A Neural Corpus Indexer for Document Retrieval

Anonymous Author(s)
Affiliation
Address
email

Abstract

Current state-of-the-art document retrieval solutions mainly follow an index-retrieve paradigm, where the index is hard to be optimized for the final retrieval target. In this paper, we aim to show that an end-to-end deep neural network unifying training and indexing stages can significantly improve the recall performance of traditional methods. To this end, we propose Neural Corpus Indexer (NCI), a sequence-to-sequence network that generates relevant document identifiers directly for a designated query. To optimize the recall performance of NCI, we invent a prefix-aware weight-adaptive decoder architecture, and leverage tailored techniques including query generation, semantic document identifiers and consistency-based regularization. Empirical studies demonstrated the superiority of NCI on a commonly used academic benchmark, achieving +51.9% relative improvement on NQ320k dataset compared to the best baseline.

1 Introduction

Document retrieval and ranking are two key stages for a standard web search engine [46, 27]. First, the document retrieval stage retrieves candidate documents relevant to the query, and then, the ranking stage gives a more precise ranking score for each document. The ranking stage is often fulfilled by a deep neural network, taking each pair of query and document as input and predicting their relevance score. Nevertheless, a precise ranking model is very costly, while typically only a hundred or thousand candidates per query are affordable in an online system. Therefore, the recall performance of document retrieval stage is very crucial to the effectiveness of web search engine.

Existing document retrieval methods can be divided into two categories, namely *term-based* and *semantic-based* approaches [18]. Term-based retrieval approaches [8, 48] build an inverted index for the entire web corpus, but they hardly capture document semantics and fail to retrieve similar documents in different wordings. Thus, semantic-based approaches [46, 29] are proposed to alleviate this discrepancy. First, they learn dense representations for both queries and documents through a twin-tower architecture; then Approximate Nearest Neighbor (ANN) search is applied to retrieve relevant documents for the designated query. Despite of their success in real applications, these approaches can not fully leverage the power of deep neural networks for the following reasons. First, a single embedding vector has limited capacity to memorize all semantics in a document, and it performs even worse than term-based methods in applications that heavily rely on exact match [30]. Second, the model is unable to incorporate deep query-document interactions. Because ANN algorithms theoretically require a strong assumption for the Euclidean space, we have to adopt simple functions such as cosine similarity to capture the query-document interactions [16].

Given the above limitations, several research works have explored end-to-end models that directly retrieve relevant candidates without using an explicit index. Gao et al. [16] proposed a Deep Retrieval (DR) framework for item recommendation, which learned a retrievable structure with historical user-item interactions. Nevertheless, it is more challenging to design a universal model for semantic text retrieval, as we need to leverage the power of both pre-trained language models and deep retrieval

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

networks simultaneously. Tay et al. [41] proposed Differentiable Search Index (DSI), a text-to-text model that maps queries directly to relevant docids. To the best of our knowledge, this is the first attempt to propose a differentiable index for semantic search. However, the vanilla transformer decoder in DSI does not fully leverage the hierarchical structures of document identifiers, and the model is pruned to over-fitting with limited training data. Furthermore, Bevilacqua et al. [4] proposed SEAL by leveraging all n-grams in a passage as its identifiers. But for long documents, it is hard to enumerate all possible n-grams. In general, the recall performance of end-to-end document retrieval remains a large room to be improved.

In this paper, we show that the traditional text retrieval frameworks can be fundamentally changed by a unified deep neural network with tailored designs. To this end, we propose a Neural Corpus Indexer (NCI), which supports end-to-end document retrieval by a sequence-to-sequence neural network. The model takes user query as input, generates query embedding through the encoder, and outputs the identifiers of relevant documents using the decoder. It can be trained by ground-truth and augmented query-document pairs. During inference, the top N documents are retrieved directly via beam search based on the decoder. Designing and training such a model is non-trivial, so we propose several crucial techniques to ensure its effectiveness. First, to get sufficient query-document pairs for training, we leverage a query generation network to obtain possible pairs of queries and documents. Second, we utilize the hierarchical k-means algorithm to generate a semantic identifier for each document. Third, we design a prefix-aware weight-adaptive decoder to replace the vanilla one in a sequence-to-sequence architecture. Specifically, the same token will be assigned different embedding vectors at different positions in the identifiers, while another transformer-based adaptive module is applied to the classification weights for token prediction in the context of a certain prefix. This makes the classifiers customized to different prefixes when decoding along the hierarchical tree structure. Besides, a consistency-based regularization loss is taken for training both encoder and decoder networks to mitigate the over-fitting problem.

