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 Ecnoders, 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 Generation

Given the MS Marco Dataset, we need to perform some preprocessing in order to create an association between queries and docids. This Dataset is composed by 100k passages text which are partitioned into Training, Validation and Test. Each partition is composed by lists of passages texts and related queries. We create a set $\mathcal{U} = \mathcal{O} \cup \mathcal{P}$, where $\mathcal{O}$ is the

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

set composed by each segment (passage text) of the MS Marco Dataset, while $\mathcal{P}$ is the set of pseudo-queries from each segment of $\mathcal{O}$, which are artificially generated by a model (docT5query).

2.1 Filtering with Rankers

This new set $\mathcal{U}$ must be filtered in order to maintain only the relevant segments. This is done by using a ranker that takes as input each segment, which is used like a query and gives as output the first $||$ ranked documents that have relevant information with respect to the segment. This procedure is fundamental since checks if a segment has enough information to represent the document and can be used as a suitable query. In our approach we have tried three different approaches:

- **Lucene**: sparse ranker
- **Faiss**: dense ranker
- **Hybrid**:

With this filtering procedure we are creating the real dataset $\mathcal{T}$, which will be the one used in the training phase.

3 Document Representation Strategies

4 Docid Representation for Retrieval

5 Headings: first level

See Section 5.

5.1 Headings: second level

5.1.1 Headings: third level

Paragraph

6 Examples of citations, figures, tables, references

[1].

The documentation for `natbib` may be found at

`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`

6.1 Figures

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

6.2 Tables

See awesome Table 1.

[2]Sample of the first footnote.

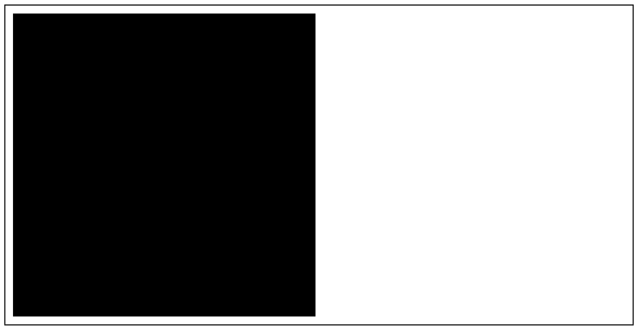

Figure 1: Sample figure caption.

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

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