12, 19, 28, 43], showing sustainable improvements in response quality. RAG has also been studied for downstream tasks such as recommender systems [54], report generation [9] and code generation [45]. Our main interest in this work is in RAG's contribution to the LLM fundamental task of text completion. In this task, the LLM is asked to generate a natural language response that complements the provided input. We follow the in-context RAG framework [36] where the LLM and the retriever are fixed; content retrieved from an external source is used to augment the original LLM's input to improve text completion.

However, external content is not always needed for successful generation [30]. Moreover, retrieved content can sometimes be irrelevant or even distracting [7]. In such cases, RAG may badly affect the LLM's response. To mitigate this potential flaw, it was suggested to apply RAG selectively [21, 43], or to filter retrieved content by predicting its potential contribution before augmenting the original input [2, 49–51]. A reliable performance predictor for RAG can be utilized to invoke RAG selectively, only when needed.

In this paper, we focus on prediction approaches for RAG performance in the context of the text completion task. Prediction can be done a priori, based on the query used for retrieval (pre-retrieval), a posteriori, based on the retrieved content (post-retrieval), or even by analyzing the characteristics of the next-token distribution induced by the LLM (post-generation).

We begin with the analysis of classical query performance prediction (QPP) approaches [5] which have long been studied by the IR community. These methods were devised to predict the effectiveness of search results in lack of relevance judgments. In our RAG setting, search results are effective to the extent that they contribute to successful generation [11]. This is a departure from the standard notion of relevance effectiveness. A case in point, it was recently shown that classical sparse retrieval methods (e.g., Okapi BM25) are substantially more effective for the text completion task than dense retrieval methods [18, 36], although the latter often post superior performance for standard ad hoc retrieval (e.g., [23]).

We then present novel supervised post-retrieval prediction methods that model different types of associations between parts of the input to the LLM. These methods are trained directly to predict the gain attained by using RAG. Furthermore, we use a supervised approach to integrate the post-retrieval predictors with post-generation prediction methods that analyze the LLM's induced next-token distributions [24, 48]: with and without RAG.

Our empirical evaluation yields consistent results for two LLMs. Pre-retrieval predictors devised for ad hoc retrieval are completely ineffective in the RAG setting. Post-retrieval predictors for ad hoc retrieval yield moderate prediction quality which is lower than that observed in the original task they were designed for. Our novel

post-retrieval predictors substantially outperform all the predictors we adapted from work on ad hoc retrieval.

We also empirically show that integrating our post-retrieval predictors with post-generation predictors yields prediction quality that is statistically significantly better than that of post-generation predictors. This finding attests to the complementary nature of post-retrieval and post-generation predictors.

We next address the selective RAG challenge: deciding per LLM input whether to apply RAG so as to minimize the resultant perplexity. Using our post-retrieval predictors results in perplexity that is on par with that attained by using post-generation predictors. This finding is of utmost importance as post-retrieval predictors are significantly more efficient than post-generation predictors: the latter require LLM generation while the former do not.

The main contributions of this work are:

- We introduce a new challenge of RAG performance prediction in the context of the text completion task. We suggest novel metrics for measuring the performance gain of applying RAG for text generation. To the best of our knowledge, no previous studies focused on this specific task.
- We adapt pre- and post- retrieval predictors devised for ad hoc retrieval to the RAG setting. The former are ineffective while the latter post moderate prediction quality.
- We present novel supervised post-retrieval predictors that substantially outperform all the predictors devised for ad hoc retrieval which we adapted.
- We show that integrating our post-retrieval predictors with post-generation predictors yields statistically significant improvements over using the post-generation predictors alone.
- We show that our post-retrieval predictors are as effective as post-generation predictors for selective application of RAG, while they are considerably more efficient.

2 Related Work

The two lines of work most related to ours are on query performance prediction and prediction of RAG performance.

Query performance prediction. There is a large body of work on query performance prediction (QPP) for ad hod document retrieval [5]. Pre-retrieval prediction methods operate prior to retrieval time analyzing the query and the corpus [16]. Post-retrieval predictors analyze also the retrieved list (e.g., [39, 40, 42, 55]). While traditional QPP methods are are mostly focused on sparse retrieval, there were recently proposed neural-IR prediction methods [10], and supervised prediction methods; e.g., BERT-QPP [1]. LLMs were also used to predict ad hoc retrieval effectiveness by generating pseudo relevance labels [31]. In a related line of work, QPP methods were adapted to the question answering domain [14, 25, 33, 38].

The prediction task we address is different than QPP for retrieval and question answering where relevance, or inclusion of answers, is predicted. Here, we predict the contribution of passages to the quality of text completion performed by an LLM that uses them. Since RAG in our setting is based on passage retrieval for a query, we adapt various pre- and post- retrieval predictors devised for ad hoc retrieval to our setting. We show that our novel post-retrieval predictors substantially outperform all these methods.

RAG performance prediction. As mentioned in Section 1, RAG can sometimes result in providing the LLM with irrelevant, or even distracting information, which negatively impacts the quality of text generation [7, 30, 51]. A few approaches were suggested to address this issue.

Yoran et al. [51] filter retrieved passages that do not entail the input according to a natural language inference (NLI) model. Mallen et al. [30] apply retrieval selectively, only for un-popular entities for which parametric-based knowledge may fail. Wang et al. [43] fine-tuned an LLM (RAGate) to dynamically decide on the need for external content during conversation. Ram et al. [36] trained a predictive re-ranking model that estimates the contribution of each passage for text completion, allowing better selection of augmenting content. We use Ram et al.'s re-ranking method [36] as a basis for one of our post-retrieval predictors.

In Self adaptation, the LLM is trained to adaptively decide on whether and how to deal with retrieved content [2, 49, 50]. Self-RAG [2] trains the LLM to predict if retrieval content is actually needed, and to predict the usefulness of any retrieved passage for response generation. This is done by training the model to generate "reflection tokens" which makes the LLM controllable during the inference phase, enabling it to tailor its behavior for diverse task requirements. Corrective RAG (CRAG) [49] follows the same paradigm. A lightweight retrieval evaluator is trained to evaluate the relevance of the retrieved passages, and to estimate a confidence degree based on which different retrieval actions can be triggered. Besides, it selectively focuses on key information in retrieved passages and filters irrelevant information out of them.

The approaches just described were devised for question answering and are mainly based on finetuning the LLM and/or the retriever, or filtering results. We operate in the in-context RAG framework [36] for the text completion task where the LLM and the retriever are fixed. Our goal is the predict the contribution of passages retrieved by a given retriever to the text completion performed by a given LLM.

There is work on selectively invoking the retrieval process during generation (a.k.a., selective RAG). Jiang et al. [22] proposed to look ahead for making an augmentation decision. The LLM generates a temporary text; this text is used to query the external corpus only if it contains low-probability tokens. Wang et al. [44] trained several classifiers to discern between questions which require external knowledge and those which do not. Jeong et al. [21] learn to dynamically select the most suitable strategy for RAG-LLMs, from no retrieval at all to multiple retrievals, based on the query complexity.