Our NCI design solves the limitations of traditional index-retrieve pipelines from multiple perspectives. On one hand, a whole neural network model replaces the traditional inverted index or vector search solutions. It can be optimized end-to-end using realistic query-document pairs, which fully capture both term-based and semantic-based features and is adaptive to the changing of workloads. On the other hand, the model is able to capture deep interactions between queries and documents via encoder-decoder attention, which enlarges the capacity of vector-based representations. Moreover, NCI achieves much better ranking results than ANN-based approaches as it is optimized directly by the final target. Thus, it can be served as an end-to-end retrieval solution and release the burden of re-ranking for a long candidate list.

In addition to the superior performance, the invention of Neural Corpus Indexer is also promising from the perspective of system design. As nowadays, ranking and query-answering modules are already implemented by neural networks, NCI finishes the last piece of puzzle for the next-generation information retrieval system based on a unified differentiable model architecture. This reduces the dependency among different sub-modules, while the process of system deployment and maintenance could be greatly eased.

Our **contributions** are highlighted as follows.

- For the first time, we demonstrate that an end-to-end differentiable document retrieval model can significantly outperform both inverted index and dense retrieval solutions. This finding will inspire research on further steps towards the next-generation search systems, for instance, unifying informational retrieval, ranking and question answering in a single differentiable framework.
- We design a sequence-to-sequence model, named Neural Corpus Indexer (NCI), which generates relevant document identifiers directly for a specific query. In our experiments, the proposed NCI model improves the state-of-the-art performance of existing methods by a significant margin, achieving +51.9% and +19.2% relative enhancement for Recall@1 and Recall@10 respectively on NQ320k dataset. Also, NCI itself can achieve a competitive MRR score without using an explicit ranking model.
- We propose a novel decoder architecture, namely *prefix-aware weight-adaptive (PAWA)* decoder, to generate document identifiers. As verified by ablation studies, this invention is very crucial for NCI to achieve an outstanding performance. Moreover, query generation, semantic document identifiers and consistency-based regularization are all accountable for the superior capability of Neural Corpus Indexer.

2 Related work

In this section, we briefly introduce the related works and leave more discussions about the traditional web search techniques in the Appendix A.

Sparse retrieval. Traditional document retrieval methods are based on *Sparse Retrieval*, which is built upon inverted index with term matching metrics such as TF-IDF [37], query likelihood [26] or BM25 [36]. In industry-scale web search, BM25 is a difficult-to-beat baseline owing to its outstanding trade-off between accuracy and efficiency. In recent years, there are some attempts to incorporate the power of neural networks into inverted index. The Standalone Neural Ranking Model (SNRM) [47] learns high-dimensional sparse representations for query and documents, which enables the construction of inverted index for efficient document retrieval. Doc2Query [33] predicts relevant queries to augment the content of each document before building the BM25 index, and DocT5Query [32] improves the performance of query generation by the pre-trained language model T5 [5]. Furthermore, DeepCT [8] calculates context-aware term importance through neural networks to improve the term matching metrics of BM25.

Dense retrieval. Another line of research lies in *Dense Retrieval*, which presents query and documents in dense vectors and models their similarities with inner product or cosine similarity. These methods benefit from recent progresses of pre-trained language models, such as BERT [13] and RoBERTa [28] to obtain dense representations for queries and documents. At inference time, efficient Approximate Nearest Neighbor (ANN) search algorithms, such as k-dimensional trees [3], locality-sensitive hashing [9], and graph-based indexes (e.g., HNSW [31], DiskANN [23] and SPANN [7]) can be utilized to retrieve relevant documents within a sublinear time. Besides, Luan et al. [30] analyze the limited capacity of dual encoders, and propose a combination of sparse and dense retrieval methods with multi-vector encoding to achieve better search quality.

Autoregressive retrieval. The other way to approach retrieval is utilizing an end-to-end autoregressive models. Firstly, several efforts have been done on entity linking [12, 11, 10], which can be regard as a special type of retrieval task, *e.g.*, using an entity to ask the posed question. Recently, different from the entity linking task, Tay et al. [41] proposed the DSI (differentiable search index) to generate relevant document identifiers directly according to the query. Bevilacqua et al. [4] employ the autoregressive model to generate the relevant words for a query and utilize the generated string to retrieve relevant documents. Besides, the Deep Retrieval (DR) [16] approach for recommendation is also related to this category, which learns a deep retrievable network with user-item clicks and gets rid of the ANN algorithms based on the Euclidean space assumption.

Pre-trained language models. Recently, pre-trained Language Models (LMs), such as BERT [13] and RoBERTa [28], have led to a revolution in web search techniques. The representation vectors for all documents can be calculated and indexed offline. In the online serving stage, it calculates the representation vector for the input query and applies a crossing layer to calculate the relevance score between each query and document. The crossing layer usually adopts simple operators such as cosine similarity or a single feed-forward layer to retain a high efficiency. Gao et al. [14] find that a standard LMs' internal attention structure is not ready-to-use for dense encoders and propose the Condenser to improve the performance of dense retrieval. Moreover, ANCE [45] leverages hard negatives to improve the effectiveness of contrastive learning, which generates better text representations for the retrieval tasks.

