Generative model for Retrieval and Their Applications

A MS thesis first semester report submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science, Master of Science
in
Data Science and Engineering
by

Kavin R V
Roll No.: 19163

Under the Supervision of
Dr. Maunendra Sankar Desarkar (IIT Hyderabad)

Department of Data Science and Engineering
Indian Institute of Science Education and Research Bhopal
Bhopal - 462066, India

November, 2023

ABSTRACT

Generative retrieval models present a promising avenue in the field of neural information retrieval, outperforming established methods like Best Matching 25 (BM25) and Dense Passage Retrieval (DPR). These models employ a single transformer for document indexing and retrieval, generating results based on identifiers such as docid, titles, or substrings. Despite their advantages, these generative retrievers encounter challenges that hinder their practical application. One notable issue is their difficulty in seamlessly incorporating new documents. Since all information about a specific document is stored within the model's parameters, addressing catastrophic forgetting requires retraining the model on the entire corpus. This research delves into potential techniques to overcome these limitations, demonstrating their effectiveness on less resource-intensive tasks formulated similarly to retrieval, such as Few Shot Intent Detection and Re-ranking.

Contents

Contents

List of Figures

List of Figures

List of Tables

List of Tables

1 Introduction

Generative Retrieval represents a novel paradigm in the realm of neural Information Retrieval. These models take a query as input and autoregressively produce a sequence of tokens that uniquely corresponds to a document in the corpus. We can categorize these models based on the type of tokens they generate. Early efforts primarily focused on generating titles (De Cao et al. 2020), substrings, and keywords (Bevilacqua et al. 2022). Recently, the Differentiable Search Index (DSI) (Tay et al. 2022) introduced a groundbreaking approach to information retrieval. DSI employs a single encoder-decoder model to autoregressively generate a docid, which serves as a unique identifier for a particular document, producing a ranked list in response to a query. Expanding on the success of DSI, the Neural Corpus Indexer (Wang et al. 2022) demonstrated that docid-based models can surpass both state-of-the-art generative retrieval models and dense retrieval models.

These models heavily depend on the knowledge stored within their parameters, posing challenges when adding new documents without jeopardizing prior knowledge through catastrophic forgetting. Furthermore, when adding new documents, the model is presented solely with the documents, lacking the query input. This necessitates the model to learn the document semantics without the aid of a query during training. Contrastive learning, frequently employed by dense retrieval models (Karpukhin et al. 2020), proves effective in zero-shot scenarios. Domain adaptation techniques may also prove valuable, aiding the model in learning distinct information from query and document inputs during training.

While scoring and contrastive learning are straightforward for dense retrieval, assessing the sequence of text is non-trivial. We propose a method to score sequences of text through a simplified task: re-ranking passages for dialogue systems, involving generative passage estimation for text sequence scoring. Subsequently, we illustrate the impact of these techniques in the implementation of the recreated version of NCI(Wang et al. 2022)

2 Background

2.1 Re Ranking

Passage Ranking: Typically ranking model use cross attention based model where input is a sequence of concatenated query and candidate passage and fed into the model. This model stucture is mostly only explored for BERT-like encoder only models. RankT5 (Zhuang et al. 2023) introduces a different approach that out performs previous approaches. Recent State-of-Art Conversation model like GripRank(Bai et al. 2023), QKConv (Cai et al. 2022) have shown adding a Re-Rank component to the model significantly boosts the performace of these model, improves the quality of dialogue generation. Also most of the previous works have only used the previous conversation as query for the re-ranking/retrieval, but using them as a query would bring some complications like for example the conversation with a several topics could confuse such models, also using only the last conversation won't be helpfull when the question is based on some statement already made deep in conversation.

Knowledge Distillation: Recent models for Knowledge intensive task like FiD(Izacard and Grave 2020), GripRank(Bai et al. 2023), has shown that distilling knowledge from Reader model to revtriever model significantly helps to boost their performance. Different from these model we introduce a way to utilize self distillation specific for conversation models.