In a conceptually related vein, Xu et al. [48] used the entropy of the two next-token distributions (induced with and without RAG), and their KL divergence, to selectively apply RAG. Kim et al. [24] fuse the two next-token distributions by using the entropy of the distributions to control the reliance on each of them.

Inspired by this line of work, we devise post-generation predictors for the text completion task. These predictors utilize the entropy of the next-token distributions (induced with and without RAG) [24, 48] and their KL divergence [48]. We show that integrating these predictors with our proposed post-retrieval predictors statistically significantly improves prediction quality. Furthermore, we use various predictors to address the selective RAG task. We

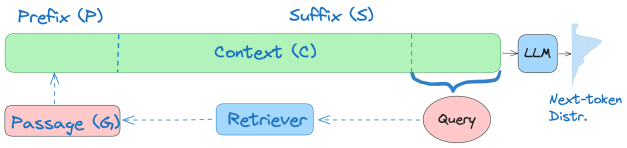

Figure 1: The in-context RAG-LLM setting [36]. The text that results from detokenizing the last k tokens (Q) is used to query the Retriever. The highest ranked passage (G) replaces the prefix of the context.

show that our post-retrieval predictors are as effective as the post-generation predictors in terms of resultant perplexity. This is an important result because post-generation predictors are far less efficient than post-retrieval predictors as they require inference of the LLM to make a prediction.

3 Prediction Framework

We address the fundamental text completion task using LLMs. That is, an LLM is provided (prompted) with a sequence of tokens which are sub-word units. The LLM then generates the following text to complete the input.

Formally, the generation probability of a token sequence, $t_1, \ldots, t_m$, using an autoregressive LLM is based on next-token prediction:

$$p(t_1, \ldots, t_m) = \prod_{i=1}^{m} p_\theta(t_i|t_{<i}); \quad (1)$$

θ is the model parameters of the LLM; $t_{<i} = t_1, \ldots, t_{i-1}$ is the subsequence of tokens preceding t_i, also referred to as its *context*. During text generation, at each step i the LLM uses the current prefix $t_{<i}$ to induce a next-token distribution over the vocabulary: $p_\theta(t|t_{<i})$. The next token t_i is then selected by a pre-defined selection strategy; e.g., $t_i = \text{argmax}_t\, p_\theta(t|t_{<i})$.

3.1 In-context RAG

The next-token probability distribution is based on knowledge encoded in the LLM's parameters. To ground the LLM-based token generation to knowledge manifested in a specific corpus — e.g., for reducing the risk of hallucinations — retrieval augmented generation (RAG) is often applied [12]. That is, the next-token distribution is based not only on the prompt but also on content retrieved from external corpora.

Herein, we subscribe to the in-context RAG-LLM framework [36] depicted in Figure 1. We assume the context is of a fixed length, determined by the LLM's input capacity. We split it to a prefix (P), the $l > 0$ leftmost tokens in the context, and a suffix (S). Accordingly, the next token distribution can be written as: $p_\theta(t|P, S)$. A query is derived from the context and is used by the Retriever to retrieve a ranked passage list, L, from a corpus. The top-n ranked passages $(G_1, \ldots, G_n)$ are used to replace the prefix (P)[1].

Following Ram et al. [36], we use the text that results from detokenizing the k rightmost tokens in the context as a query (Q)[2]. Ram et al. [36] also demonstrated in their setting that using more than one passage for text completion has no merit[3]. We therefore fix n to 1 in our setting. Finally, the prefix length ($l > 0$) is set to be the same as a passage length. The resultant modified context, namely the passage followed by the suffix, is used as a prompt to the LLM to induce a RAG-based next-token distribution over the token space: $p_\theta^{RAG}(\cdot|G, S)$.

3.2 Prediction task

The core challenge we address is predicting the relative effectiveness of using RAG for text completion with respect to not using it. To measure effectiveness, for a given context $C = (P, S)$, and the corresponding retrieved passage G, we induce the next-token probability distributions $p_\theta(\cdot|P, S)$ and $p_\theta^{RAG}(\cdot|G, S)$ defined over the token space. We then follow standard practice in work on evaluating language models [13], and measure the ratio between the perplexity of the two next-token distributions. Since we focus only on the next token, the perplexity of a next-token distribution $p(\cdot)$ with respect to the actual next token $\hat{t}$ in the text is $\frac{1}{p(\hat{t})}$. Accordingly, given context $C = (P, S)$ followed by the actual next token $\hat{t}$, and a LLM with parameters θ, we define the relative (log) gain for next-token generation of using RAG with a retrieved passage G as:

$$gain(G|\hat{t}, C, \theta) \stackrel{def}{=} \log \frac{p_\theta^{RAG}(\hat{t}|G, S)}{p_\theta(\hat{t}|P, S)}. \quad (2)$$

The prediction task we address below is predicting the gain defined in Equation 2 with no knowledge of the actual next token $\hat{t}$.

3.3 Prediction methods

In classical query-performance prediction for ad hoc retrieval [5], the challenge is to estimate search effectiveness in lack of relevance judgments. There are two major differences between the prediction task we address here and that addressed in ad hoc (query based) retrieval. First, in our RAG setting, there is no clear notion of "passage relevance". A passage is effective/valuable to the extent that using it results in high gain in Equation 2; that is, the extent to which using the passage improves text generation quality with respect to not applying RAG. Still, we study whether predictors originally devised for ad hoc retrieval can be effective in our RAG setting. Below we survey a few pre-retrieval predictors (Section 3.3.1) and post-retrieval predictors (Section 3.3.2) which we evaluate in Section 4 in the RAG setting. These methods use as input the query Q, the passage corpus upon which retrieval is performed and the retrieved passage list, L. The correlation between the prediction value

[1] It is often the case that the passages are prepended to the context rather than override it. We follow the override approach used by Ram et al. [36] in their evaluation which was shown to yield highly effective generation as is the case in our experiments. The motivation is to allow longer context to be utilized when RAG is not applied, specifically, when using LLMs with a relatively short input-length capacity.
[2] https://github.com/huggingface/tokenizers
[3] We arrived to the same finding. Actual numbers are omitted as they convey no additional insight.

assigned by a method and the actual gain in using RAG (Equation 2) is used in Section 4 as evaluation measure for prediction quality.

The special structure of the RAG-LLM setting we use here is the second major differentiator between our prediction task and that in ad hoc retrieval. A passage is used to replace the prefix; together with the suffix it serves as the context used by the LLM to induce the next-token distribution. Accordingly, in Section 3.3.3 we present novel prediction methods that account for the special structure of the RAG-LLM setting.

3.3.1 Adopting pre-retrieval predictors devised for ad hoc retrieval. Pre-retrieval predictors operate prior to retrieval [5]. These unsupervised methods analyze the query terms (Q) and some corpus term statistics.

