纯python的BM25s框架，算法er文本搜索可以不用ES了，速度与效率的王者！

原创 nipi NLP前沿 2024年06月19日 11:35 湖北

许多流行的 BM25 库都是在 Java 中基于 Lucene 构建的。尽管速度很快，但由于需要 Java 运行时，在 Python 中使用会相对麻烦。比如使用Es，需要搭一个es服务，然后再python脚本里边连接，相对而言没有完全python实现的简单。不论正式环境，今天介绍的这个框架做一些测试会方便很多。

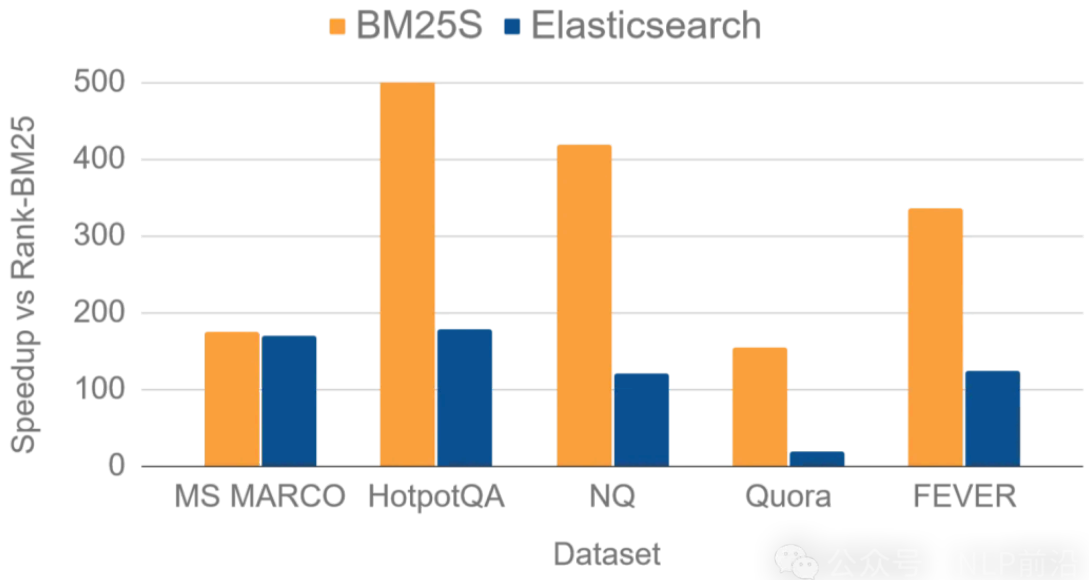
一些通过基于 Python 的实现（例如 BM25S 和 Rank-BM25），可以在大约 10 行内对文本进行标记、索引和检索。然而，简单地使用 Numpy 实现可能无法达到与基于 Java 实现相同的速度。所以有了这个新的工具包，性能超高，可用于在python中快速的完成bm25检索，BM25S 站在巨人的肩膀上：Rank-bm25（第一个 Python 实现）、Pyserini 和 bm25-pt（启发了这个项目）。

特色

fast：bm25s 用纯 Python 实现，并利用 Scipy 稀疏矩阵来存储所有文档标记的急切计算的分数。这允许在查询时进行极快的评分，从而将性能比流行的库提高几个数量级（请参见下面的基准测试）。

simple：bm25s 的设计易于使用和理解。您可以使用 pip 安装它并在几分钟内开始使用它。不依赖于 Java 或 Pytorch - 您只需要 Scipy 和 Numpy，以及用于词干提取的可选轻量级依赖项。

下面，我们将 bm25s 与 Elasticsearch 相对于 rank-bm25 （BM25 最流行的 Python 实现）的加速比进行比较。我们在 BEIR 的一些流行数据集上测量每秒查询次数 (QPS) 的吞吐量。



Throughput (Queries per second)

We compare the throughput of the BM25 implementations on various datasets. The throughput is measured in queries per second (QPS), on a single-threaded Intel Xeon CPU @ 2.70GHz (found on Kaggle). For BM25S, we take the average of 10 runs. Instances exceeding 60 queries/s are in **bold**.

Dataset	BM25S	Elastic	BM25-PT	Rank-BM25
arguana	**573.91**	13.67	**110.51**	2
climate-fever	13.09	4.02	OOM	0.03
cqadupstack	**170.91**	13.38	OOM	0.77
dbpedia-entity	13.44	10.68	OOM	0.11
fever	20.19	7.45	OOM	0.06
fiqa	**507.03**	16.96	20.52	4.46
hotpotqa	20.88	7.11	OOM	0.04
msmarco	12.2	11.88	OOM	0.07
nfcorpus	**1196.16**	45.84	256.67	**224.66**
nq	41.85	12.16	OOM	0.1
quora	**183.53**	21.8	6.49	1.18
scidocs	**767.05**	17.93	41.34	9.01
scifact	**952.92**	20.81	**184.3**	47.6
trec-covid	**85.64**	7.34	3.73	1.48
webis-touche2020	**60.59**	13.53	OOM	1.1

快速安装

```
https://github.com/xhluca/bm25s
pip install bm25s
```

简单应用示例

```
import bm25s
import Stemmer    # optional: for stemming

# Create your corpus here
corpus = [
        "a cat is a feline and likes to purr",
        "a dog is the human's best friend and loves to play",
        "a bird is a beautiful animal that can fly",
        "a fish is a creature that lives in water and swims",
]

# optional: create a stemmer
stemmer = Stemmer.Stemmer("english")

# Tokenize the corpus and only keep the ids (faster and saves memory)
```

```
corpus_tokens = bm25s.tokenize(corpus, stopwords="en", stemmer=stemmer)

# Create the BM25 model and index the corpus
retriever = bm25s.BM25()
retriever.index(corpus_tokens)

# Query the corpus
query = "does the fish purr like a cat?"
query_tokens = bm25s.tokenize(query, stemmer=stemmer)

# Get top-k results as a tuple of (doc ids, scores). Both are arrays c
results, scores = retriever.retrieve(query_tokens, corpus=corpus, k=2)

for i in range(results.shape[1]):
        doc, score = results[0, i], scores[0, i]
        print(f"Rank {i+1} (score: {score:.2f}): {doc}")

# You can save the arrays to a directory...
retriever.save("animal_index_bm25")

# You can save the corpus along with the model
retriever.save("animal_index_bm25", corpus=corpus)

# ...and load them when you need them
import bm25s
reloaded_retriever = bm25s.BM25.load("animal_index_bm25", load_corpus=True)
# set load_corpus=False if you don't need the corpus
```

支持内存友好的检索场景，可以使用 mmap 选项将BM25索引作为内存映射文件加载，这样您就可以加载索引而无需将完整索引加载到内存中。当有一个大索引并且想要节省内存时，这非常有用！

Memory Efficient Retrieval

`bm25s` is designed to be memory efficient. You can use the `mmap` option to load the BM25 index as a memory-mapped file, which allows you to load the index without loading the full index into memory. This is useful when you have a large index and want to save memory:

```
# Create a BM25 index
# ...

# let's say you have a large corpus
corpus = [
    "a very long document that is very long and has many words",
    "another long document that is long and has many words",
    # ...
]
# Save the BM25 index to a file
retriever.save("bm25s_very_big_index", corpus=corpus)

# Load the BM25 index as a memory-mapped file, which is memory efficient
# and reduce overhead of loading the full index into memory
retriever = bm25s.BM25.load("bm25s_very_big_index", mmap=True)
```

For an example of how to use retrieve using the `mmap=True` mode, check out `examples/retrieve_nq.py`.