3 Neural corpus indexer

The neural corpus indexer (NCI) is a sequence-to-sequence neural network model. The model takes *query* as input and outputs the most relevant *document identifier (docid)*, which can be trained by a large collection of <*query, docid*> pairs. The documents are encoded into semantic *docids* by the hierarchical k-means algorithm [19], which makes similar documents have "close" identifiers in the hierarchical tree. As shown in Figure 1, NCI is composed of three components, including *Query Generation*, *Encoder* and *Prefix-Aware Weight-Adaptive (PAWA) Decoder*. Query generation is implemented by a sequence-to-sequence transformer model [43] that takes as an input the document terms and produces a query as output [33]. The encoder, following the standard transformer architecture, is composed of N_1 stacked transformer blocks, which outputs the representation for an input query. For the decoder network, we stack N_2 transformer layers. To better align with the hierarchical nature of the semantic identifiers, we propose a weight adaptation mechanism based on another transformer

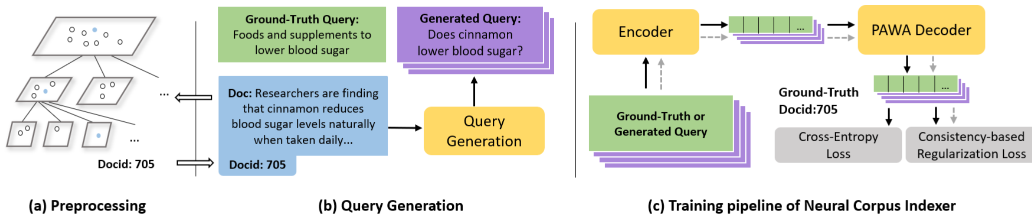


Figure 1: Overview of Neural Corpus Indexer (NCI). (a) Preprocessing. Each document is represented by a semantic identifier via hierarchical k-means. (b) Query Generation. Queries are generated for each document based on the content. (c) The training pipeline of NCI. The model is trained over augmented <*query, docid*> pairs through a standard transformer encoder and the proposed Prefix-Aware Weight-Adaptive (PAWA) Decoder.

to make the decoder aware of semantic prefixes. At inference time, the top N relevant documents can be easily obtained via beam search. Due to the hierarchical property of semantic identifiers, it is relatively easy to constrain the beam search decoding process on the prefix tree so that only valid identifiers will be generated.

3.1 Representing document with semantic identifiers

NCI generates document identifiers solely based on the input query without explicit document content, which is difficult when the size of the corpus is very large. Thus, we aim to inject useful priors into the identifiers, so that the semantic information of documents can be considered in the tree-based decoding process. In other words, we hope the documents with similar information have close *docids* to facilitate the learning process of NCI. To achieve this, we leverage the hierarchical k-means algorithm to encode documents. As shown in Figure 1(a), given a collection of documents be indexed, all documents are first classified into k clusters by using their representations encoded by BERT [13]. 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, we 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 $r_i \in [0, k)$ represents the internal cluster index for level i, and $r_m \in [0, c)$ 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. For simplicity, we set $k = 10$ and $c = 10$ in all experiments, leaving the optimization of these hyper-parameters to future work. The detailed procedure of hierarchical k-means will be described in Algorithm 1 in the Appendix B.2.

3.2 Query generation

One challenge of generating document identifiers by single query input is how to make the identifiers aware of the document semantics. Since the content of each document is not explicitly known at inference, it must be incorporated into the model parameters during training. To facilitate the training process, we generate a bunch of queries with a query generation module and bind the information of document content through training the sequence-to-sequence model with generated queries and their document identifiers. We adopt a standard sequence-to-sequence transformer [43] based on the implementation of Doc2Query [1], which takes as an input the document terms and produces relevant queries via random sampling. Note that we use random sampling instead of beam search to ensure the diversity of generated queries.

3.3 Prefix-aware weight-adaptive decoder

The probability of generating a document identifier can be written as follows:

$$p(l|x, \theta) = \prod_{i=1}^{m} p(r_i|x, r_1, r_2, ..., r_{i-1}, \theta_i), \tag{1}$$

where r_i is the i-th token in the current identifier; x is the representation output from encoder; θ denotes the total parameters and θ_i is the parameter for the i-th step.

This probability can be modeled by a transformer-based decoder. For an internal node with level i, the probability is calculated by:

$$h_i = \text{TransformerDecoder}(x, h_1, h_2, ..., h_{i-1}; \theta_i), \quad (2)$$

$$p(r_i|x, r_1, r_2, ..., r_{i-1}, \theta_i) = \text{Softmax}(h_i W). \quad (3)$$

Here h_i is the hidden representation for step i, which is calculated by a multi-head attention over encoder representation x and token representations of previous decoding steps. The linear classification weight is denoted by $W \in \mathbb{R}^{d \times v}$, d is the hidden dimension size and v is the vocabulary size of identifiers.