- **SCQ**: The mean, max or min tf.idf weight of a query term with respect to the corpus [53]. The premise is that high query-terms weights indicate effective retrieval.
- **IDF**: The mean, max and min of the idf (inverse document frequency) values of query terms [26]. The premise is that using highly discriminative query terms results in effective retrieval.
- **VAR**: The mean, max and min of the variance of a query term's tf.idf weight in documents in the corpus in which it appears [53]. As the IDF predictors, the VAR predictors estimate the discriminative power of query terms.

3.3.2 Adopting post-retrieval predictors devised for ad hoc retrieval. Post-retrieval predictors analyze the retrieved list, L, in addition to the query and the corpus. While we use only the highest ranked passage, G, to induce the next-token distribution as mentioned in Section 3.1, the post-retrieval predictors analyze additional documents in L; indeed, they were designed to predict list effectiveness rather than single-document effectiveness. Therefore, here we study whether the predicted list effectiveness is correlated with the gain attained by using the highest ranked passage for text generation. We explore the following highly effective post-retrieval predictors, all of which, except for BERT-QPP, are unsupervised.

- **WIG**: The difference between the average retrieval score of the top-ν passages in L and the corpus retrieval score [55]; ν is a parameter. The assumption is that the higher the difference, the more effective the retrieval as the corpus is essentially a pseudo non-relevant document. We also study a variant, **U_WIG**, which uses the average of top scores without the corpus score regularization [40].
- **NQC**: The standard deviation of retrieval scores of the top-ν passages in L normalized by the corpus retrieval score [40]. We also study a variant, **QC**, which does not use the corpus normalization. The assumption is that high standard deviation indicates reduced query drift and, hence, improved retrieval effectiveness [40].
- **SMV**: Combines the mean and variance of the retrieval scores of the top-ν passages in L; the corpus retrieval score is used for normalization [42]. We also study a variant, **U_SMV**, which does not apply corpus normalization.
- **REF**: A reference-list predictor [39] that measures the RBO similarity [46] (with parameter p) between the top-ν passages in the list L and the top-ν passages in a highly effective ranked passage list which is used for reference. We use two predictors which produce a reference list by re-ranking L. The first predictor, **REF_PRED**, uses a recently proposed predictive re-ranker [36]. The second predictor, **REF_RM3**, uses relevance model #3 (RM3) [20, 27] for re-ranking. The **REF_both** predictor uses the (unweighted) average RBO with both reference lists just described.
- **BERT-QPP**: A supervised predictor that models the textual association of a query with top-retrieved documents [1]. We adapt BERT-QPP to our RAG setting as follows. We use the query Q and the top passage in L as input to DeBERTa [17] — a highly effective variant of BERT [8]. We add a linear regression layer, trained with a min squared error (MSE) loss with respect to the gain in Equation 2.

3.3.3 Novel post-retrieval predictors. We now turn to present novel post-retrieval predictors devised for the RAG setting.

PredReranker. This (supervised) predictor is inspired by Ram et al.'s [36] highly effective passage (re-)ranking method for RAG-LLM, dubbed Predictive ReRanker [36]. The ranking model[4] is a DeBERTa-based classifier trained to maximize the probability assigned by the LLM to the actual next token while using the retrieved passage for RAG [28, 36]. The input to DeBERTa is the passage G and the suffix S separated with a token[5]. The assumption is that the association between G and S which together constitute the (new) context used for LLM generation can attest to G's contribution to generation quality.

Our **PredReranker** prediction method uses the difference between the retrieval score assigned by the trained reranker to the passage G and the score assigned to P (the prefix) as the prediction value. The difference is an estimate for the potential gain (in terms of generation quality) in using the passage instead of the prefix in the context provided to the LLM[6].

TripletPred. This family of supervised predictors utilize three types of textual associations or subsets of which. The predictors are trained to directly predict the gain in Equation 2.

- (G, P): Since the passage (G) replaces the prefix (P), the strength of their association is presumably an indicator for an effective next-token distribution or lackthereof.
- (G, S): The association of the passage (G) with the suffix (S) can potentially also serve as an indicator as together they constitute the context used by the LLM to induce the next-token distribution. This association is used by the Predictive ReRanker ranking method [36] underlying the PredReranker predictor from above.
- (P, S): The association between the prefix (P) and the suffix (S) which together constitute the original context. Their association is also hypothesized to attest to the effectiveness of the next-token distribution for two reasons. First, since the prefix is replaced with the passage, the less it is associated with the suffix the more potential there is for a passage containing a completely different text to "blend in". Second, since (i) we predict the relative

[4] It was originally used to re-rank an initially retrieved passage list [36].
[5] The suffix is trimmed from the left to adjust to DeBERTa's input-lenth limit. Originally, the passage and the entire context were used as input to re-rank the passage list [36]. Since the prefix is overriden by the passage, we use here only the passage and the suffix which results in highly effective re-ranking performance. The next predictor accounts for relations between all parts of the context and the passage.
[6] Using only the score assigned to G resulted in prediction quality substantially inferior. These results are omitted as they convey no additional insight.

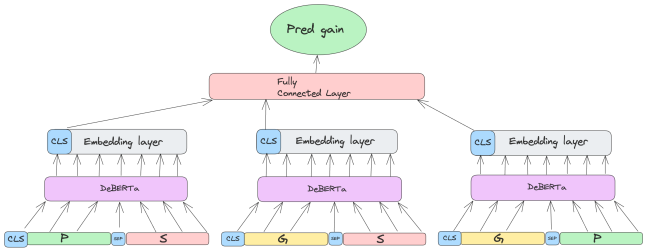

Figure 2: An example of a predictor from the TripletPred family which uses all three types of association: (P, S), (G, S) and (G, P). P, S and G are the prefix, suffix and passage, respectively, from Figure 1. The three instances of DeBERTa have shared weights.

effectiveness (gain) of using a retrieved passage with respect to not using it, and (ii) the prefix together with the suffix is the context used for producing a next-token distribution with no retrieval, then the association of the prefix and the suffix can potentially attest to the effectiveness of this distribution.

We model the inter-relations between prefix, suffix and the passage by training a DeBERTa-based model. Figure 2 shows as an example of one predictor in this family which uses all three types of associations just described. We use three instances of DeBerta with shared weights. Each pair of the suffix (S), prefix (P) and passage (G) is fed as an input to one of the three instances, with a [SEP] token separating them, and a [CLS] token positioned at the beginning of the input[7]. We clip tokens at the left of the suffix to meet the length limit of DeBERTa. The three embedding vectors of the [CLS] tokens are fed to a fully connected network with a regression layer; specifically, we use the Mean Squared Error loss function where the target value is $gain(G|\hat{t}, C, \theta)$ from Equation 2.