2.2 Information Retrieval

Retrieval is a fundamental part of many Knowledge Intensive NLP tasks. The task involves retrieving relevant information from a large corpus of knowledge based on a given query. Formally, given a query q and a corpus C, the goal is to retrieve the top k documents from C that are most relevant to q, where relevance is typically measured using a similarity metric between the query and documents in the corpus. Retrieval is often done using a combination of lexical matching, vector similarity, and neural methods. ***Sparse Retrievers*** are a type of retrieval

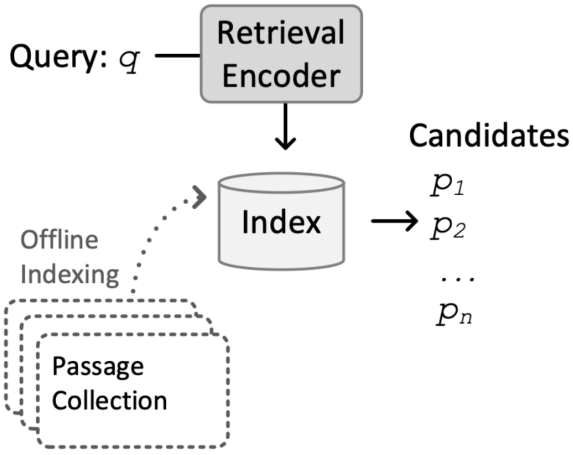

Figure 1: Retrieval Framework

method that use explicit lexical matches between the query and the documents in the corpus. In other words, they retrieve documents that contain specific keywords or phrases that match the query. One common approach is to use an inverted index, which is a data structure that stores all the terms in the corpus along with the list of documents in which they appear. Given a query, the inverted index can quickly retrieve the documents that contain the query terms. Some examples of sparse retrieval methods include TF-IDF, and BM25.

TF-IDF:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

where

$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{i \in d} f_{i,d}}$$

and

$$\text{idf}(t, D) = \log \frac{N}{n_t}$$

BM25:

$$\text{BM}(d, q) = \sum_{i=1}^{n} \text{idf}(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}$$

where

$$\text{idf}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

here q_i represents the frequency of term i in the query q, D represents the collection of documents, b is a free parameter that controls the scaling of the term frequency, and $avgdl$ is the average length of documents in D, N is the total number of documents, and $n(q_i)$ is the number of documents containing the i-th term in the corpus.

Dense Retrievers are a type of retrieval method that use transformer based model to encode the query and the documents in the corpus into dense vectors, and then retrieve documents that are most similar to the query vector. Unlike sparse retrieval methods, dense retrieval methods can capture semantic similarity between the query and documents, and are more effective at retrieving relevant documents even when they do not contain exact matches to the query terms. Here indexing(i.e. converting documents to vector) doesn't need to online, these can be pre-computed and stored to later access them while searching. Some examples of dense retrieval methods include dense passage retrieval(Karpukhin et al., 2020) and dual-encoder models(Reimers and Gurevych, 2019).

Dense retrievers, unlike sparse retrievers, use dense vector representations of text to perform similarity search. In dense retrievers, the document and the query are first encoded as dense vectors, usually using a neural network, and then the similarity between the two vectors is computed using some distance metric such as cosine similarity. These dense vectors capture the semantics of the text, and hence, can provide better retrieval performance than sparse retrievers. One common approach for dense retrieval is to use pre-trained language models such as BERT, RoBERTa, and others to encode the text. The advantage of this approach is that pre-trained models can capture rich and complex semantic relationships between words, which is beneficial for downstream retrieval tasks. The disadvantage is that computing the dense vector representations can be computationally expensive and slow, especially when the number of documents is large. The BERT representation of a document d is computed as $b_d = BERT(d)$, and the cosine similarity score between the query and the document is obtained as $S(b_q, b_d) = \text{COS}(b_q, b_d) = \frac{b_q \cdot b_d}{\|b_q\|_2 \|b_d\|_2}$.

