NEURAL INVERTED INDEX FOR FAST AND EFFECTIVE INFORMATION RETRIEVAL

Gianmarco Scarano
Università degli Studi di Roma "La Sapienza"
Rome, Italy
scarano.2047315@studenti.uniroma1.it

Emanuele Rucci
Università degli Studi di Roma "La Sapienza"
Rome, Italy
rucci.2053183@studenti.uniroma1.it

ABSTRACT

This project explores the challenges and strategies encountered while training a Differentiable Search Index (DSI) model for Information Retrieval (IR) using the MSMarco Dataset. Key areas of investigation include Document and Doc ID representation, training methodologies, and architectural designs. Despite computational constraints, experiments with models such as Flan-T5, BERT, and Encoder MoE-Decoder shed light on the effectiveness of the various approaches. Over-fitting issues, dataset limitations, and hardware constraints are discussed, along with proposed solutions for future research.

***K*eywords** Information Retrieval · Differentiable Search Index · Mixture-of-Experts

1 Introduction

Information Retrieval (**IR**) systems, nowadays, play a crucial role by accessing vast amounts of information available in various digital formats. These systems facilitate the retrieval of relevant information from large collections, such as documents, web pages, multimedia content, and databases, in response to certain user queries, along with bridging the gap between users and the information they seek, enabling users to find relevant content efficiently. The problem of information retrieval is typically addressed through a combination of indexing and querying techniques:

- Indexing: is the process of creating and maintaining structured representations of the content of documents to facilitate efficient search and retrieval.
- Retrieval: searching for and obtaining relevant information from a collection of documents or data based on a user's query.

2 Differentiable Search Index

In **DSI**, a unique Deep Neural Network is used to perform both the *Indexing* and *Retrieval* phases, meaning that the model has to give as output the Doc ID that contains the answer for a given query, thus, employing the well-known *Seq2Seq* paradigm.

When dealing with Deep Neural Networks for Information Retrieval (**IR**), we typically address two main problems:

- **Indexing**: The model learns to recall documents based on a set of tokens representing the document's content.
- **Retrieval**: Given a query represented by tokens, the model produces a ranked list of document identifiers.

Along with those, in order to have a proper DSI Model, one also needs to correctly address various multiple aspects, such as: **Doc representation**, where we want to cover which parts of the document are essential for the model to associate with its document identifier, **Doc ID representation**, as in choosing how to correctly represent Doc IDs (e.g., random strings, incremental numbers, etc.), **Neural Network Architectures**, including specific implementations, hyperparameters and optimization strategies and **Data**, since we want to provide the model with meaningful input data consisting of document texts and associated queries to train and evaluate the model's performance in retrieval tasks.

3 Our methods

Our setup includes various combinations of Document and Doc ID representation, two different training strategies (namely *Discriminative* and *Generative*) and two types of architecture design. We will dive into each combination explaining the rationale behind each choice.

3.1 Document Representation

Each document representation strategy has its own advantages and drawbacks. An ideal representation should maximize the likelihood of containing the relevant information for most or all associated queries, which we'll refer to as *"representation power"*. The following strategies have been implemented and tested:

- **Direct Indexing**: consider a certain amount of *N* words from a document and associate them with a Doc ID. We did set $N = 32$, taking effectively the first 32 words from the document itself. If the selected chunk contains exactly a response to a certain linked query, its *representation power* will be maximum. As for the cons, although this strategy is easy and fast to implement, it's not guaranteed that the N contiguous words selected have the maximum *representation power*. This, for example, happens when the first N tokens of a document represent meaningless information such as metadata headers, poorly formatted text and so on.
- **Set Indexing**: Take all the document's body, remove duplicates and stopwords and use it as document representation. It is clear that now the representation is more significant, but it is now up to model to actually properly interpret the text in order to give the answer to the query. As cons, we can consider that the model as to deal with a bigger input w.r.t *Direct Indexing*.
- **Summarization**: We use Falcon's Text-Summarization[1] model to generate summaries from documents. It offers various benefits like specifying summary length, providing prompts for summarization and even customizing output (e.g. we can focus the summarization on a given aspect). However, it may become impractical for large datasets due to the elevated time and computational cost required for summarization.