In Section 4 we show that using two out of the three association types resulted in the best prediction quality, specifically, better than using all three types of association together. The best performing predictor in this family uses (G, S) and (G, P). Herein, we will refer to this specific predictor as **TripletPred**. The second best performing predictor in this family utilizes (G, S) and (P, S). These two predictors are trained by using two instances of DeBerta with shared weights rather than all three instances depicted in Figure 2. We also study the performance of using a single type of association (e.g., (G, S)) where a single instance of DeBERTa is used. The empirical study of the prediction quality of all TripletPred predictors is presented in Section 4.2.5.

One might wonder why the query that served for passage retrieval is not used by TripletPred; specifically, its relations with the other parts of the RAG setting (e.g., prefix and retrieved passage). We found that utilizing these query relations results in prediction of substantially lower quality than that of TripletPred. (Actual numbers are omitted as they convey no additional insight.) We attribute this finding to the following. Prediction quality is determined based on the effectiveness of the next-token distribution with respect to the actual text. This quality is affected by two factors: (i) the LLM and (ii) the context used to prompt it. Hence, in our setting the query is just a means to retrieving passages whose effectiveness (gain in Equation 2) is determined with respect to the down stream task of generation rather than with respect to relevance.

3.3.4 Post-generation predictors. The predictors presented thusfar operate before the LLM is prompted to induce a next-token distribution. There is recent work on using properties of the next-token distributions induced with and without RAG so as to fuse them [24] and to decide whether to apply RAG [48]. We use these distribution properties to devise two post-generation predictors so as to address the following research question: does the integration of post-retrieval and post-generation prediction yield merit?[8] It is important to note that post-retrieval prediction is highly more efficient than post-generation prediction as there is no need for LLM inference to perform the prediction. Specifically, while post-retrieval predictors are based on retrieval, post-generation predictors are based on retrieval and two calls to the LLM to induce next-token distributions (with and without the retrieved passage).

The first post-generation predictor, **EntPred**, adapted from work on fusing next-token distributions induced with and without RAG [24], is the difference between the entropy of the next-token distribution when not using RAG ($p_\theta(\cdot|P, S)$) and the entropy of the next-token distribution when using RAG ($p_\theta^{RAG}(\cdot|G, S)$) ; entropy is defined over the token space. High entropy of a next-token distribution can potentially attest to increased uncertainty about the next token [24, 48]. Accordingly, the likelihood for an error with respect to the actual text can potentially be relatively high.

The second post-generation predictor, **DiverPred**, adopted from work on selective application of RAG [48], assigns the prediction value: $KL(p_\theta^{RAG}(\cdot|G, S)||p_\theta(\cdot|P, S))$: the KL divergence between the next-token distribution induced using RAG and that induced without using RAG. The premise is that the higher the KL divergence (i.e., the less similar the distributions), the higher the gain of using the passage for LLM generation (Equation 2).

3.3.5 Integrating predictors. As noted above, a research question we are interested in is whether there is merit in integrating post-retrieval and post-generation predictors. To integrate predictors, we apply a feature-based mixture of experts approach: we use the prediction values assigned by the predictors as features in Gradient Boosted Regression Trees (XGBoost implementation) [6] trained for the gain defined in Equation 2. Additional technical details are provided in Section 4.1.

4 Evaluation

4.1 Experimental Setup

Our main experimental setting follows common practice in the query performance prediction (QPP) literature for ad hoc document retrieval [5]. We apply each prediction method over various contexts to predict the gain in applying RAG (Equation 2). Prediction quality is measured by the Pearson correlation between the values assigned by the prediction method to the contexts and the true gain measured with respect to the actual token following the context

[7]We cannot use a single instance of DeBERTa whose input is all three parts (prefix, suffix and passage) due to the input-length limit.

[8]We show in Section 4 that in contrast to post-retrieval prediction, pre-retrieval prediction is ineffective.

(next token)[9]. In Section 4.2.4 we study the effectiveness of using the predictors for selective application of RAG.

LLMs. We evaluate the predictors with two pre-trained LLMs: 1) Llama-3-8B[10] and 2) Falcon2-11B[11]. As is common, we use Softmax (with temperature=1) over the LLM logits to induce the next token probability distribution.

Retriever. The external passage corpus used for retrieval is based on the English Wikipedia dump from Dec. 20, 2018[12]. Specifically, articles are split to 100-word disjoint passages which constitute the corpus. We use Okapi BM25 [37] as the passage retrieval method. The Pyserini implementation was used with default parameter values [29]. Recent work [18, 36] shows that for RAG in the text-completion task, lexcial (sparse) retrieval methods as Okapi BM25 substantially outperform dense retrieval methods in terms of resultant perplexity. The query is the text that results from de-tokenization of the last $k = 32$ tokens in the suffix; $k = 32$ was shown to be effective for text completion with Okapi BM25 retrieval over the Wikipedia dump [18, 36].

Data. We use the Wikitext-103 dataset [32] to train and evaluate the predictors. This dataset has been extensively used in evaluating RAG for text completion [3, 13, 18, 36]. It is a collection of over 100 million tokens extracted from 120 articles on Wikipedia. To avoid contamination, we removed all passages of the 120 articles from the Wikipedia corpus used for retrieval.

To train the supervised models — the BERT-QPP predictor, the Predictive ReRanker [36] used by the PredReranker predictor, and the TripletPred predictors — we use GPT2 tokenizer [34] for tokenizing the text in the train set of Wikitext, due to efficiency considerations: GPT2 (1.5B parameters) is significantly smaller than Llama-3-8B and Falcon2-11B. We split the resultant tokenized text to disjoint contexts of 1024 tokens[13]. We then sample 1% (297, 680) of the contexts, each paired with its following token[14]. We use GPT2 upon each sequence (with and without RAG) to induce next-token distributions. The DeBERTa model [17] invoked in BERT-QPP, PredReranker and TripletPred, uses a different tokenizer than that of the LLMs in use (GPT2, Llama-3-8B and Falcon2-11B). Hence, we de-tokenize the token sequence here (for GPT2) and below (for Llama-3-8B and Falcon2-11B) to attain text to be used by DeBERTa.

To diversify the training set, we couple each context with each of the top-5 retrieved passages by Okapi BM25. The original next-token distribution (with no RAG) is the same for the 5 resulting instances, and the RAG-based next-token distribution changes with respect to the passage used. In the evaluation, only the top-retrieved passage (G) and the context are used as described in Section 3.1.

Note that our supervised predictors, PredReranker and TripletPred, operate in a domain-adaptation manner: GPT2 serves for training and Llama-3-8B and Falcon2-11B serve for evaluation. Hence, their prediction quality can potentially be further improved[15].

We use the test set of Wikitext-103 once with the Llama-3-8B tokenzier and once with the Falcon2-11B tokenizer to produce two evaluation datasets, one for each LLM. The Llama-3-8B dataset contains 71, 912 contexts of 1024 tokens. The Falcon2-11B dataset contains 82, 127 sequences of 1024 tokens. Each context is paired with the next token and the Okapi BM25 retrieved passage list (L).