2.3 Generative Retrieval

Generative retrieval is a type of retrieval technique that generates text sequences as a response to a query rather than retrieving pre-existing text. In this approach, a language model is trained to generate text that is relevant to the query. The training process usually involves fine-tuning a pre-trained language model such as BART(Lewis et al. 2019) or T5(Raffel et al. 2020) on a large corpus of text and a set of queries. During inference, the model generates a text sequence that is conditioned on the query. Generative retrieval has the advantage of being able to produce more diverse and fluent responses than retrieval-based models. However, it also has some challenges such as generating relevant and coherent responses, avoiding repetition and maintaining factual accuracy. One approach to mitigate these challenges is to use techniques such as controlled generation, which involves guiding the language model to generate text that follows certain constraints, by using data stuctures like prefix tree(aka trie, T). In prefix tree, nodes are annotated with tokens from the vocabulary. For each node $t \in T$, its children indicate all the allowed continuations from the prefix defined traversing the trie from the root to t. Docid based models like DSI(Tay et al. 2022), NCI(Wang et al. 2022) rely on semantic docid based tree structure for constrained generation, this will be explained in later section.

3 Methodology

3.1 Foresight- Re Ranker

3.1.1 Problem Formulation

In our methodology, we address the task of re-ranking a list of passages to improve their ordering based on their relevance to a given query. Specifically, we are provided with a set of retrieved passages denoted as $P = \{p_1, p_2, \ldots, p_k\}$, where each p_i represents a passage, and k signifies the number of retrieved passages. These passages are associated with a given query denoted as q. The objective of the re-ranker is to generate a re-ordered list of passages denoted as $P' = \{p'_1, p'_2, \ldots, p'_k\}$, such that the new order better reflects the relevance of passages to the query. To accomplish this task, we introduce a re-ranker denoted as $\mathcal{R}(P'|q, P; \theta_{\mathcal{R}})$, which is parameterized by $\theta_{\mathcal{R}}$. The re-ranker takes as input the query q, the original list of retrieved passages P, and the parameterization $\theta_{\mathcal{R}}$.

3.1.2 Masked Response as Query

In this section, we introduce our novel approach of employing a masked response as the query for the re-ranking process. The rationale behind this technique is to enhance the re-ranking process's efficacy by focusing on key information within the conversation. To implement this approach, we follow a multi-step procedure.

First, we initiate the creation of the masked response by utilizing an unsupervised keyword extractor called YAKE Campos et al. 2020. The output of this step is a set of extracted keywords denoted as $K = \{k_1, \ldots, k_n\}$, extracted from the original response. For the masking process we consider the history of previous utterances denoted as $U = \{u_1, \ldots, u_t\}$, along with the relevant passage p_r. The masking process involves determining whether a particular keyword k_x should be masked in the response $r = u_{t+1}$. This decision is based on the following criteria: if the keyword k_x is absent in all previous utterances u_y within U, and if the keyword k_x is present in the relevant passage p_r. If both conditions are met, the keyword k_x is masked in the response r.

The result of this masking procedure is the creation of a masked response, denoted as r_{masked}. This masked response retains key contextual information while focusing on elements that have not been extensively covered in prior dialogue. For the re-ranking task, this masked response r_{masked} is combined with the last utterance u_t to form an effective query denoted as $q = \{u_t, r_{\text{masked}}\}$.

3.1.3 Passage Ranker

In our ranking methodology, we employ an architecture inspired by RankT5 Zhuang et al. 2023 to effectively rank passages based on their relevance to given queries. This architecture, compared to the traditional Cross-Encoder model used for re-ranking, has demonstrated significant performance improvements, along with the added capability of generation, which is a key aspect discussed in Section 3.1.4

For a given i-th query q_i and it's j-th retrieved passage p_{ij}, the input representation s_{ij} is structured as follows:

$$s_{ij} = \text{question: } q_i \text{ context: } p_{ij}$$

This architecture can be viewed as a simple adaptation of the T5 model Raffel et al. 2020, focusing on the first output token from the decoder. By passing the input through the T5 model, we obtain unnormalized logits z across the entire vocabulary:

$$z = \text{RankHead}(\text{Dec}(\text{Enc}(s_{ij})))$$

Here, RankHead is a dense layer that projects the output hidden state to the vocabulary space $\mathcal{V}$. Notably, the absence of a softmax layer over the vocabulary results in arbitrary real numbers in the logits z. Additionally, since our objective is ranking and not token generation, the need for previous tokens $t_{1:k-1}$ is eliminated.