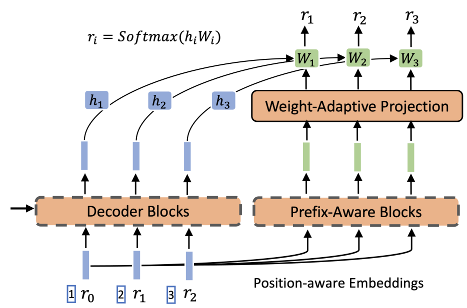


Figure 2: Overview of the Prefix-Aware Weight-Adaptive (PAWA) Decoder.

As the encoder and decoder utilize distinct vocabulary spaces, we do not share the embedding space for their tokens. Different from a standard decoding task, the meanings of the same token appearing at different places of the same identifier are different, as they correspond to different clusters in the hierarchical tree structure. For instance, the "$5_{\underline{2}}$" and "$5_{\underline{3}}$" of the same identifier "$3_{\underline{1}}5_{\underline{2}}5_{\underline{3}}$" correspond to different semantic meanings. Moreover, the same token in the same position may have different semantics with different prefixes. For example, in identifiers "$1_{\underline{1}}1_{\underline{2}}5_{\underline{3}}$" and "$2_{\underline{1}}4_{\underline{2}}5_{\underline{3}}$", the same token "$5_{\underline{3}}$" has different semantics in two different identifiers, as they are routed from different prefix paths. The two properties of the hierarchical semantic identifiers motivate us to design the novel **P**refix-**A**ware **W**eight-**A**daptor (PAWA) decoder.

Unlike a standard transformer decoder, the probabilities at different tree levels, such as $p(r_i|x, r_{1..i-1}, \theta_i)$ and $p(r_j|x, r_{1..j-1}, \theta_j)$ when $i \neq j$, do not share parameters with each other. To distinguish different semantic levels, we concatenate the position and token values as input for each decoding step, as shown in the left corner of Figure 2. Specifically, we have "$(1, 3)(2, 5)(3, 5)$" for the semantic identifier "$3_{\underline{1}}5_{\underline{2}}5_{\underline{3}}$", while "$(2, 5)$" and "$(3, 5)$" represent different tokens in the vocabulary space. As the token embedding and linear classification layers share the same weights, the same token value in different positions would correspond to different model parameters. Moreover, to reflect the influence of different prefixes, we expect the linear classification layer to be aware of different prefixes for predicting a specific token. Concretely, instead of using the same projection weight W in the linear classification layer, we employ the prefix-aware adaptive weights for each token classifier, which can be calculated by another transformer decoder,

$$W_{ada}^{i} = \text{AdaptiveDecoder}(e; r_1, r_2, ..., r_{i-1}) W_i \quad (4)$$

where e is the query embedding vector taken as initial input to the transformer decoder; $\{r_t | t \in (1, 2, ..., i-1)\}$ are prefix tokens before the i-th position, AdaptiveDecoder stacks N_3 transformer decoding layers with dimension d, and $W_{ada}^{i} \in \mathbb{R}^{d \times v}$ is the adapted weight matrix for the corresponding classifier. Finally, the i-th token in the given prefix can be predicted by $\text{Softmax}(h_i W_{ada}^{i})$.

For instance, to predict the third tokens in the identifiers "(1,3)(2,1)(3,5)" and "(1,2)(2,4)(3,5)" respectively, the corresponding adaptive weights are derived separately for different prefixes "(1,3)(2,1)" and "(1,2)(2,4)". As we already know the previous tokens for each position in the teacher forcing setting, the prefix-aware adaptive weights can be calculated and trained in parallel in different positions while adding little burden to the entire model.

3.4 Training and inference

Consistency-based regularization. To alleviate over-fitting, we employ a consistency-based regularization loss for training each decoding step. Given an input query q, we denote the model probabilities predicted by two forward passes with independent dropouts as $p_1(r_i|E(q), r_{1,...,i-1}, \theta_i)$ and $p_2(r_i|E(q), r_{1,...,i-1}, \theta_i)$ respectively, where $E(\cdot)$ denotes the encoder network. The consistency-based regularization loss tries to regularize the model prediction by minimizing the bidirectional Kullback-Leibler (KL) Divergence between two output probabilities with random dropout. The regularization loss of query q for the i-th decoding step is defined as,

$$\begin{aligned}\mathcal{L}_{reg} = \frac{1}{2}\Big(\sum_{i=1}^{m} &\mathcal{D}_{KL}\big(p_1(r_i|E(q), r_{1,...,i-1}, \theta_i) \| p_2(r_i|E(q), r_{1,...,i-1}, \theta_i)\big) \\ &+ \mathcal{D}_{KL}\big(p_2(r_i|E(q), r_{1,...,i-1}, \theta_i) \| p_1(r_i|E(q), r_{1,...,i-1}, \theta_i)\big)\Big).\end{aligned} \quad (5)$$