Training. The supervised predictors, BERT-QPP, PredReranker, TripletPred were trained on the training set described above using 2 epochs, learning rate of $5 * 10^{-5}$, and batch size of 32. The same learning rate and batch size were used in [36] to train the Predictive ReRanker used by our PredReranker predictor. Its train-set performance, and that of TripletPred, did not improve when using more than 2 epochs. Following standard practice, in the first epoch, all layers but the last of DeBERTa are frozen; in the next epoch, all layers are unfrozen. DeBERTa has a limited input length of 512 tokens. In all models that use the suffix (S) in DeBERTa, it is trimmed from the left to fit the limit. For other models not using the suffix, trimming is not needed as the input length is not exceeded.

Evaluation. We evaluate the predictors separately for Llama-3-8B and Falcon2-11B using the evaluation sets described above. We apply a ten-fold cross validation procedure over each set to (i) set hyper-parameter values, and (ii) train feature-based regressors that integrate a few predictors. (Refer back to Section 3.3.5 for further details.) Specifically, in each of the 10 iterations of the cross-validation procedure, one fold serves for test and 9 folds serve for train (of regressors) and/or validation. For the unsupervised predictors, the 9 folds are used to set the hyper-parameter values. For the regressors, we randomly select a fold out of the 9 for validation and the other 8 are used to train the regressor. Once hyper-parameter values are selected for the regressor, we use all 9 folds to train it. Hyper-parameter tuning is performed to maximize prediction quality measured using Pearson correlation as described above: the correlation between the prediction value assigned to the contexts and the true gain attained by applying RAG (Equation 2).

To report prediction quality of a predictor, we do the following. We use all contexts in the evaluation set, each coupled with (i) the true gain (Equation 2) attained by applying RAG, and (ii) the prediction value assigned to the context when it was part of a test fold. (Each context is a member of a single test fold.) We report the average Pearson correlation over 1000 random samples of 1000 contexts; for each sample of 1000 contexts, the Pearson correlation is measured between the predicted value for the context and the true RAG gain. Statistically significant differences of prediction quality (i.e., average correlation over the 1000 samples) are determined using a two tailed paired t-test with $p \leq 0.05$.[16]

[9]We additionally measured Kendall's-τ correlation but do not report it as it exhibits the same pattern as Pearson correlation and provides no additional insights.
[10]https://huggingface.co/meta-llama/Meta-Llama-3-8B
[11]https://huggingface.co/tiiuae/falcon-11B
[12]https://huggingface.co/datasets/facebook/wiki_dpr
[13]This is the maximum input length in GPT2 which was also used by Ram et al. [36].
[14]This sample size resulted in model convergence over the train data.

[15]The BERT-QPP predictor [1] also operates in a domain-adaptation manner. As shown and explained below, it is ineffective in the RAG setting.
[16]Alternative evaluation paradigm of computing average Pearson correlation over multiple samples per test fold, and averaging over the test folds, yields very similar prediction quality numbers. Furthermore, the relative prediction quality patterns and statistical significance are exactly the same as we report.

Hyper parameters. The pre-retrieval predictors do not have hyper parameters. The number of highest ranked passages, ν, used from the retrieved passage list L in the WIG, U_WIG, NQC, QC, SMV and U_SMV predictors is set to values in $\{1, 5, 10, 20, 30, 40, 50\}$. The number of highest ranked passages, ν, used to compute the RBO estimate [46] in the REF predictors, and the parameter p which controls the RBO estimate, are set to values in $\{2, 4, 5, 7, 10, 15, 20\}$ and $\{0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99\}$, respectively.
The parameters of RM3 used in the REF_RM3 and REF_both predictors are: (i) original query weight is set to values in $\{0.25, 0.5, 0.75\}$, (ii) the number of top-retrieved passages from which RM3 is constructed is set to values in $\{3, 5, 7, 10\}$, and (iii) the number of terms to which the model is clipped is selected from $\{5, 10, 33, 66, 100, 200\}$. We used Pyserini's implementation of RM3 [29] where Okapi BM25 retrieval scores are used instead of query likelihood scores. In the XGBoost regressors (Section 3.3.5), the hyper parameter values were set using Optuna[17] applied over the validation fold. The number of trees and the maximal tree depth were set to values in $[2, 150]$ and $[1, 32]$, respectively. The α (controls the L_1 regularization of weights) and λ (controls the L_2 regularization of weights) parameters are set to values in $[0, 1]$. We allowed two tree growth policies: depthwise (split at nodes closest to the tree root) and lossguide (split at nodes with highest loss change) [47].

4.2 Results

4.2.1 Pre- and Post- retrieval predictors adapted to the RAG setting. Our first order of business is studying the prediction quality of using pre- and post- retrieval predictors that were originally designed for ad hoc document retrieval. (See Sections 3.3.1 and 3.3.2.) Table 1 presents the prediction quality numbers.

We first see in Table 1 that the prediction quality patterns are in general consistent across the two LLMs used: Llama-3-8B and Falcon2-11B. We also see in Table 1 that pre-retrieval predictors completely fail in predicting the gain of using RAG. This result is somewhat expected as pre-retrieval predictors analyze only the query and the corpus. These predictors were designed to analyze short queries for ad hoc retrieval. They do not utilize any information about the context used for the LLM including the retrieved passage in case of RAG.

Table 1 shows that post-retrieval predictors, which analyze also the retrieved passage (G) used to replace the prefix (P), substantially outperform pre-retrieval predictors. Two predictors that do not use corpus normalization (U_SMV and QC) outperform their counterparts that do (SMV and NQC); U_WIG which does not apply corpus normalization is underperformed by WIG which does.

BERT-QPP, the only supervised predictor in Table 1, posts the worst prediction quality among the post-retrieval predictors. BERT-QPP models semantic relations between the query and the passage G. Semantic matching was recently shown to be quite a weak signal in terms of generation quality for the text-completion task [18]. Furthermore, as discussed in Section 3.3.3, the query is only a means to retrieving a passage. The resultant context (composed of the passage and the suffix) is a major factor affecting generation quality which is not accounted for by BERT-QPP.

[17] https://xgboosting.com/xgboost-hyperparameter-optimization-with-optuna/

Table 1: Prediction quality (Pearson correlation) of pre- and post-retrieval predictors devised for ad hoc retrieval. U_SMV and QC outperform all other predictors for both LLMs in a statistically significant manner; the differences between U_SMV and QC are not statistically significant. Boldface marks the best result in a column.

	Llama-3-8B	Falcon2-11B
meanSCQ	0.012	0.013
minSCQ	0.006	0.015
maxSCQ	0.003	0.005
meanIDF	0.01	0.012
minIDF	0.003	0.017
maxIDF	0.009	0.011
meanVAR	0.007	0.013
minVAR	0.013	0.017
maxVAR	0.016	0.014
NQC	0.183	0.076
QC	**0.299**	**0.27**
WIG	0.222	0.196
U_WIG	0.184	0.129
SMV	0.187	0.075
U_SMV	0.296	**0.27**
REF_PRED	0.169	0.157
REF_RM3	0.107	0.1
REF_both	0.181	0.165
BERT-QPP	0.003	0.003

Table 1 shows that using two reference lists produced by the Predictive ReRanker [36] and RM3 [27] (i.e., the REF_both predictor) yields prediction quality that transcends that of using each alone (REF_PRED and REF_RM3). This implies that the two rankers yield complementary information.