To determine the ranking score for a given pair of query q_i and passage p_{ij}, we specify an unused token in the T5 vocabulary, denoted as "<extra_id_80>". The corresponding unnormalized logits for this token are taken as the ranking score:

$$\hat{y}_{ij} = z_{\text{<extra\_id\_80>}}$$

Here, $z_{\{w\}}$ represents the logits corresponding to the token $w \in \mathcal{V}$. For this objective we use a listwise softmax cross entropy lossBruch et al. 2019; Burges et al. 2005 based on the predicted scores $\hat{y}_i$ and relevance labels y_i.

$$l_{\text{ranking}}(y_i, \hat{y}_i) = -\sum_{j=1}^{m} y_{ij} \log \left(\frac{e^{\hat{y}_{ij}}}{\sum_{j'} e^{\hat{y}_{ij'}}} \right)$$

3.1.4 Keyword Estimation and Self Distillation

For knowledge distillation we take logit score of all the keywords that are masked in response from the decoder output. So for t-th keyword k_t we calculate keyword estimation score $\hat{z}^t$ of t-th token

$$\hat{z}^t = \text{LMHead}(\text{Dec}(\text{Enc}(s_{ij}), k_{1:t-1}))$$

Notice, we again only obtain unnormalized logits $\hat{z}^t$ across the entire vocabulary. To get the estimation score of keyword k_t

$$g_{ij}^{k_t} = \hat{z}^t_{\{k_t\}}$$

And we to obtain the overall keyword estimation score of a passage we sum g_{ij}^{kt} over all the keywords.

$$g_{ij} = \sum_t g_{ij}^{k_t}$$

Based on this we experiment with two different losses, listwise softmax cross entropy loss and KL divergence loss.

$$l_{\text{KE}}(y_i, g_i) = -\sum_{j=1}^{m} y_{ij} \log \left(\frac{e^{g_{ij}}}{\sum_{j'} e^{g_{ij'}}} \right)$$

and

$$l_{\text{KLKE}}(y_i, \hat{y}_i, g_i) = \text{KLDiv}(y_i || g_i)$$

we train the model two losses, $l_1 = l_{\text{ranking}}(y_i, \hat{y}_i) + l_{\text{KE}}(y_i, g_i)$ and $l_2 = l_{\text{ranking}}(y_i, \hat{y}_i) + l_{\text{KLKE}}(y_i, \hat{y}_i, g_i)$

3.1.5 Dataset

We use Wizard of Wikipedia Dinan et al. 2018 dataset for all our experiments. In training split of this data contains around 18000 conversations between wizard and apprentice. Each turn contains a knowledge pool containing 7 passage that was retrieved based on the previous conversation and it also contains the label for the correct passage only for wizards turns/responses. So after cleaning the data and restructuring it to contain all the response from wizards as output we had dataset with around 74000 training examples.

3.1.6 Implementation Details

We Initialize all our model with T5-base Checkpoint. For all the models we set the token size as 512. And they are all trained with batch size of 4 and gradient accumulation steps of 8, so an effective batch size of 32 on a single V100 32GB GPU for 2 epochs.

3.2 NCI Contra

Major contributions of NCI paper include Semantic docid, Question generation and a Prefix-Aware Weight-Adaptive (PAWA) Decoder. Each of these components are explained in the following subsections. We do all the training and experiments with two popular open source question answering datasets called Natural Questions (Kwiatkowski et al. 2019) and TriviaQA (Joshi et al. 2017). Natural Questions (NQ) was introduced by Google in 2019. The version they used consists of 320k query-document pairs, where the documents are gathered from Wikipedia pages and the queries are natural language questions. TriviaQA is a reading comprehension dataset, which includes 78k query-document pairs from the Wikipedia domain. Unlike the NQ dataset, a query may include multiple answers in TriviaQA.