Training loss. Given a set of training examples $\mathcal{D} = \{(q, d)\}$ composed of queries (training queries and augmented queries) and document identifiers, the loss function can be written as follows:

$$\mathcal{L}(\theta) = \sum_{(q,d)\in\mathcal{D}} \left(\log p(d|E(q), \theta) + \alpha\mathcal{L}_{reg}\right), \tag{6}$$

where $p(d|E(q), \theta)$ denotes the probability of generating d with q as the input. The first part is the seq2seq cross-entropy loss with teacher forcing and the second part is the consistency-based regularization loss summed by all decoding steps. The whole process formulates a sequence-to-sequence neural network, which can be optimized end-to-end via gradient descent. The hyper-parameter α denotes a scaling factor of regularization loss, which will be analyzed in Section 4.4.

Inference via beam search. In the inference stage, we calculate the query embedding through the encoder network and then perform the beam search on the decoder network. Due to the hierarchical nature of *docid*, it is convincing to constrain the beam search decoding process with a prefix tree, which in turn only generates the valid identifiers. The time complexity of beam search is $O(LBF)$, where L is the max length of identifiers (the depth of tree), B is the beam size and F is the max fanout of the tree (10 in our experiments). Given a balanced tree structure built by a corpus with M documents, the average time complexity for beam search is $O(B\log M)$. We leave detailed descriptions of the constrained beam search algorithm in Appendix B.3.

4 Experiments

In this section, we empirically verify the performance of NCI and the effectiveness of each component on the document retrieval task, which generates a ranking list of documents in response to a query. In the following, we discuss the datasets and evaluation protocol in Section 4.1, describe the implementation details and baseline methods in Section 4.2, and present empirical results and analyses in Section 4.3 and 4.4 respectively.

4.1 Datasets & evaluation metrics

Datasets. Following DSI [41] and SEAL [4], we conduct our experiments on the Natural Questions [25] dataset. *Natural Questions* (NQ) dataset was introduced by Google in 2019 [25]. The version we use is often referred to as NQ320k, which consists of 320k query-document pairs, where the documents are gathered from Wikipedia pages and the queries are natural language questions. We use its predetermined training and validation split for evaluation.

Metrics. We use widely accepted metrics for information retrieval, including Recall@N and Mean Reciprocal Rank (MRR). Recall@N measures how often the desired document is hit by the top-N retrieved candidates. MRR calculates the reciprocal of the rank at which the first relevant document is retrieved. A high recall means that the ground truth document is contained in the retrieved candidate list, while a high MRR indicates that the corresponding document has already been ranked at the top position without a need for re-ranking.

4.2 Implementation details

Hierarchical semantic identifier. For semantic identifiers, we apply hierarchical k-means algorithm over the document embeddings obtained through a 12 layers BERT model with pre-trained parameters provided by the HuggingFace [44]. For each hierarchical layer, we employ the default k-means algorithm implemented in scikit-learn [34] with $k = 10$. For simplicity, the recursion terminal condition is also set as $c = 10$.

Query generation. We leverage the pre-trained model docT5query [32] for query generation. We provide all documents in the NQ320k dataset to predict augmented query-document pairs. For each document, we generate 10 queries with the first 512 input tokens of the document as the input and constrain the maximum length of the generated query as 64.

Training and inference. Neural Corpus Indexer (our approach) are implemented with python 3.6.10, PyTorch 1.8.1 and HuggingFace transformers 3.4.0. We utilize the parameters of the T5 pre-trained model [5] to initialize the encoder and randomly initialize the PAWA decoder. All NCI experiments are based on a learning rate 2×10^{-4} for encoder and 1×10^{-4} for decoder with a batch size 16 per GPU. We set the scaling factor of the consistency-based regularization loss as $\alpha = 0.015$, and the dropout ratio is 0.1. For inference, we apply the partial beam search algorithm to the trained seq2seq

Table 1: Performance comparison on NQ320k retrieval task

Method	Recall@1	Recall@10	Recall@100	MRR@100
Neural Corpus Indexer (Ours)	**88.72**	**95.84**	**97.43**	**91.59**
DSI (T5-Base)	27.40	56.60	–	–
DSI (T5-XXL)	40.40	70.30	–	–
SEAL (BART-Base)	26.55	53.61	72.67	35.64
ANCE (FirstP)	51.33	80.33	**91.78**	61.71
ANCE (MaxP)	52.63	**80.38**	91.31	62.84
BERT + BruteForce	28.65	53.42	73.16	36.60
BERT + ANN (Faiss)	27.92	53.63	73.01	37.08
BM25 + DocT5Query	**58.39**	75.76	89.51	**64.53**
BM25	30.23	47.02	68.54	36.26

model. We set the length penalty and the beam size as 0.3 and 100 respectively. All experiments are based on a cluster of NVIDIA V100 GPUs with 32GB memory. Each job takes 8 GPUs, resulting in a total batch size of 128 (16×8).