We see that the best prediction quality in Table 1 is attained by QC [40] which measures the variance of retrieval scores in the retrieved passage list, and U_SMV [42] which measures both the variance and average of the retrieval scores. Both predictors statistically significantly outperform all other predictors ; the differences between these two predictors are not statistically significant.

Finally, we note that the prediction quality of the post-retrieval predictors in Table 1 is quite lower than their prediction quality for ad hoc document retrieval — the task for which they were originally designed [5]. This is not a surprise as generation quality also depends on the LLM and the context used for generation.

4.2.2 Our novel post-retrieval predictors. Table 2 depicts the prediction quality of our two novel (supervised) post-retrieval predictors designed for the RAG setting: PredReranker and TripletPred[18]. We present for reference the prediction quality of the two best predictors from Table 1 which were originally devised for ad hoc retrieval: QC and U_SMV; both are unsupervised. In addition, we report the prediction quality of an XGBoost regressor, denoted **PostRetReg**, which integrates our two novel predictors, PredReranker and TripletPred. The prediction value assigned by each of the two methods is a feature in the regressor. Refer back to Section 4.1 for information about training the regressor

We see in Table 2 that, unsurprisingly, our novel predictors which were devised for the RAG setting, PredReranker and TripletPred, statistically significantly outperform those devised for ad hoc retrieval: QC and U_SMV. Furthermore, our predictors are supervised

[18] Recall that TripletPred is a family of predictors. Unless otherwise stated, TripletPred refers to the predictor that utilizes (G, S), the association between the passage and the suffix, and (G, P), the association between the passage and the prefix. This is the best performing TripletPred predictor as we show in Section 4.2.5.

Table 2: Prediction quality of our novel supervised predictors. The two best-performing predictors from Table 1 are presented for reference. The best result in a column is boldfaced. Statistically significant differences with PredReranker, TripletPred and PostRetReg are marked with 'p', 't' and 'r', respectively.

	Llama-3-8B	Falcon2-11B
QC	0.299^{pt}_{r}	0.27^{pt}_{r}
U_SMV	0.296^{pt}_{r}	0.27^{pt}_{r}
PredReranker	0.383^{t}_{r}	0.341^{t}_{r}
TripletPred	0.488	0.463
PostRetReg	**0.51**	**0.465**

Table 3: Prediction quality of the post-generation predictors The post-retrieval predictors serve for reference. 'e', 'd', 'g', 'a': statistically significant differences with EntPred, DiverPred, PostGenReg and PostRetGenReg, respectively. Boldface: the best result in a column

	Llama-3-8B	Falcon2-11B
PredReranker	0.383^{ed}_{ga}	0.341^{ed}_{ga}
TripletPred	0.488^{ed}_{ga}	0.463^{ed}_{ga}
PostRetReg	0.51^{ed}_{ga}	0.465^{ed}_{ga}
EntPred	0.589^{d}_{ga}	0.535^{d}_{ga}
DiverPred	0.687_{ga}	0.622_{ga}
PostGenReg	0.705_{a}	0.659_{a}
PostRetGenReg	**0.735**	**0.689**

(although trained for predicting GPT2-based RAG gain) while QC and U_SMV are unsupervised.

Table 2 shows that TripletPred statistically significantly outperforms PredReranker. PredReranker is the difference between the retrieval score assigned by a Predictive ReRanker [36] to the passage G and that assigned to the prefix (P) replaced by the passage. In contrast, TripletPred is trained to directly predict the gain in Equation 2 and accounts for both the association between the passage and the prefix and the association between the passage and the suffix. The best prediction quality in Table 2 is of PostRetReg which integrates our two novel predictors. PostRetReg statistically significantly outperforms all other predictors except for TripletPred[19].

4.2.3 Post-generation predictors. Table 3 depicts the prediction quality of the post-generation predictors, EntPred and DiverPred and their integration in a regressor, **PostGenReg**. We also present the prediction quality of our novel post-retrieval predictors, PredReranker and TripletPred, and their integration in the PostRetReg regressor. The **PostRetGenReg** regressor integrates the two post-retrieval predictors and the two post-generation predictors.

We see in Table 3 that the difference between the entropy of the next-token distribution induced without RAG and that induced with RAG (EntPred) and the KL divergence between the two distributions (DiverPred) are highly effective predictors; they statistically significantly outperform the post-retrieval predictors; DiverPred statistically significantly outperforms EntPred.

The findings just mentioned attest to the fact that (i) high entropy of a next token distribution implies to reduced text generation quality, probably due to the uncertainty which results in higher likelihood of error in the next-token generation, and (ii) increased divergence of the RAG-based next-token distribution from the no-RAG next-token distribution is an indicator for increased gain attained by applying RAG. Integrating EntPred and DiverPred in PostGenReg yields statistically significant improvements.

It is not a surprise that the post-generation predictors outperform the post-retrieval predictors as the former analyze the next-token distribution while the latter do not. On the other hand, post-generation predictors are significantly less efficient as they require (two) calls to the LLM in contrast to the post-retrieval predictors.

The best prediction quality in Table 3 is attained by PostRetGenReg which integrates the post-retrieval and the post-generation predictors, with very high correlation (0.735 for Llama-3-8B and 0.689 for Falcon2-11B). Furthermore, PostRetGenReg's prediction quality is statistically significantly better than that of all other predictors in Table 3. This finding leads us to the conclusion that post-retrieval and post-generation predictors are of complementary nature.

4.2.4 Selective RAG. Considering the high prediction quality reported above, we evaluate the effectiveness of some of the predictors for selective application of RAG: deciding for a given context (P, S) and the corresponding retrieved passage G, whether to apply RAG to induce the next token distribution; i.e., whether to use the context (G,S). The decision is based on the prediction of whether the gain in Equation 2 is greater than 0. This is a binary classification task per given context and retrieved passage.

We use the same cross-validation procedure and the same hyper-parameter value ranges described in Section 4.1 to train and evaluate classifiers rather than regressors. We use gradient boosted decision trees (XGBOOST [6]) for classifiers. The validation fold serves to set hyper-parameter values by minimizing the resultant perplexity over the contexts in the fold; that is, for each context the classifier decides whether to apply RAG and the resultant next-token distributions for the fold are evaluated using perplexity. For evaluation, we report the perplexity attained by a classifier over 10 test folds according to the per-context classifier's decision (with or without RAG). We note that whenever a classifier is built on top of a single feature, it is actually a thresholding criterion.