3.2.1 Semantic Docid

NCI leverage hierarchical k-mean algorithm to encode document. As shown in figure 2 given a collection of documents to be indexed, all documents are first classified into k clusters by using their representations encoded by BERT (Devlin et al. 2018). For cluster with more than c documents, the k-means algorithm is applied recursively. For each cluster containing c documents or less, each document is assigned a number starting from 0 to at most c-1. In this way, they organize all documents into a tree structure T with root r_0. Each document is associated with one leaf node with a deterministic routing path $l = \{r_0, r_1, ..., r_m\}$ from the root, where $r_i[0, k)$ represents the internal cluster index for level i, and $r_m[0, c)$ is the leaf node. The semantic identifier for a document is concatenated by the node indices along the path from root to its corresponding leaf node. For documents with similar semantics, the prefixes of their corresponding identifiers are likely to be the same. They set $k = 30$ and $c = 30$ in all experiments.

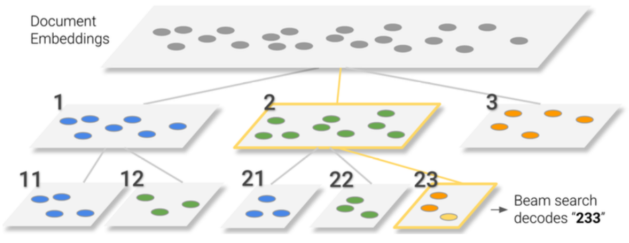

Figure 2: Hierarchical K-means Clustering for Semantic Docid Creation.

3.2.2 Question Generation

They use DocT5Query which takes as input the document terms and produces relevant queries via random sampling. Addition to this they also utilize the first 64 terms for each document as queries. Besides, we randomly selected 10 groups of 64 consecutive terms from the whole article as additional queries. This makes the NCI model aware of the semantic meaning of each document.

3.2.3 PAWA Decoder

Differing from DSI, NCI uses docid in a way that same token at different position represents different things. This is done by giving an entirely new identity to the same token in a different position. So for a docid 343, NCI represents it as (3, 1), (4, 2), (3, 3) so (3, 1) and (3, 3) doesn't represent the same thing. The decoder also uses an adaptive weight which is aware of the prefix for mapping the hidden vector to the vocab as shown in the following equation. The overall picture of this decoder is shown in figure 3

$$W^i_{ada} = \text{AdaptiveDecoder}(e; r_1, r_2, ..., r_{i-1})W_i$$

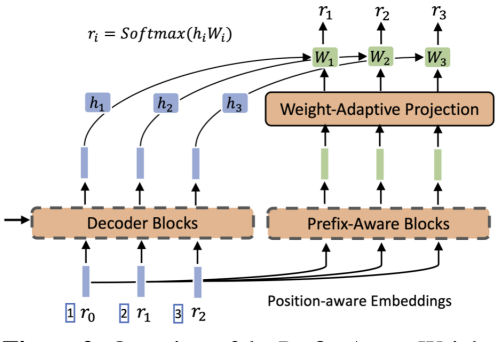

Figure 3: Overview of the Prefix-Aware Weight- Adaptive (PAWA) Decoder.

All the experiments in the following subsection are done on a trimmed down version of Natural Question. NQ dataset consist of of around 68k document, in trimmed down version we only have about 10k documents. We keep all the documents respective question for training. Also constructed a two Evaluation datasets, eval_normal containing only questions that have the respective document's question that are present in the training set. And another one called eval_zero which contains questions relevant to documents that are not present in the training set. Also augment the training with 10 random 64 tokens of all documents present in both training and eval_zero.