3.2 Document Identifier Representation

Due to computational constraints, we've opted for a single Doc ID representation which is the *Unstructured Atomic Identifiers*. We have the idea that a hierarchical Doc ID generation with semantic meaningful tokens may allow the model to work better. Also, as stated in [2](Page4), the Unstructured strategy outperforms Semantic and Naive approaches.

3.3 Training type

As in[3], we employed two different strategies for training. Both of them make use of an **LLM** as Neural Engine. The difference lies in how the output Doc IDs are generated. The *generative* training type involve the use of decoder's generation capabilities to produce Doc IDs as tokenizable strings. Conversely, the *discriminative* approach involves outputting a probability distribution over the entire set of possible Doc IDs. Due to limited computational resources, we opted to exclude experiments dedicated to analyzing to which extent the indexing method affects the overall behavior of the model. Specifically, we just deployed the ***Input2Target*** strategy for indexing, which maps a document representation (Paragraph 3.1) to a unique Doc ID for each document in the corpus. An image of the two training types can be found in 1

3.4 Architecture design

While the reference paper[3] uses the T5 model, we propose using two similar architecture setups, exploring 3 different Encoder-Decoder models: **Flan-T5**[4] as the baseline, a smaller version of **BERT**[5] with parameter sharing between its Encoder-Decoder, and a **Mixture-of-Expert (MoE) Switch Transformer**[6] as Encoder and **Flan-T5** as Decoder. In our analysis, we extensively use quantization strategies with the recent **LoRA**[7] and **QLoRA**[8] studies, along with the fine-tune approach of some baseline (Flan-T5) layers without quantization strategies. The upcoming section will highlight the motivations behind each architectural design selection.

3.4.1 Flan-T5 & BERT

The baseline using *FlanT5* aims to replicate the entire pipeline suggested in the reference paper. We opted for Flan-T5 because, as noted in their publication[4](Page11), it outperforms the classic T5 model. In the code, we refer to this model as `FoundationModel()` with '*model_id* = *google/flan-t5-base*'.

The *BERT* Encoder-Decoder experiments have been designed to investigate parameter sharing between the Encoder and Decoder[9], hypothesizing that it could lead to a more efficient representation of documents and queries. Unlike traditional setups, where the *Q, K, V* projection layers are separate, this configuration shares these layers between the Encoder and Decoder. In our code, this model is referenced as `FoundationModel()` with '*encoder_id/decoder_id* = *bert-base-uncased*'. Both models are utilized within the `DSI_Model()` class for training and inference.

3.4.2 Encoder MoE - Decoder

The underlying investigation of this study is pretty straightforward: it should be intuitive that more parameters typically result in a better indexing (and retrieval) capability, hence, certain documents might benefit from specialized models or layers. As a result, we explored the possibility of using a *Switch Transformer* Encoder, paired with a classic *Flan-T5* Decoder. As for the Switch Transformer, it switches the traditional FFN Layers of the Transformer with a pre-trained MoE guided by a **Router** which chooses the best FFN Layer according to the input, following a noisy top 1 expert selection. In our setup, we chose the default `base` configuration with 8 experts and a capacity factor of 64 tokens for balancing efficiency and cost.

In practice, we pass the input through the MoE Encoder, extract the embeddings, and then use the Flan-T5 Decoder for computing the loss and generating the Doc IDs. In the code, we refer to this model as `SwitchTransformer()` for the Encoder, `FlanT5()` for the Decoder, which will then be passed to the `DSI_EncoderDecoder()` class for training/inference.