Baselines. We evaluate BM25 on both raw documents and those augmented by DocT5Query. The performance of DSI [40] is referred from its original paper as the implementation has not been open-sourced. To avoid the difference in data processing, we reproduce SEAL [4] and ANCE [45] by their official implementations. We leave the detailed settings in Appendix B.4.

4.3 Results

In Table 1, we compare the empirical results of NCI and corresponding baselines. On the NQ320k dataset, the proposed NCI model outperforms all baselines by a large margin across four different metrics. Compared with the state-of-the-art model, NCI improves 51.9% on Recall@1, 19.2% on Recall@10, 6.2% on Recall@100, and 41.9% on MRR@100 relatively. It is worth noting that we are the first to verify the superiority of deep text retrieval over traditional sparse and dense retrieval methods. Previous deep text retrieval methods (*i.e.*, DSI and SEAL) obtain relatively poor results even with a very large model size (e.g., T5-XXL). Consistent with previous studies, BM25 is an efficient and effective baseline. It even outperforms BERT-based dense retrieval solutions, perhaps owning to its capability to retrieve precise documents based on exact match. Further, we notice that query generation plays a key role in boosting the retrieval performance. With query generation, the BM25 + DocT5Query method achieves much higher performance than its vanilla version. ANCE also achieves competitive performance after fine-tuned by the training pairs, but the performance is far lower than our proposed NCI model. Moreover, the Recall@1 and MRR@100 metrics of NCI are outstanding, indicating that more than 90% of the queries can be fulfilled without re-ranking on the retrieved document list. This shows the potential of NCI to be served as an end-to-end solution that replaces the entire index-retrieve-rank pipeline in traditional web search engines.

Furthermore, to study the effect of each component, we report ablation results on NQ320k dataset in Table 2. In general, all five components are able to improve the performance of document retrieval, which are detailed below.

w/o query generation. This configuration removes the query generation module for data augmentation. Remarkably, the query generation boosts the performance greatly. The result is aligned with our expectation because training with the generated queries allows the model to be agnostic to the semantic meaning of each documents. Besides, although training on <*doc-content, docid*> pairs like DSI [40] also make the model aware of the semantic meaning of each documents, we argue that

Table 2: Ablation Study on NQ320k retrieval task

Method	Recall@1	Recall@10	Recall@100	MRR@100
Neural Corpus Indexer (Ours)	**88.72**	**95.84**	**97.43**	**91.59**
w/o query generation	53.63	67.84	78.43	59.16
w/o PAWA decoder	87.01	95.27	97.18	90.79
w/o semantic id	87.22	95.34	97.25	90.85
w/o regularization	87.34	95.42	97.27	90.89
w/o constrained beam search	87.41	95.71	97.32	90.84

training with generated queries is able to avoid the distribution shift problem, which also benefit the generalization performance.

w/o PAWA decoder. This configuration removes the adaptive decoder layer in Equation (4) and leverages shared weights with token embedding for the linear classification layer. We notice that the prefix-aware weight-adaptive decoder has a noticeable influence on the performance, which indicates that, instead of borrowing the vanilla transformer decoder, it is necessary to design a tailored decoder architecture for the task of semantic identifier generation.

w/o semantic id. This configuration replaces the semantic identifier of each document to a random generated one. We find a relative drop in the model performance on all four metrics, demonstrating that the semantic identifiers derived by hierarchical k-means have injected useful priors. We conjecture that the performance enhancement would be more significant on a larger document corpus.

w/o regularization. There is a performance drop on all four metrics without using consistency-based regularization loss. The reason is that the decoder network is prone to over-fitting. By making the prediction results for two augmented versions of the decoder to be consistent, the decoder model becomes more generalizable and resistant to over-fitting.

w/o constrained beam search. This configuration disables the validating constraint in beam search. In other words, the decoder network does not have a tree-based prior structure. Instead, all tokens in the vocabulary can be generated in each decoding step. We observe a performance drop on four evaluation metrics. This indicates that it is difficult to remember all information of valid identifiers in the network, and an explicit prior could be helpful for improving the quality of beam search.