3.2.4 Curriculum Based Contrastive Learning

For contrastive learning experimented with two types of negatives, first one where we have 9 randomly selected

Table 1: Re-ranking performance using query as last utterance and masked response compared to only using last utterance

	MRR	Recall@			NDCG@		
		1	2	3	1	2	3
RankT5 Last utterance only	88.57	81.12	91.59	95.29	81.12	87.72	89.58
RankT5 Without KE	94.79	90.94	97.09	98.39	90.94	94.82	95.47
RankT5 With KE	94.79	91.00	96.93	98.39	91.00	94.74	95.47
RankT5 With KLKE	94.96	91.29	96.98	98.56	91.29	94.88	95.67

Table 2: Re-ranking performance using query as last utterance and generated masked response compared to only using last utterance

	MRR	Recall@			NDCG@		
		1	2	3	1	2	3
RankT5 Last utterance only	88.57	81.12	91.59	95.29	81.12	87.72	89.58
RankT5 Without KE	85.29	76.14	88.08	93.33	76.14	83.68	86.30
RankT5 With KE	85.65	76.41	89.04	93.99	76.41	84.38	86.85
RankT5 With KLKE	86.56	77.97	89.80	93.88	77.97	85.43	87.47

negatives. And for curriculum learning in initial training data negatives are ones that are completely irrelevant but gradually as the training proceeds more similar documents are added as part of the negatives.
For the contrative learning score of jth passage for the ith question is the probability of generating the docid, i.e. $g_{ij} = \prod_{k=0}^{n} P(d_{ij}^k)$ where d_{ij}^k is the kth token of the predicted docid. The contra loss has been shown in the following equation, $y_{ij} = 1$ if jth passage is provenance to the ith question.

$$l_{\text{contra}}(y_i, g_i) = -\sum_{j=1}^{m} y_{ij} \log \left(\frac{e^{g_{ij}}}{\sum_{j'} e^{g_{ij'}}} \right)$$

All the experiments in this section are conducted without the use of PAWA decoder and query generation and trained for 5 epochs to reduce the time taken to train the model and conduct experiments. For the baseline, result of model without the contra is also shown along with other variations in the table 3.

4 Results and Discussion

4.1 Foresight

To compare the proposed model we train RankT5 using only the last utterance as query. We also generate masked response given the previous conversation, using a simple conditional generation using T5. The result from this generated response is shown in Table 2. From Table 1 we can see using masked response significantly improves the performance. Also while not helpful in Table1 Keyword Estimation seems to produce better result when using the generated masked response when compared to one without any KE loss. All the metrics are calculated using Torchmetrics.

4.2 NCI Contra

Based on the results from Table 3 we can see little to no improvement on the performance for normal evaluation while using contrastive loss, but we could around 1% improvement in performance while using contrastive loss in zero-shot evaluation. We could also see that significant improvement in performance while curriculum learning by about 2% in both normal and zero-shot evaluation.

4.3 Evaluation Metrics

Recall@n is the standard evaluation metric for the retrieval task, this is formulated as follows:

$$Recall = \frac{|\textit{(relevant docs)} \cap \textit{(retrieved docs)}|}{|\textit{(relevant docs)}|}$$

In this case $|\textit{(relevant article)}| = 1$ for all the query will makes $Recall = |\textit{(relevant article)} \cap \textit{(retrieved article)}|$, and this recall will be "1" if the model retrieves the relevant article and "0" otherwise. R@k accounts for total article retrieved by the model which k, it is controlled by user. Recall in this case can also be consider as accuracy as for R@k the model will retrieve k articles and we score 1 if the relevant article is present in the k article retrieved and "0" other wise this similar to the accuracy metric.