3.4.3 Quantization and fine-tuning

Considering the large size of the models used, we employed a technique called quantization, which falls under the model compression theory, to facilitate training in Colab. We use `BitsAndBytes` to load up these large models into our small hardware system via model quantization up to 8-bit precision. Usually, once a model is quantized, it's not further trained for downstream tasks due to potential instability caused by lower precision weights and activations. However, LoRA comes in handy here, since it enables us to train only the *Q, K*, and *V*[8](Page10) Projection Layers of the quantized model. By combining quantization with LoRA (known as **QLoRA**), we can effectively train even the largest models on a single GPU. With this setup, Flan-T5 passes from 248M trainable parameters to 1.3M. For BERT, we pass from 139M to 1.3M. For Encoder MoE-Decoder, we pass from 868M to 2.6M. We conducted additional experiments using the Flan-T5 Baseline setup, where we froze all layers except the last four for both the Encoder and Decoder. This study aimed to assess the impact of model's performance and speed when fine-tuning. It's important to note that we did not use the QLoRA method in these experiments. The decision to freeze the last four layers was based on the idea that the initial layers capture coarse-grained aspects of the input sequence, regardless of the specific task. In contrast, the last layers are optimized to achieve a specific goal. During fine-tuning, QLoRA allowed Flan-T5 to pass from 248M trainable parameters, to 66M.

3.4.4 Multitask training setup

As suggested in the paper[3], we have employed a two-Multitask setup for training:

- **Task Prompting**: Involves adding a prompt as a prefix to both the document and the query within our model, informing the model itself about the task it needs to perform. The prompts used were: *"Generate a document identifier with 5 digits between 0 and 9 for the following document:"* (Indexing) and *"Generate a document identifier with 5 digits between 0 and 9 as response to the following query:"* (Retrieval). This approach ensures uniformity across tasks, allowing for consistent training procedures and decoding processes for both tasks.

- **Indexing/Retrieval ratio**: Retrieval depends on Indexing, therefore we allocate batches exclusively for Indexing and include Retrieval as well in the final batches of each epoch, creating a Multitask setup. We only index documents from the chunk for which we have corresponding queries. More in the next chapter.

4 The Dataset

The dataset that we have used in this study is the **MSMarco Dataset** containing $3.21M$ documents[10] divided in 59 chunks. For our experiments, we focused on the first chunk (Chunk 00), which contains around $200K$ documents with mean 'Body' length of 7773 words. We conducted experiments using both the entire chunk and subsections of it. We filtered the first $10K$ documents from Chunk 00, ensuring they had corresponding queries, allowing training on a total of 759 document-query pairs. To overcome this challenge, we devised a solution, which consists of creating a Training set that combines the full Chunk 00 (for *Indexing*) with a subset of *Retrieval* examples. The Indexing set is comprised of documents belonging to Train, Test and Validation sets, while the Retrieval set is divided between the Training, Validation and Test splits. This dataset setup is graphically explained in 2. Additionally, we varied the number of selected documents in other setups to observe the behavior of the baseline model with varying document counts.

5 Experiments

In this section, we will highlight the experiments that we have done, anticipating that the result are really poor in terms of performances, but we will try to give some explanation about that in the Result section. The different experiments are grouped by the models that power the overall DSI. Then, for each of them, we explored different variations of hyperparameters (Doc Strategy, Doc ID Strategy, etc.). Moreover, we fixed the *Optimizer* and *Learning Rate* to be the same for each experiment, respectively **Adam** with **LR** $= 1e-3$. Runs with **AdamW** along with a slightly smaller **LR** $= 3e-4$ have been proposed, which did not show overall better performances. A LR scheduler have also been tested, but more on this in Chapter 6.

5.1 Training experiments

Table 1: Flan-T5, BERT and Encoder MoE-Decoder tests

Model	Doc Strategy	Doc ID Strategy	Prompting	Training Type	N. Docs	Fine-Tuning
Flan-T5	Direct Indexing	Naively Structured	True	Generative	10k	False
Flan-T5	Direct Indexing	Naively Structured	True	Generative	50k	False
Flan-T5	Direct Indexing	Naively Structured	False	Generative	10k	False
Flan-T5	Direct Indexing	Naively Structured	True	Discriminative	10k	False
Flan-T5	Set Indexing	Naively Structured	True	Generative	10k	False
Flan-T5	Summarization	Naively Structured	True	Generative	10k	False
Flan-T5	Summarization	Naively Structured	True	Generative	10k	True
Flan-T5	Summarization	Naively Structured	True	Generative	200k	True
BERT	Direct Indexing	Naively Structured	True	Generative	10k	False
BERT	Summarization	Naively Structured	True	Generative	10k	False
EncMoE-Decoder	Direct Indexing	Naively Structured	True	Generative	10k	False
EncMoE-Decoder	Direct Indexing	Naively Structured	False	Generative	10k	False