Perplexity results are presented in Table 4; lower perplexity indicates improved next-token distribution. We use the noRAG and alwaysRAG baselines where RAG is never applied and always applied, respectively. The oracle baseline selects for each context whether to apply RAG based on the *actual* merit (or lack thereof) of applying RAG.

The PredReranker classifer is a single feature using the PredReranker prediction value. The TripletPred classifier is also a single feature classifier which uses the output of the TripletPred predictor. Training TripletPred as a classifier (i.e., replacing the regression layer with a classification layer) resulted in worse perplexity. The **PostRetClass** classifier integrates our novel post-retrieval predictors: PredReranker and TripletPred.

EntPred and DiverPred are single feature classifiers using the post-generation predictors. The **PostGenClass** classifier integrates the post-generation predictors, EntPred and DiverPred. The **PostRetGenClass** classifier integrates the two post-retrieval predictors and the two post-generation predictors.

[19]Including in the regressor QC and U_SMV in addition to PredReranker and TripletPred showed no prediction quality merit.

Table 4: Perplexity of selective RAG. 'a' and 'p' mark statistically significant differences with alwaysRAG and PostRetClass, respectively. Boldface marks the best result in a column excluding oracle.

	Llama-3-8B	Falcon2-11B
noRAG	5.767	4.78
alwaysRAG	5.13	4.33
oracle	4.681^{a}	3.956^{a}
TripletPred	5.093^{a}_{p}	4.311
PredReranker	5.068^{a}	$\mathbf{4.29}^{a}$
PostRetClass	$\mathbf{5.064}^{a}$	$\mathbf{4.29}^{a}$
EntPred	5.075^{a}	4.318
DiverPred	5.077^{a}	4.318
PostGenClass	5.076^{a}	4.316
PostRetGenClass	5.07^{a}	4.307^{a}

Table 5: Prediction quality of the TripletPred predictors and the best post-retrieval predictors from Table 1. Boldface: the best result in a column. '$*$' marks a statistically significant difference with the best predictor: (G,S) + (G,P).

	Llama-3-8B	Falcon2-11B
QC	0.299^{*}	0.27^{*}
U_SMV	0.296^{*}	0.27^{*}
(G,S) + (P,S) + (G,P)	0.43^{*}	0.396^{*}
(G,S) + (P,S)	0.478	0.454
(G,S) + (G,P)	**0.488**	**0.463**
(P,S) + (G,P)	0.088^{*}	0.08^{*}
(G,S)	0.406^{*}	0.375^{*}
(P,S)	0.003^{*}	0.001^{*}
(G,P)	0.092^{*}	0.085^{*}

We see in Table 4 that applying RAG un-selectively (alwaysRAG) results in preplexity which is substantially better than that of not applying RAG at all (noRAG). However, alwaysRAG is substantially outperformed by the oracle which selectively applies RAG based on true performance. These findings are aligned with a previous report on the in-context RAG framework for text completion [36].

Table 4 shows that all post-retrieval and post-generation classifiers (and their combination) outperform alwaysRAG; many improvements are statistically significant. This attests to the merit in using the classifiers for selective RAG. We also see that most post-retrieval classifiers outperform the post-generation classifiers. The best perplexity is attained by the PostRetClass post-retrieval classifier; its performance difference with PostGenClass and PostRetGenClass is statistically indistinguishable.

These findings about post-retrieval classifiers posting perplexity on par with that of post-generation classifiers stands in clear contrast to our findings from above: post-generation prediction was clearly superior to post-retrieval prediction as a regression task evaluated using correlation over different contexts. The contrast can be explained as follows. Pearson correlation across contexts of predicted and true gain of applying RAG can be high even if for each context the predicted gain is shifted as long as the shift is consistent across contexts (e.g., using affine transformation). On the other hand, such shifts can be detrimental to classification performance.

The contrast just discussed is reminiscent of that reported in work on query performance prediction for ad hoc retrieval [15, 35]. That is, the order among predictors in terms of Pearson correlation measured across different queries for the same retrieval method, does not necessarily indicate their benefit in selecting the right retrieval method per query.

Finally, we note that the fact that post-retrieval classifiers post perplexity on par with that of post-generation classifiers is of utmost importance in terms of efficiency. That is, to decide whether to apply RAG in the text completion task, one presumably need not call the LLM but rather apply a post-retrieval classifier.

4.2.5 Deeper inside TripletPred. In Section 4.2.2 we showed that our TripletPred predictor was the best performing post-retrieval predictor. Recall that this is a specific predictor in the family of TripletPred predictors. The TripletPred predictors can utilize (i) (G,S): the association between the passage and the suffix, (ii) (G,P): the association between the passage and the prefix, and (iii) (P,S) the association between the prefix and the suffix. Table 5 depicts the prediction quality (Pearson correlation) of all TripletPred predictors.

We see in Table 5 that four out of the seven TripletPred predictors substantially outperform QC and U_SMV. The latter two were the best performing predictors originally devised for ad hoc retrieval in Table 1. The (G,S) + (G,P) predictor, insofar referred to as TripletPred, outperforms all other predictors in the table. All the improvements are statistically significant except for those with (G,S) + (P,S).

The (G,S) + (P,S) + (G,P) predictor is outperformed by the best predictor, (G,S) + (G,P), although they share two components. This means that the association between the prefix and the suffix, (P,S), which constitute the original context, is less important than the association of the prefix and the suffix with the passage; furthermore, adding (P,S) can evidently hurt prediction quality. The association between the prefix and the suffix is indirectly accounted for when using (G,S) and (G,P) as input to the DeBERTa model. Nevertheless, we found that (G,S) + (P,S) + (G,P) statistically significantly outperforms both QC and U_SMV for both LLMs.

Finally, we see in Table 5 that the lowest prediction quality is posted by (P,S) which is the only pre-retrieval predictor in the table; i.e., the only one that does not utilize the retrieved passage G.

5 Conclusions

We addressed a novel prediction challenge for LLMs: estimating the relative gain of applying RAG for the text completion task. We showed that pre-retrieval predictors originally devised for ad hoc retrieval are ineffective in this setting. Post-retrieval predictors designed for ad hoc retrieval post moderate prediction quality.

We devised novel supervised post-retrieval predictors that model different types of associations between parts of the input provided to the LLM. The resultant prediction quality substantially transcends that of all ad hoc retrieval predictors we studied. Integrating our novel predictors with post-generation predictors resulted in prediction quality that is statistically significantly better than that of the post-generation predictors. Furthermore, our post-retrieval predictors are as effective as post-generation predictors for selective application of RAG, while they are significantly more efficient.

Acknowledgments. We thank the reviewers for their comments. The work presented in the paper was supported in part by the Israel Science Foundation (grant no. 403/22).

References

[1] Negar Arabzadeh, Maryam Khodabakhsh, and Ebrahim Bagheri. 2021. BERT-QPP: Contextualized Pre-trained transformers for Query Performance Prediction. In *Proceedings of CIKM*. 2857–2861.