Mean Reciprocal Rank (MRR) is a metric used to evaluate the effectiveness of search engines, recommendation systems, and information retrieval systems. It is particularly common in the context of ranking-based evaluations. MRR is defined as the average of the reciprocals of the ranks of the first correct item.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

Where:

Table 3: Results on Natural Questions with Contrastive and Curriculum Learning

	Normal Evaluation				Zero-Shot Evaluation			
	R@1	R@10	R@100	MRR@100	R@1	R@10	R@100	MRR@100
Without Contra	56.34	78.93	86.54	69.06	44.21	69.78	82.14	62.67
With Contra	56.51	79.18	86.62	68.99	46.02	70.32	82.78	64.31
Curriculum	58.13	80.84	88.04	71.17	48.37	72.15	83.81	66.00

- $|Q|$ the total number of queries or instances.
- $rank_i$ is the rank of the first correct item for the i-th query.

Normalized Discounted Cumulative Gain (nDCG) is another metric commonly used to evaluate the quality of a ranked list, especially in information retrieval and recommendation systems. It takes into account both the relevance and the position of items in the ranked list.

The formula for nDCG is as follows:

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

Where:

- DCG@k$DCG@k$ is the Discounted Cumulative Gain at position kk,
- IDCG@k$IDCG@k$ is the Ideal Discounted Cumulative Gain at position kk.

The Discounted Cumulative Gain is calculated as:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

And the Ideal Discounted Cumulative Gain is obtained by sorting the true relevance scores in descending order:

$$IDCG@k = \sum_{i=1}^{k} \frac{2^{rel_{(i)}} - 1}{\log_2(i+1)}$$

In these formulas:

- rel_i is the relevance of the item at position i
- $rel_{(i)}$ is the relevance of the i-th item in the ideal ranking.

5 Conclusion And Future Work

The findings presented above underscore the importance of incorporating contrastive and curriculum learning techniques in the context of generative retrieval. The forthcoming research will primarily center on the development and refinement of these techniques, with the aim of enhancing the robustness of generative models. The focus is on formulating strategies that enable these models to seamlessly adapt to new documents without succumbing to the detrimental effects of catastrophic forgetting. This research strives to contribute novel insights and methodologies that foster the continual evolution and adaptability of generative retrieval models in dynamic information environments.

6 Acknowledgement

I would like to express my sincere gratitude to my principle advisor Dr. Maunendra Sankar Desarkar for his invaluable guidance throughout this research project. I am thankful for his assistance in providing relevant literature and datasets, and for his insightful feedback and suggestions. I would also like to thank my internal adviser Dr. Jasabanta Patro for his valuable inputs and feedbacks.

References

J. Bai, H. Guo, J. Liu, J. Yang, X. Liang, Z. Yan, and Z. Li. Griprank: Bridging the gap between retrieval and generation via the generative knowledge improved passage ranking. *arXiv preprint arXiv:2305.18144*, 2023.

M. Bevilacqua, G. Ottaviano, P. Lewis, W.-t. Yih, S. Riedel, and F. Petroni. Autoregressive search engines: Generating substrings as document identifiers. *arXiv preprint arXiv:2204.10628*, 2022.

S. Bruch, X. Wang, M. Bendersky, and M. Najork. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, page 75–78, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368810. doi: 10.1145/3341981.3344221. URL `https://doi.org/10.1145/3341981.3344221`.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

M. Cai, S. Bao, X. Tian, H. He, F. Wang, and H. Wu. Query enhanced knowledge-intensive conversation via unsupervised joint modeling. *arXiv preprint arXiv:2212.09588*, 2022.

R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 509(C):257–289, jan 2020. ISSN 0020-0255. doi: 10.1016/j.ins.2019.09.013. URL `https://doi.org/10.1016/j.ins.2019.09.013`.

N. De Cao, G. Izacard, S. Riedel, and F. Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

G. Izacard and E. Grave. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*, 2020.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Y. Tay, V. Tran, M. Dehghani, J. Ni, D. Bahri, H. Mehta, Z. Qin, K. Hui, Z. Zhao, J. Gupta, et al. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843, 2022.

Y. Wang, Y. Hou, H. Wang, Z. Miao, S. Wu, Q. Chen, Y. Xia, C. Chi, G. Zhao, Z. Liu, et al. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614, 2022.

H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313, 2023.