6 Results

Considering that the MSMarco Dataset has a really absurd number of documents, our project is constrained to using only a very small subset of it, consisting of just 10K documents. As a result, computing the proposed metrics, such as MAP, Recall@1000 or even Hits@K and Accuracy, becomes impractical or even impossible, since results are all equal to 0 even with different decoding methods (beam, greedy search). However, we can still assess model performance using a simple metric like the Loss, which helped us in making few assumptions about our proposed experiments. The logs are listed in the WandB website. Another metric that we have analyzed is the percentage of strings decoded by the generative models which are considered as valid Doc IDs present in the corpus. We report that the range vary between 15% and 21% across all models; This is indeed suggesting that the approach is not really effective neither in the understanding of the Doc ID representation.

6.1 Flan-T5

In our experiments with Flan-T5, we found that using prompting during training improved stability and led to slightly lower loss. However, increasing the number of documents to 200K significantly extended training time and resulted

in higher loss compared to other models, as the model needed more time to index all the documents properly. In a Fine-Tune setup, we observed a notable increase in loss with an increase in the number of documents. For instance, with 10K documents, the loss was 3.7, whereas with 200K documents, it rose up to 9.8. Regarding summarization, we noticed that generative training exhibited more stable training, achieving the best loss of **3.003** for this model. In contrast, discriminative training reached a loss peak of 8.07, indicating potential over-fitting issues that we will be discussed further in Sub-Chapter 6.4.

6.2 BERT

In our experiments with BERT, we primarily focused on assessing the effectiveness of parameter sharing. However, we also conducted an experiment specifically targeting summarization in a generative setting. We ran two separate experiments, differing only in the Learning Rate settings. In one run, we used $lr = 1e - 3$, while in the other, we used $lr = 3e - 4$ with a LR scheduler known as *Linear Warmup*. This scheduler gradually increased the lr for a few epochs and then linearly decreased it to around 0. Surprisingly, this simple adjustment allowed us to relax the training and achieve the best loss for this project, approximately **2.943**, compared to the other run which achieved a loss of around 3.162 and compared to the Flan-T5 baseline which got 3.003 at best.

6.3 Encoder MoE - Decoder (SwitchTransformer & Flan-T5)

We conducted two primary experiments using this Neural Engine. Specifically, we tested Direct Indexing and Unstructured Atomic strategies for Documents and Doc IDs, but the main focus was on assessing the impact of prompting, aiming to determine if it affects the SwitchTransformer Encoder part as well. We observed that without prompting, the model achieved a minimum loss of approximately 4.298, whereas with prompting, it reached around **3.568**. Although the difference may seem slight, it amounts to 0.73 points. This result reinforces our belief that having an Encoder representation and projecting it into a different Decoder is still effective. We consider this outcome a notable achievement, as it surpassed our initial expectations regarding the model's performance in these tests.

6.4 Problems

Several challenges emerged during this project. We are almost sure that there must be a problem into the Training procedure, the way on which the Dataset is organized or even our ratio of Retrieval/Indexing. Additionally, we observed that was easy for this setup to fall into over-fitting issues. We believed this phenomena was related to the small number of documents that were being used for Indexing (and Retrieval), although it has been proven that even with a lot more document the over-fit still persist. The problem in general is that the model quickly learns to index documents but fails to retreive them in all the various proposed setups. Another idea, that we have regarding the failure of our approach, is related to the mismatch between the query and document content (e.g. A query response to a Document could be located in a different portion then the indexed one). This problem might be overcame with more clever Indexing strategies that allows a more powerful preprocessing. Another aspect: the length of the input data at Indexing (Document) is very different from the length of input data at Retrieval (Query), thus the DSI model suffers from a data distribution mismatch[11](Pag.3−Paragraph2). It's worth notice the discrepancy between the embeddings in the *Encoder MoE - Decoder* setup, since the Decoder's task must be performed on top of another model's Encoder. Additionally, the Quantization strategy significantly compromises accuracy and precision to facilitate the training process, prioritizing speed over performance. Concluding, papers like [11] and [10] make use of 8 Tesla A100 GPUs by 40GB each, while we had access to a simple Tesla T4/RTX 3060 GPUs with respectively 14.7GB and 12GB of VRAM.