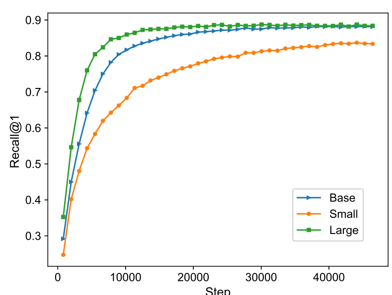


Figure 3: Learning curves of NCI with different model capacities

Setting	Recall@1	Recall@10	Recall@100	MRR@100
#layer = 0	87.01	95.27	97.18	90.79
#layer = 1	88.54	95.62	97.16	91.44
#layer = 2	88.56	95.67	97.28	91.48
#layer = 4	88.65	95.72	**97.54**	91.51
#layer = 6	**88.72**	**95.84**	97.43	**91.59**
#layer = 8	85.31	94.17	96.34	89.25

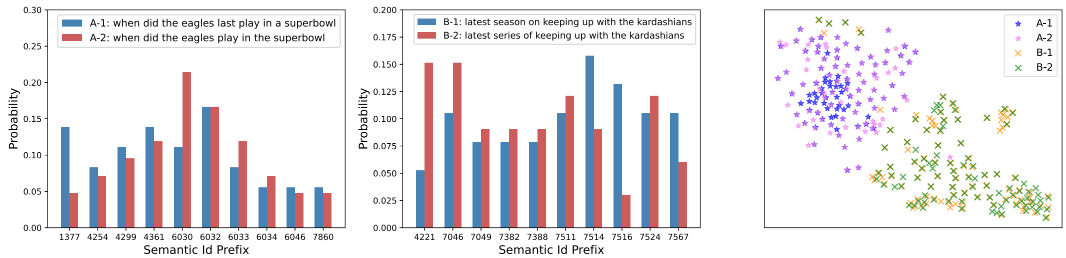
Table 3: NCI with different number of layers in PAWA adapter and different regularization hyper-parameter α in loss function

4.4 Analysis

Model capacity. Figure 3 compares the learning curves of NCI with different model capacities, which are identical to the small, base, and large settings of ordinary T5 [35]. We observe that with the increase of model size, NCI convergences more quickly with fewer epochs. At convergence, the small model achieves a relatively lower recall@1. Instead, both the base and large models achieve similar results after sufficient training epochs. This implies that the model capacity has a critical impact on the retrieval performance, and the capacity of base model seems to be enough to memorize all documents in NQ320k dataset. For a larger corpus, one may need to increase the model size to obtain satisfactory performance.

Layer number of PAWA adapter. We study the influence of the number of transformer layers in the PAWA adapter and choose the layer number from {0,1,2,4,6,8}. The results are summarized in Table 3. We notice that with the increasing of layer number, *i.e.* from 0 to 6, the overall performance is consistently improved under four metrics, except the Recall@100. But when the number of layer achieves 8, the performance is dropped significantly. We attribute that to the overfitting caused by a large PAWA adapter. Therefore, we adopt the design with 6 layer adapter in NCI.

Retrieved documents and their semantics identifiers. To verify the effectiveness of retrieval as well as the semantic identifiers learned by hierarchical k-means, we analyze the retrieval results of NCI for some exemplar queries. To illustrate, we select four queries denoted by *A-1*, *A-2*, *B-1* and *B-2*, where two queries inside the same group are semantically similar, and the queries in different groups correspond to distinct topics. In Figure 4(a) and 4(b), we show the probabilities of retrieved documents for each query in group *A* and *B* respectively. The digits along x-axis denote the four-bit prefixes for semantic identifiers of retrieved documents, and the y-axis stands for their probabilities.



(a) Query Group A (b) Query Group B (c) Document Embeddings

Figure 4: Analyses of retrieved documents with semantic identifiers. (a) The probabilities of retrieved documents for Query Group A; (b) Query Group B. (c) The t-SNE visualization of BERT-based document embeddings.

We notice that similar queries result in close document distributions, while dissimilar queries in different groups result in un-overlapped document collections. In addition, the documents retrieved by the same group of queries have close prefixes for the identifiers, *e.g.*, 6030, 6032, 6033, 6034 in group A and 7511, 7514, 7516 in group B. Also, we visualize BERT-based document embeddings by t-SNE [42] in Figure 4(c), in which each color represents the corresponding documents for a specific query. As shown in the figure, these documents naturally form two clusters with respect to different query groups. Thus, we conclude that the semantic document identifiers generated by the hierarchical k-means algorithm have positive effects on the retrieval performance.

Efficiency Analysis. We use an NVIDIA V100-32G GPU to analyze the efficiency of NCI. As the inference speed is influenced by both model capacity and beam size, we report the latency and throughput measures for multiple settings in Table 4. As NCI is an end-to-end retrieval method and achieves competitive performance without re-ranking, the latency and throughput are already affordable for some near-real-time applications. The latency of NCI is on par with DSI and SEAL using the same model size and beam size as all of them conduct beam search based on transformer decoders. BM25 is very efficient (<100ms per query on CPU using an open-source implementation [2], but the recall metrics are much lower. Furthermore, we can leverage other techniques to improve the efficiency of NCI, which will be discussed in the later section.