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. [n. d.]. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *Proceedings of ICLR*.

[3] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In *Proceedings of ICML*. 2206–2240.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] D. Carmel and E. Yom-Tov. 2010. *Estimating the Query Difficulty for Information Retrieval*. Morgan & Claypool Publishers.

[6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *CoRR* abs/1603.02754 (2016).

[7] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for RAG systems. In *Proceedings of SIGIR*. 719–729.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[9] Yujuan Ding, Yunshan Ma, Wenqi Fan, Yige Yao, Tat-Seng Chua, and Qing Li. 2024. Fashionregen: Llm-empowered fashion report generation. In *Companion Proceedings of the ACM on Web Conference 2024*. 991–994.

[10] Guglielmo Faggioli, Thibault Formal, Stefano Marchesin, Stéphane Clinchant, Nicola Ferro, and Benjamin Piwowarski. 2023. Query Performance Prediction for Neural IR: Are We There Yet?. In *Advances in Information Retrieval*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer Nature Switzerland, Cham, 232–248.

[11] Chunjing Gan, Dan Yang, Binbin Hu, Hanxiao Zhang, Siyuan Li, Ziqi Liu, Yue Shen, Lin Ju, Zhiqiang Zhang, Jinjie Gu, et al. 2024. Similarity is Not All You Need: Endowing Retrieval Augmented Generation with Multi Layered Thoughts. *arXiv preprint arXiv:2405.19893* (2024).

[12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL]

[13] Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, and Deyi Xiong. 2023. Evaluating Large Language Models: A Comprehensive Survey. arXiv:2310.19736 [cs.CL]

[14] Helia Hashemi, Hamed Zamani, and W Bruce Croft. 2019. Performance prediction for non-factoid question answering. In *Proceedings of SIGIR*. 55–58.

[15] Claudia Hauff, Leif Azzopardi, Djoerd Hiemstra, and Franciska de Jong. 2010. Query performance prediction: Evaluation contrasted with effectiveness. In *Proceedings of ECIR*. 204–216.

[16] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. 2008. A survey of pre-retrieval query performance predictors. In *Proceedings of SIGIR*. 1419–1420.

[17] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. arXiv:2006.03654 [cs.CL]

[18] Oz Huly, Idan Pogrebinsky, David Carmel, Oren Kurland, and Yoelle Maarek. 2024. Old IR Methods Meet RAG. In *Proceedings of SIGIR*. 2559–2563.

[19] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research* 24, 251 (2023), 1–43.

[20] Nasreen Abdul Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of TREC*, Vol. 500-261.

[21] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 7029–7043.

[22] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. In *Proceedings of EMNLP*. 7969–7992.

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

[24] Youna Kim, Hyuhng Joon Kim, Cheonbok Park, Choonghyun Park, Hyunsoo Cho, Junyeob Kim, Kang Min Yoo, Sang-goo Lee, and Taeuk Kim. 2024. Adaptive Contrastive Decoding in Retrieval-Augmented Generation for Handling Noisy Contexts. In *Findings of the Association for Computational Linguistics: EMNLP*. 2421–2431.

[25] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation. In *Proceedings of ICLR*.

[26] K. L. Kwok. 1996. A new method of weighting query terms for ad-hoc retrieval. In *Proceedings of SIGIR*. 187–195.

[27] Victor Lavrenko and W. Bruce Croft. 2001. Relevance-Based Language Models. In *Proceedings of SIGIR*. 120–127.

[28] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[29] 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 SIGIR*. 2356–2362.

[30] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceeding of ACL*. 9802–9822.

[31] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Query Performance Prediction using Relevance Judgments Generated by Large Language Models. *CoRR* abs/2404.01012 (2024).

[32] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. In *Proceedings of ICLR*.

[33] Ella Rabinovich, Samuel Ackerman, Orna Raz, Eitan Farchi, and Ateret Anaby-Tavor. 2023. Predicting Question-Answering Performance of Large Language Models through Semantic Consistency. In *Proceedings of EMNLP*.

[34] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[35] Fiana Raiber and Oren Kurland. 2014. Query-performance prediction: setting the expectations straight. In *Proceedings of SIGIR*. 13–22.

[36] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics* (2023). https://arxiv.org/abs/2302.00083

[37] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. 1992. Okapi at TREC. In *Proceedings of TREC*. 21–30.

[38] Haggai Roitman, Shai Erera, and Guy Feigenblat. 2019. A study of query performance prediction for answer quality determination. In *Proc. of SIGIR*. 43–46.

[39] Anna Shtok, Oren Kurland, and David Carmel. 2016. Query Performance Prediction Using Reference Lists. *ACM Trans. Inf. Syst.* 34, 4, Article 19 (2016).

[40] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. 2012. Predicting Query Performance by Query-Drift Estimation. *ACM Trans. Inf. Syst.* 30, 2 (2012), 11:1–11:35.

[41] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. 3784–3803.

[42] Yongquan Tao and Shengli Wu. 2014. Query Performance Prediction By Considering Score Magnitude and Variance Together. In *Proceedings of CIKM*. 1891–1894.

[43] Xi Wang, Procheta Sen, Ruizhe Li, and Emine Yilmaz. 2024. Adaptive Retrieval-Augmented Generation for Conversational Systems. *arXiv preprint arXiv:2407.21712* (2024).

[44] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-Knowledge Guided Retrieval Augmentation for Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 10303–10315.

[45] Zora Zhiruo Wang, Akari Asai, Xinyan Velocity Yu, Frank F Xu, Yiqing Xie, Graham Neubig, and Daniel Fried. 2024. CodeRAG-Bench: Can Retrieval Augment Code Generation? *arXiv preprint arXiv:2406.14497* (2024).

[46] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.* 28, 4, Article 20 (2010).

[47] XGBOOST [n. d.]. https://xgboost.readthedocs.io/en/stable/parameter.html.

[48] Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2024. A Theory for Token-Level Harmonization in Retrieval-Augmented Generation. arXiv:2406.00944 [cs.CL] https://arxiv.org/abs/2406.00944

[49] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884* (2024).

[50] Chengyuan Yao and Satoshi Fujita. 2024. Adaptive Control of Retrieval-Augmented Generation for Large Language Models Through Reflective Tags. *Electronics* 13, 23 (2024), 4643.

[51] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. [n. d.]. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

[52] Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020. Knowledge-Grounded Dialogue Generation with Pre-trained Language Models. In *Proceedings of EMNLP*. 3377–3390.
[53] Ying Zhao, Falk Scholer, and Yohannes Tsegay. 2008. Effective Pre-retrieval Query Performance Prediction Using Similarity and Variability Evidence. In *Proceedings of ECIR*.
[54] Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2024. Recommender Systems in the Era of Large Language Models (LLMs). *IEEE Transactions on Knowledge and Data Engineering* 36, 11 (2024).
[55] Yun Zhou and W. Bruce Croft. 2007. Query performance prediction in web search environments. In *Proceedings of SIGIR*. 543–550.