7 Conclusion

As first experience with LLMs, this project has been really useful to understand how **IR** works under the hood, despite the poor performance obtained, which we believe are caused by an incomplete way of dealing with the data. We hardly believe that, for this project, further technical and practicals tricks on this data may lead to any better results. Another experiment we considered was utilizing a MoE Decoder such as Mixtral-8x7B in conjunction with the existing MoE Encoder implemented in this project, but for obvious reasons this has not been tested.

Acknowledgments

This work has been done as project for the Deep Learning course held by Professor Fabrizio Silvestri during the academic year 2023/2024 in "La Sapienza" University of Rome.

References

[1] HuggingFace. Fine-Tuned T5 Small for Text Summarization. 2023. `https://huggingface.co/Falconsai/text_summarization`.

[2] Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. DSI++: Updating Transformer Memory with New Documents, 2023.

[3] Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. Transformer Memory as a Differentiable Search Index. *arXiv:2202.06991*, 2022.

[4] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, and Xuezhi Wang et al. Scaling Instruction-Finetuned Language Models. *arXiv:2210.11416*, 2022.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, arXiv:1810.04805, 2018.

[6] William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *CoRR*, arXiv:2101.03961, 2021.

[7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, arXiv:2106.09685, 2021.

[8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs, 2023.

[9] HuggingFace. BERT Encoder-Decoder parameter sharing. 2023. `https://github.com/huggingface/transformers/blob/v4.37.2/src/transformers/models/encoder_decoder/modeling_encoder_decoder.py#L257-L267`.

[10] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. Semantic-enhanced differentiable search index inspired by learning strategies, 2023.

[11] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. Bridging the Gap Between Indexing and Retrieval for Differentiable Search Index with Query Generation, 2023.

8 Appendix

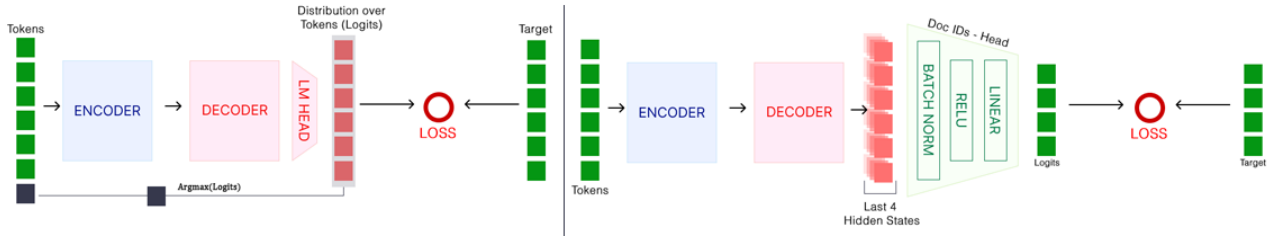

Figure 1: Comparison between generative (left) and discriminative (right) training approaches for the DSI models. In the generative approach, Doc IDs are generated autoregressively, whereas in the discriminative approach, a linear projection into the Doc ID space is employed via a Linear Layer.

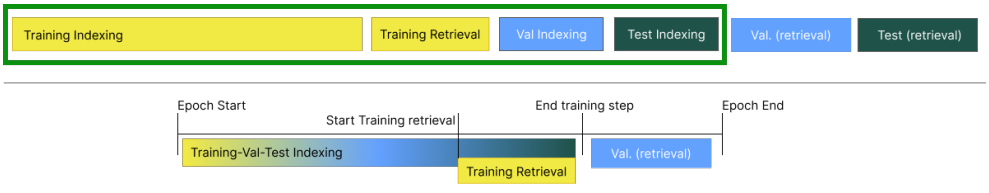

Figure 2: Custom Dataset generation scheme
The green box on the left represent the Train split which is composed of Training Documents on which is performed Indexing and Retrieval, plus Validation and Test Documents on which is performed Indexing only. On the bottom, the timeline of batches managed during an entire epoch is shown.