DIFFERENTIABLE SEARCH INDEXING [*]

Alessio Borgi, Eugenio Bugli, Damiano Imola
1952442, 1934824, 2109063
Sapienza Università di Roma
Rome
`{borgi.1952442, bugli.1934824, imola.2109063}@studenti.uniroma1.it`

ABSTRACT

***K*eywords** First keyword · Second keyword · More

1 Introduction

Trainable Information Retrieval (IR) systems are characterized by two phases:

- **Indexing:** Indexing of a corpus, which means to associate the content of each document with its corresponding docid.
- **Retrieval:** Learn how to retrieve efficiently from the index, which means to

Instead of using Contrastive Learning based Dual Encoders, the paper proposes an architecture which directly map a query **q** to a relevant docid **j**. This architecture, called DSI, it's implemented with a pre-trained Transformer and all the information of the corpus are encoded within the parameters of the language model. When we are doing inference, the give to the trained model a text query as input and we expect to obtain a docid as output. If we are interested in a ranked list of relevant documents we can also use Beam Search. Our DSI system uses standard model inference to map from encodings to docids, instead of learning internal representations that optimize a search procedure. DSI can be extended in different ways:

- **Document representation:** there are several ways to represent documents (e.g. full text, bag-of-words representations, ...)
- **Docid representation:** (e.g. unique tokens, structured semantic docids, text strings, ...)

1.1 Indexing Methods

Given a sequence of document tokens, the model is trained to predict the docids. There can be used different strategies:

- **Inputs2Target**:
- **Targets2Inputs**:
- **Bidirectional**:

2 Dataset

In our work, we have used the first version of the MS Marco Dataset (reference bib), which is composed by 100k real Bing questions and human generated answers. The dataset is already partitioned into Training (82326 samples), Validation (10047 samples) and Test (9650 samples). Each partition is organized as a dictionary where the most important keys are the following:

[*] *Citation*: **Authors. Title. Pages.... DOI:000000/11111.**

- **answers**: the answer related to the query based on the text informations.
- **passages**: contains another dictionary where we can find the complete corpus of the text and each passage that composes it.
- **query**: it is the question asked.

Since we are interested in the ranking of the documents, we have used the SimpleSearcher from pyserini [1], in order to obtain the most relevant 1000 documents. The pre-processing applied to the dataset is explained in the following subsections.

2.1 Tokenization

Starting from the dataset, we have computed the maximum length of the inputs of the encoder (1797) and decoder (4), which will be useful during the tokenization process. We have used the pretrained tokenizer from the small version of the T5 model (reference). Our tokenized dataset is composed only by the following parts:

- **Query**: tokenized version of the original query
- **Query and Corpus**: tokenized input text, which is composed by the concatenation of the query and the corpus
- **Document IDs**: tokenized version of the identificator of each document
- **Ranked Document IDs**: tokenized version of the ranked list of the first 1000 document ids

2.2 DSI Multi-Generation

In this subsection we have described the procedure to generate semantically structured document ids, which are characterized by the associations between queries and standard document ids. In other words, the docid should be able to capture some informations related to the semantics of the associated document. To do this, we have followed the algorithm provide by (reference ...)

Algorithm 1 Generating Semantically Structured Identifiers

```
Require: Document embeddings $X_{1:N}$, where $X_i \in \mathbb{R}^d$ generated by a small 8-layer BERT model with $c = 100$
Ensure: Corresponding docid strings $J_{1:N}$
function GENERATE_SEMANTIC_IDS($X_{1:N}$)
    $C_{1:10} \leftarrow$ CLUSTER($X_{1:N}$, $k = 10$) # k-means clustering
    $J \leftarrow$ empty list
    for $i \leftarrow 0$ to 9 do
        $J_{\text{current}} \leftarrow [i] \times |C_{i+1}|$
        if $|C_{i+1}| > c$ then # recursion if there are more than c documents
            $J_{\text{rest}} \leftarrow$ GENERATE_SEMANTIC_IDS($C_{i+1}$)
        else
            $J_{\text{rest}} \leftarrow [0, \ldots, |C_{i+1}| - 1]$ # assign arbitrary number from 0 to $c - 1$
        $J_{\text{cluster}} \leftarrow$ ELEMENTWISE_STR_CONCAT($J_{\text{current}}$, $J_{\text{rest}}$)
        $J \leftarrow J$.APPEND_ELEMENTS($J_{\text{cluster}}$) # Append all elements of $J_{\text{cluster}}$ to $J$
    $J \leftarrow$ REORDER_TO_ORIGINAL($J$, $X_{1:N}$, $C_{1:10}$)
    return $J$
```

2.3 Data Augmentation

3 Model

Our main architecure is T5.

4 Training and Results

gradient accumulation for each 4 batches mixed precision 16 bits

The documentation for `natbib` may be found at

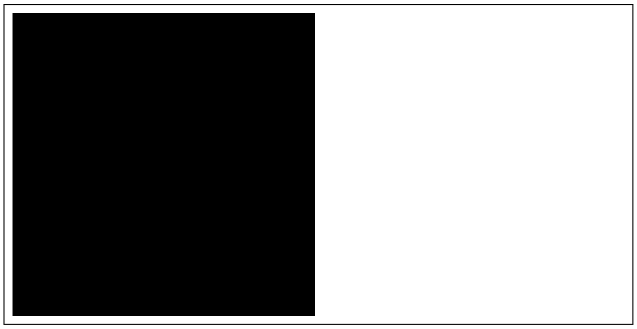

Figure 1: Sample figure caption.

Table 1: Sample table title

Part		
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

> Hasselmo, et al. (1995) investigated…

`https://www.ctan.org/pkg/booktabs`

4.1 Figures

See Figure 1. Here is how you add footnotes. [2]

4.2 Tables

See awesome Table 1.

5 Conclusion

Your conclusion here

Acknowledgments

This was was supported in part by......

References

[1] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings*

[2] Sample of the first footnote.

of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021), pages 2356–2362, 2021.