Table 4: Efficiency analysis

Model size	Beam size	Latency (ms)	Throughput (queries / s)
Small	10	78.46	58.48
Base	10	115.17	52.55
Large	10	188.60	43.39
Small	100	216.01	6.12
Base	100	269.31	5.62
Large	100	356.07	4.75

5 Limitation & Future Works

Despite the significant breakthrough, the current implementation of NCI still suffers from several limitations before deployment in an industrial web search system. Firstly, it requires a much larger model capacity for extending NCI to the web scale. Secondly, its inference speed needs to be improved to serve online queries in real time. Thirdly, it is difficult to update the model-based index when new documents are added to the system. In future works, we may tackle these problems from four aspects. (1) The architecture of sparsely-gated Mixture of Expert (MoE) [38] can be employed to enhance the model capacity. (2) Documents can be grouped into semantic clusters, and then NCI is used to retrieve relevant cluster identifiers. In this way, all documents in relevant clusters can be retrieved efficiently. (3) Model compression techniques, like weight quantization [22] and knowledge distillation [20], can be further taken to speed up inference. (4) We plan to explore a hybrid solution by building another index that serves new documents through traditional indexing algorithms.

6 Conclusion

In this work, we introduce a novel learning paradigm that unifies the learning and indexing stages by an end-to-end deep neural network framework. The proposed Neural Corpus Indexer (NCI) retrieves the identifiers of relevant documents directly for an input query, which can be optimized end-to-end with augmented query-document pairs. To optimize the recall and ranking performance, we invent the tailored prefix-aware weight-adaptive decoder. Empirically, we evaluate NCI on NQ320k dataset and demonstrate its outstanding recall and MRR performance over state-of-the-art solutions.

References

[1] https://github.com/castorini/docTTTTTquery.

[2] https://github.com/castorini/anserini.

[3] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. Autoregressive search engines: Generating substrings as document identifiers. *arXiv preprint arXiv:2204.10628*, 2022.

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] Wei-Cheng Chang, X Yu Felix, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*, 2019.

[7] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. Spann: Highly-efficient billion-scale approximate nearest neighbor search. *arXiv preprint arXiv:2111.08566*, 2021.

[8] Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*, 2019.

[9] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[10] Nicola De Cao, Wilker Aziz, and Ivan Titov. Highly parallel autoregressive entity linking with discriminative correction. *arXiv preprint arXiv:2109.03792*, 2021.

[11] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.

[12] Nicola De Cao, Ledell Wu, Kashyap Popat, Mikel Artetxe, Naman Goyal, Mikhail Plekhanov, Luke Zettlemoyer, Nicola Cancedda, Sebastian Riedel, and Fabio Petroni. Multilingual autoregressive entity linking. *arXiv preprint arXiv:2103.12528*, 2021.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[14] Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, 2021.

[15] Luyu Gao, Zhuyun Dai, and Jamie Callan. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042, 2021.

[16] Weihao Gao, Xiangjun Fan, Chong Wang, Jiankai Sun, Kai Jia, Wenzhi Xiao, Ruofan Ding, Xingyan Bin, Hui Yang, and Xiaobing Liu. Deep retrieval: Learning a retrievable structure for large-scale recommendations. *arXiv preprint arXiv:2007.07203*, 2020.

[17] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 55–64, 2016.

[18] Tonglei Guo, Jiafeng Guo, Yixing Fan, Yanyan Lan, Jun Xu, and Xueqi Cheng. A comparison between term-based and embedding-based methods for initial retrieval. In *China Conference on Information Retrieval*, pages 28–40. Springer, 2018.

[19] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[20] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[21] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[22] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[23] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[25] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[26] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, 2001.

[27] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. Parade: Passage representation aggregation for document reranking. *arXiv preprint arXiv:2008.09093*, 2020.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[29] Wenhao Lu, Jian Jiao, and Ruofei Zhang. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2645–2652, 2020.

[30] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021.

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

[32] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to doctttttquery. *Online preprint*, 2019.

[33] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

[34] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[36] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[37] Stephen E Robertson and Steve Walker. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 16–24, 1997.

[38] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations (ICLR)*, 2017.

[39] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*, pages 373–374, 2014.

[40] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*, 2020.

[41] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991*, 2022.

[42] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[44] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[45] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2020.

[46] Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019.

[47] Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 497–506, 2018.

[48] Shengyao Zhuang, Hang Li, and G. Zuccon. Deep query likelihood model for information retrieval. In *ECIR*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.
   (b) Did you mention the license of the assets? [Yes] See appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]